FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Simulation and Control of Humanoid Robots for Applications in Unstructured Environments in Civil Construction

Maria Luís Silva Garrido



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Rafael Arrais

July 31, 2026

© Maria Garrido, 2026

Simulation and Control of Humanoid Robots for Applications in Unstructured Environments in Civil Construction

Maria Luís Silva Garrido

Mestrado em Engenharia Eletrotécnica e de Computadores

July 31, 2026

Resumo

A indústria da construção tem enfrentado desafios persistentes relacionados com a escassez de mão-de-obra e a baixa produtividade, com poucas expectativas de melhorias significativas num futuro próximo. Embora a automação tenha sido amplamente estudada como uma possível solução, a implementação de sistemas robóticos em locais de construção continua a ser um desafio devido à natureza dinâmica, não estruturada e perigosa destes ambientes. Muitas das plataformas robóticas existentes são ou específicas para tarefas concretas ou possuem mobilidade limitada, o que reduz a sua eficácia em ambientes de construção do mundo real.

Os robôs humanoides representam uma alternativa promissora, ao combinar locomoção semelhante à humana com capacidades versáteis de manipulação, permitindo a sua utilização em ambientes concebidos para trabalhadores humanos. O objetivo principal deste trabalho é criar, em simulação, as *skills* essenciais para que um robô humanoide seja capaz de instalar pavimento vinílico de forma autónoma. O robô humanoide Unitree H1 foi selecionado como plataforma devido à sua estabilidade, mobilidade e design antropomórfico.

A abordagem proposta será implementada no NVIDIA Isaac Sim, com o Robot Operating System 2 (ROS 2) a servir como framework para controlo e comunicação do robô. O trabalho centra-se no desenvolvimento e integração das funcionalidades essenciais de um robô humanoide, incluindo locomoção, manipulação e perceção dentro de um ambiente de construção simulado e não estruturado. Os resultados deste estudo têm como objetivo fornecer insights sobre o papel potencial dos robôs humanoides na resolução de desafios históricos de produtividade e mão-de-obra na indústria da construção, ao mesmo tempo que identificam limitações atuais e possíveis direções para investigação futura.

Abstract

The construction industry has faced persistent challenges related to labor shortages and low productivity, with little expectation of significant improvement in the near future. Although automation has been widely studied as a potential solution, the deployment of robotic systems on construction sites remains challenging due to the dynamic, unstructured, and hazardous nature of construction environments. Many existing robotic platforms are either task-specific or possess limited mobility, which reduces their effectiveness in real-world construction scenarios.

Humanoid robots represent a promising alternative by combining human-like locomotion with versatile manipulation capabilities, enabling deployment in environments designed for human workers. The main objective of this work is to create, in simulation, the essential skills that enable a humanoid robot to autonomously install vinyl flooring. The Unitree H1 humanoid robot is selected as the platform due to its stability, mobility, and anthropomorphic design.

The proposed approach will be implemented in NVIDIA Isaac Sim, with Robot Operating System 2 (ROS 2) serving as the framework for robot control and communication. The work focuses on the development and integration of core humanoid functionalities, including locomotion, manipulation and perception within a simulated, unstructured construction environment. The results of this study aim to provide insights into the potential role of humanoid robots in addressing long-standing productivity and labor challenges in the construction industry, while also identifying current limitations and directions for future research.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for addressing major societal, economic, and environmental challenges, promoting sustainable development worldwide. The 17 goals established by the United Nations aim to improve quality of life while encouraging technological innovation, environmental responsibility, and social progress.

The research developed in this dissertation contributes primarily to the objectives related to industrial innovation, sustainable construction, and the improvement of working conditions through robotic technologies. By investigating the simulation, control, and autonomous operation of a humanoid robot in construction environments, this work supports the development of more efficient and safer approaches for future industrial applications.

The specific SDGs addressed in this dissertation are:

SDG 3 Good Health and Well-being: Ensure healthy lives and promote well-being for all at all ages.

SDG 9 Industry, Innovation and Infrastructure: Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 8 Decent Work and Economic Growth: Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all.

SDG 11 Sustainable Cities and Communities: Make cities and human settlements inclusive, safe, resilient and sustainable.

SDG 12 Responsible Consumption and Production: Ensure sustainable consumption and production patterns.

SDG	Target	Contribution	Performance Indicators and Metrics
3	3.9	The development of humanoid robots for construction applications can reduce human exposure to hazardous, physically demanding, and repetitive tasks. By enabling robots to operate in environments designed for humans, this work contributes to improving occupational safety in the construction sector.	Evaluation of autonomous humanoid operation in a simulated construction environment. Assessment of the robot's ability to perform navigation tasks without human intervention.
9	9.4	The development of a humanoid robotic platform for construction applications contributes to the digital transformation of industrial processes. The integration of these robots supports the adoption of advanced robotic technologies in construction environments.	Successful integration of simulation, locomotion, navigation, and perception modules. Evaluation of autonomous robot operation in a simulated construction scenario.
8	8.8	The use of humanoid robots in construction has the potential to reduce human exposure to hazardous and physically demanding tasks while supporting workers in complex industrial environments.	Ability of the robot to autonomously perform navigation tasks in a construction-like environment.
11	11.3	The proposed framework contributes to the development of smarter and more automated construction processes by enabling robots to operate in environments designed for human use.	Validation of robotic operation inside a simulated building environment based on real construction layouts.
12	12.5	Simulation-based development and validation of robotic systems can reduce the need for physical prototypes and help optimize construction processes, potentially reducing material waste and improving resource efficiency.	Use of digital simulation tools for robot development and evaluation before physical deployment.

Acknowledgements

Esta dissertação representa o culminar do meu percurso académico, pelo que não poderia deixar de agradecer a todos aqueles que fizeram parte desta jornada e me acompanharam ao longo deste caminho.

Gostaria de agradecer ao meu orientador, Professor Rafael Arrais, e ao meu coorientador, Roberto Silva, pelo apoio, orientação e disponibilidade demonstrados ao longo deste trabalho. A partilha dos seus conhecimentos foi fundamental para o desenvolvimento desta dissertação.

À minha família, aos meus pais e ao meu irmão, agradeço por todo o apoio, por acreditarem em mim e por sempre me encorajarem e incentivarem a dar sempre o meu melhor.

Aos meus amigos mais antigos, obrigada por me acompanharem e apoiarem ao longo de todos estes anos. Porque todos os momentos convosco são bons momentos, mesmo quando discutimos enquanto jogamos Catan.

Aos meus amigos que conheci em Poznań, um obrigada por me terem feito sentir em casa e por terem tornado a experiência de estudar e viver fora tão divertida e especial.

Aos meus amigos, com quem tive a sorte de cruzar o meu percurso nesta faculdade em 021, agradeço por todos os momentos, por me fazerem sempre rir e pela vossa amizade. A vocês, que tornaram esta jornada mais fácil e estes cinco anos repletos de memórias que levarei para a vida.

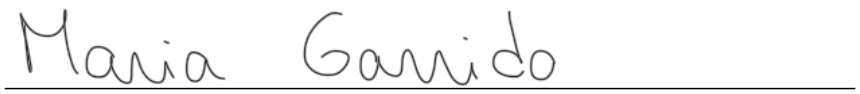
Por fim, agradeço ao Gonçalo, por ter partilhado este percurso comigo, por estar sempre presente e pelo apoio constante.

Maria Garrido

Declaration of honor

I declare that this dissertation is my original work and has not previously been submitted or assessed in any other course or curricular unit, whether at this or any other institution.
References to other authors (statements, ideas, thoughts), as well as the use of published works fully complies with the rules of attribution and are appropriately cited in the text and bibliography, in accordance with reference standards. I am aware that plagiarism and self-plagiarism constitute academic misconduct.
I further declare that I have not used Artificial Intelligence tools to produce any part(s) of this dissertation or, when such tools were used, their use and results are clearly indicated and were carefully reviewed for the detection of errors, inaccuracies, unfounded attributions, or other forms of content and argument bias, as well as for the determination of authorship.
I further declare that I have included studies carried out with multiple co-authors, having specified, with transparency and rigor, the contribution of each co-author, provided information on the possible existence of any other dissertation in which the same article has been included, reproduced the work in full with the permission of the journal in which it was published, and obtained the explicit written consent of all co-authors for the inclusion of the article.

Maria Luís Silva Garrido



x

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 2
- 1.3 Problem Statement . . . 2
- 1.4 Research Objectives . . . 2
- 1.5 Dissertation Structure . . . 4

2 State of Art **5**
- 2.1 Automation in Civil Construction . . . 5
- 2.2 Humanoid Robots . . . 7
  - 2.2.1 Current humanoid technology and market overview . . . 7
  - 2.2.2 Current Humanoid Models overview . . . 8
  - 2.2.3 Humanoid Robots as a Solution to Construction Industry Challenges . . . 11
  - 2.2.4 Safety Standards . . . 12
- 2.3 Application Areas of Humanoids in Construction . . . 13
  - 2.3.1 Hazards in Construction Activities . . . 13
  - 2.3.2 Task Selection . . . 14
  - 2.3.3 Vinyl Flooring Installation . . . 16
- 2.4 Simulation Framework . . . 16
  - 2.4.1 Simulation Environment . . . 16
  - 2.4.2 Humanoid Robot Model . . . 18

3 Methodology **21**
- 3.1 Task Coordination using the Open Scalable Production System . . . 21
- 3.2 Software Development Environment . . . 22
- 3.3 Simulation Environment . . . 22
  - 3.3.1 Environment Development . . . 22
  - 3.3.2 Environment Assets . . . 23
  - 3.3.3 Final Simulation Scenario . . . 24
- 3.4 Robot Configuration . . . 24
  - 3.4.1 Robot Platform and Sensor Configuration . . . 24
  - 3.4.2 Gripper Selection . . . 25
- 3.5 Object Detection Using YOLO . . . 26
  - 3.5.1 Synthetic Dataset Generation with Isaac Sim Replicator . . . 26
  - 3.5.2 Automatic Annotation and Dataset Conversion . . . 27
  - 3.5.3 YOLO Training Configuration . . . 28
  - 3.5.4 Object Detection Skill . . . 29
- 3.6 Reinforcement Learning for Humanoid Manipulation . . . 31

3.6.1 Simulation Environment . . . 31
3.6.2 Observation Space . . . 32
3.6.3 Action Space . . . 32
3.6.4 Suction Grasp Model . . . 33
3.6.5 Reward Function . . . 33
3.6.6 Adaptive Curriculum Learning . . . 33
3.6.7 Policy Training . . . 34
3.7 Locomotion and Navigation . . . 34
3.7.1 Reinforcement learning locomotion controller . . . 35
3.7.2 Isaac Sim and ROS 2 integration . . . 36
3.7.3 Navigation strategies . . . 38
3.7.4 The WalkSkill action interface . . . 39
3.7.5 Autonomous navigation using Nav2 . . . 40
3.7.6 Relative motion strategy . . . 44

4 Results and Discussion **47**
4.1 Perception Results . . . 47
4.1.1 Training Outcome . . . 47
4.1.2 Detection Performance in Simulation . . . 49
4.1.3 Discussion . . . 52
4.1.4 Limitations . . . 53
4.2 Grasping Manipulation Results . . . 53
4.2.1 Qualitative observations . . . 54
4.2.2 Discussion . . . 55
4.2.3 Limitations . . . 56
4.3 Locomotion and Navigation Results . . . 56
4.3.1 Experimental protocol . . . 56
4.3.2 Straight line . . . 57
4.3.3 Rotation in place . . . 57
4.3.4 Diagonal route . . . 58
4.3.5 Discussion . . . 60
4.3.6 Limitations . . . 62

5 Conclusions and Future Work **65**
5.1 Answering the Research Questions . . . 65
5.2 Future Work . . . 68
5.2.1 Consolidation of the Experimental Evaluation . . . 68
5.2.2 From Two-Dimensional Detection to Metric Localization . . . 69
5.2.3 Integration of Manipulation with Mobile Locomotion . . . 69
5.2.4 Autonomous Vinyl Flooring Installation . . . 69

References **71**

List of Figures

2.1 Humanoid robot companies, makers, and technology-enabled applications. [25] . 7
2.2 H1 humanoid robot from Unitree. [71] 18
2.3 H1 main body structure. [73] 19

3.1 Environment model in Autodesk Revit. 23
3.2 Custom vinyl plank model in Blender. 24
3.3 Environment in NVIDIA Isaac Sim. 25
3.4 Hybrid gripper configuration in H1 model. 26
3.5 Batch of training images reconstructed by the Ultralytics framework from the converted YOLO format annotations. 28
3.6 ROS 2 computation graph for `/rgb` camera topic publishment. 30
3.7 Action Graph responsible for the joint state interface. 36
3.8 Action Graph responsible for odometry. The pose of the robot is computed from the simulated rigid body state and published both as a `nav_msgs/Odometry` message and as the transformation from `odom` to `base_link`. 38
3.9 Occupancy grid used for navigation, obtained from Isaac Sim. 41

4.1 Evolution of the loss components and of the evaluation metrics during the 100 training epochs. The top row corresponds to the training subset and the bottom row to the validation subset. 47
4.2 Normalized confusion matrix obtained on the validation subset. Columns correspond to the annotated class and rows to the predicted class. 49
4.3 Detections obtained with the robot positioned at the far end of the working area. 50
4.4 Detection of the work table with the robot positioned at an intermediate distance and facing the object directly. 51
4.5 Detection of the vinyl plank resting on the work table with the robot positioned at close range. 51
4.6 Execution of the learned grasping policy in Isaac Lab. 55
4.7 The diagonal route being executed under Nav2. The Isaac Sim viewport shows the robot in the simulated environment, while RViz shows the same situation as represented by the navigation stack. 58

List of Tables

2.1 Overview and comparison of representative humanoid robots, including their main technical characteristics and application domains. . . . 9
2.2 Construction activities suitable for humanoid robot automation, associated advantages, and challenges. . . . 15

3.1 Fields of the `FindobjectSkill` action. . . . 29
3.2 Main PPO training parameters. . . . 34
3.3 Computational platform used for the experiments. . . . 35
3.4 Main ROS 2 interfaces between Isaac Sim and the locomotion stack. . . . 36
3.5 Fields of the `WalkSkill` action. . . . 39
3.6 Main Nav2 parameters adapted for humanoid locomotion. . . . 42
3.7 Main parameters of the relative motion controller. . . . 46

4.1 Confidence values obtained for each class as a function of the robot position. . . . 52
4.2 Manipulation performance over 2048 parallel simulation environments and 5000 iterations. . . . 54
4.3 Influence of robot stance on grasp quality. . . . 55
4.4 Straight translation of 2 m. Mean $\pm$ standard deviation over five trials, with the observed range in brackets. . . . 57
4.5 In-place rotation of 90°. Mean ± standard deviation over five trials, with the observed range in brackets. . . . 58
4.6 Diagonal displacement of 3.54 m with curvature to the right. Mean ± standard deviation over five trials, with the observed range in brackets. . . . 59
4.7 Diagonal displacement of 3.54 m with curvature to the left. Single trial per strategy. 59

List of Acronyms

AI Artificial Intelligence
AMCL Adaptive Monte Carlo Localization
BIM Building Information Modeling
CAD Computer-Aided Design
COCO Common Objects in Context
CPU Central Processing Unit
DoF Degrees of Freedom
glTF GL Transmission Format
GPU Graphics Processing Unit
IEEE Institute of Electrical and Electronics Engineers
IFC Industry Foundation Classes
ISO International Organization for Standardization
LiDAR Light Detection and Ranging
LTS Long Term Support
LVT Luxury Vinyl Tile
MEP Mechanical, Electrical and Plumbing
OSPS Open Scalable Production System
PPO Proximal Policy Optimization
RGB Red, Green and Blue
RGB-D Red, Green, Blue and Depth
RL Reinforcement Learning
ROS 2 Robot Operating System 2
RPP Regulated Pure Pursuit
RQ Research Question
SDF Simulation Description Format
SO Specific Objective
SOP Standard Operating Procedure
TRL Technology Readiness Level
URDF Unified Robot Description Format
USD Universal Scene Description

YOLO You Only Look Once

Chapter 1

Introduction

1.1 Context

The construction sector is a fundamental component of the economy, yet it has faced persistent labor shortages since 2010. Job vacancy rates in the sector have increased by over two percentage points during this period, reflecting a demand for labor that exceeds the sector's capacity to attract workers. This challenge is further amplified by growing needs for housing and public infrastructure [1]. Several structural and demographic factors contribute to this shortage. A significant portion of the experienced workforce is approaching retirement, while overall labor force growth in advanced economies and China are projected to remain near zero or even negative through 2040 [2]. At the same time, younger workers are often discouraged from entering the sector due to its physically demanding nature, safety risks, and uncertainty regarding long-term career prospects [3].

Globally, productivity growth in construction has remained largely stagnant over the past decades. Between 2000 and 2022, labor productivity increased by only 10%, approximately 0.4% per year, while productivity in the broader economy grew at around 2% annually [2]. This widening performance gap highlights structural challenges unique to the sector. More recently, global construction productivity declined by about 8% between 2020 and 2022, suggesting that these issues are not merely cyclical [2].

Automation has been widely recognized as a potential means to address both labor shortages and stagnant productivity [2], [3]. Although several automation solutions already exist, including single-task autonomous or semi-autonomous robots designed to perform specific operations, these systems have not yet achieved large-scale deployment on construction sites and often face limitations related to mobility and adaptability [3], [4]. As a result, most automation efforts remain concentrated off-site, particularly in the manufacturing of precast and prefabricated components such as walls, floors, and beams [3]. In contrast, on-site automation remains limited due to the unstructured and dynamic nature of construction environments, which complicates the safe and reliable deployment of conventional automation robots in the field [2], [3].

1.2 Motivation

The limitations of current construction automation motivate the exploration of more versatile robotic solutions capable of operating directly on construction sites. Unlike conventional industrial robots, which are typically designed for structured and repetitive tasks, humanoid robots combine mobility, perception, and manipulation within a single platform, enabling them to operate in environments originally designed for human workers.

Their human-like morphology allows them to traverse uneven terrain, use the same circulation paths and access points as construction workers, and operate existing hand-held tools without requiring major modifications to current construction practices [3], [5]. Furthermore, their potential ability to perform a wide variety of physically demanding and repetitive tasks makes them a promising complement to the human workforce, rather than a replacement for it [3], [4], [5].

Despite these advantages, the deployment of humanoid robots in construction remains at an early stage. At the moment, relatively few studies investigate their integration into complete workflows representative of real construction activities [3], [5]. This motivates the development and evaluation of integrated robotic frameworks that combine these capabilities within realistic construction scenarios, helping to assess the feasibility of humanoid robots for future on-site applications.

1.3 Problem Statement

While the potential of humanoid robots to address labor and productivity challenges in construction has been widely discussed, most existing studies and platforms focus on isolated capabilities rather than on their integration into a complete, task-oriented pipeline. In particular, there remains limited understanding of how to design and train manipulation policies for non-standard, task-specific end-effectors, such as a suction-based gripper adapted to a humanoid platform, since most available humanoid manipulation research and tooling assumes dexterous, multi-fingered hands.

Although simulation platforms provide the tools necessary to model robots, environments, and learning pipelines, there is a lack of documented, end-to-end case studies that combine scene modeling, robot adaptation, reinforcement learning-based manipulation, object perception, and task orchestration into a single validated workflow for a representative construction task.

This dissertation addresses this gap by investigating the development of the core capabilities required for autonomous construction tasks.

1.4 Research Objectives

This dissertation aims to integrate concepts from robotics, automation, control systems, simulation, and modeling to develop and evaluate the core capabilities required for humanoid robots operating in construction-like environments. The work focuses on the development of autonomous locomotion, perception, and manipulation capabilities required for on-site material handling. These components

are evaluated through a representative case study based on the vinyl plank flooring installation. To this end, the Unitree H1 humanoid robot equipped with a custom suction gripper and a dexterous hand is adopted as the experimental platform, while Robot Operating System 2 (ROS 2) and NVIDIA Isaac Sim provide the software infrastructure for system development and validation.

The work is centered on the following Research Questions (RQs):

- **RQ1:** How can autonomous locomotion and navigation be integrated to enable a humanoid robot to safely navigate a realistic construction environment?
- **RQ2:** How can reinforcement learning be used to enable reliable manipulation of thin vinyl flooring planks?
- **RQ3:** How can vision-based object detection be integrated into a humanoid robotic platform?
- **RQ4:** How can locomotion, perception, and manipulation capabilities be combined to support representative construction tasks in a simulated environment?

To address the research questions presented above, the following Specific Objectives (SOs) are defined:

- **SO1:** Develop a realistic construction simulation environment.
- **SO2:** Configure and adapt the Unitree H1 humanoid robot by integrating perception sensors and a hybrid end-effector configuration suitable for material handling tasks.
- **SO3:** Implement a perception module based on the You Only Look Once (YOLO) algorithm for the detection of construction-related objects.
- **SO4:** Develop and train a reinforcement learning policy for autonomous manipulation of vinyl flooring planks.
- **SO5:** Develop and evaluate autonomous navigation and relative motion control for humanoid locomotion using ROS 2 and Nav2.

By achieving these objectives, this dissertation contributes to the research on humanoid robotics for construction by providing a representative case study for the development and evaluation of locomotion, perception, and manipulation capabilities in a realistic simulated construction environment. The work demonstrates the application of ROS 2, NVIDIA Isaac Sim, reinforcement learning and computer vision to representative construction tasks. Beyond the specific application of vinyl flooring installation, the developed methodologies provide a foundation for future research towards more integrated and autonomous humanoid robotic systems for construction.

1.5 Dissertation Structure

In addition to the introduction, this dissertation contains four additional chapters.

Chapter 2 presents the state of the art and discusses the main research areas related to construction automation, humanoid robotics, their application to construction, and the simulation frameworks adopted for robotic development.

Chapter 3 describes the methodology adopted in this work, including the simulation environment, the Unitree H1 humanoid platform, the integration between NVIDIA Isaac Sim and ROS 2, the perception module, the reinforcement learning-based manipulation, and the proposed locomotion and navigation approaches.

Chapter 4 presents the experimental results obtained from the developed system.

Finally, Chapter 5 summarizes the main conclusions of this dissertation and discusses possible directions for future work.

Chapter 2

State of Art

2.1 Automation in Civil Construction

Robotic systems are increasingly integrated across the construction lifecycle, from structural assembly to interior finishing. In the structural phase, large-scale automation solutions such as COBOD's BOD2 [6] and FBR's Hadrian X [7] enable gantry-based 3D concrete printing and robotic masonry, respectively, reducing labor intensity and site congestion.

Structural preparation and layout tasks are also increasingly automated. TyBot [8] autonomously identifies and ties rebar intersections in bridge decks and foundations, while Dusty Robotics [9] employs a mobile printing robot using Building Information Modeling (BIM) data, computer vision, and Light Detection and Ranging (LiDAR) to produce high-precision layouts directly on concrete slabs [10].

For Mechanical, Electrical and Plumbing (MEP) activities, mobile robotic manipulators mainly act as semi-autonomous assistants. Hilti's Jaibot [11] and Fischer's BauBot [12] leverage BIM data for precise marking, drilling, and anchor installation, particularly for repetitive or physically demanding overhead tasks [10]. Schindler's R.I.S.E. [13] self-climbing robotic system drills and installs anchor bolts in elevator shafts with significantly higher efficiency than manual methods.

Interior construction and finishing tasks are also increasingly automated. Canvas [14] automates drywall finishing, while OKIBO [15] performs plastering and painting using 3D scanning and mapping [10]. Bright Dream Robotics [16] has developed a portfolio of robots addressing surface finishing, wall grinding, painting, floor coating, concrete operations, and site measurement, often integrated with BIM-based digital workflows to enable coordinated multi-robot deployment [17]. Built Robotics [18] extends automation to earthmoving by retrofitting conventional heavy machinery for autonomous operation.

These robots are not limited to experimental research prototypes, but rather constitute operational solutions with a high Technology Readiness Level (TRL), having already been deployed in real-world scenarios. However, despite these advances, mobility remains a critical limitation. Construction sites are dynamic, unstructured environments with uneven terrain, elevation changes, and constantly changing obstacles. Most mobile robotic systems are constrained to flat surfaces

or predefined paths, limiting their applicability. Quadruped robots, such as Boston Dynamics' Spot [19], exhibit superior terrain adaptability and dynamic stability, allowing them to traverse uneven surfaces, climb stairs, and navigate cluttered sites. Equipped with advanced perception systems, they excel in autonomous site inspection, 3D mapping, and as-built documentation, supporting progress monitoring [10]. However, quadrupeds are primarily designed for sensing and data collection and cannot perform tasks that involve material handling or physical labor.

Beyond robotic platforms, BIM has emerged as a key enabling technology for digital construction. BIM is a transformative technology that involves the creation and management of intelligent, data-rich 3D models of a building or infrastructure project. It serves as a shared digital knowledge resource, providing a reliable representation of the physical and functional characteristics of a facility throughout its entire lifecycle. In practice, BIM acts as a centralized repository, containing not just geometric shapes but also rich semantic information. This includes material specifications, spatial relationships, cost data, scheduling, and logistics information [20]. These characteristics make BIM a fundamental foundation for developing digital twins, as it provides accurate and structured design data that can be continuously updated with real-time information. This integration enables monitoring and comparison between the designed and the actual state of the project, supporting more efficient operation, maintenance, and decision-making processes.

The integration of BIM with robotics, particularly humanoid robotics, represents a paradigm-shifting opportunity to address key challenges in the construction industry, offering a significant step toward bridging the gap between digital planning and physical execution. However, the integration of BIM with ROS still faces significant challenges. Although both frameworks are highly advanced within their respective domains, there are currently no standardized interfaces or middleware for direct data exchange between them. This lack of interoperability limits the full exploitation of BIM's potential in robotic applications, constraining automation workflows that rely on spatial and semantic information. One method for communication between BIM and robotic systems is through Industry Foundation Classes (IFC). This open format allows for the exchange of model data between different software platforms and has been utilized to convert BIM models into Simulation Description Format (SDF) for navigation and robot simulation, as well as into Unified Robot Description Format (URDF) to represent building environments and expose dynamic attributes for task planning [21]. Some studies have implemented this open format in robotics applications. Kim and Peavy [22] developed a BIM-based semantic building world in URDF using IFC, enabling a robot to directly query building information for task planning and execution. Kim et al. [23] focused on an interior wall painting scenario, where BIM construction models in IFC 2×3 were transformed into SDF for simulation within the Gazebo environment. Yet, a significant challenge persists since the data exchange via IFC often focuses on basic geometric information, leaving out essential semantic details [21].

With the right integration framework, BIM can provide robots with detailed spatial data, which can be converted into 2D grid maps or 3D voxel representations, enabling autonomous navigation. The model can inform the robot not only about permanent structures but also about semantic information, such as identifying fragile materials to avoid. In tasks like flooring installation,

BIM can provide robots with the precise geometric and semantic data required. This includes a component's dimensions, exact location, and its connection relationships with other elements. This data allows robots to plan and execute complex assembly sequences with high precision, optimizing the construction process.

2.2 Humanoid Robots

2.2.1 Current humanoid technology and market overview

Recent advances in robotics and Artificial Intelligence (AI) have enabled the development of increasingly sophisticated robotic systems capable of operating in environments designed for humans. In this context, humanoid robots are engineered to replicate human body structure and behavior, including perception, decision-making, and interaction capabilities [24]. By adopting human-like proportions and physical characteristics, these robots can perform tasks typically carried out by humans. Consequently, humanoid robotics has emerged as a rapidly evolving and promising research area within the broader field of robotics.

Numerous companies are actively developing humanoid robots for a wide range of applications. Figure 2.1 provides an overview of the main organizations currently involved in humanoid robotics, either as robot manufacturers or as developers of enabling technologies.

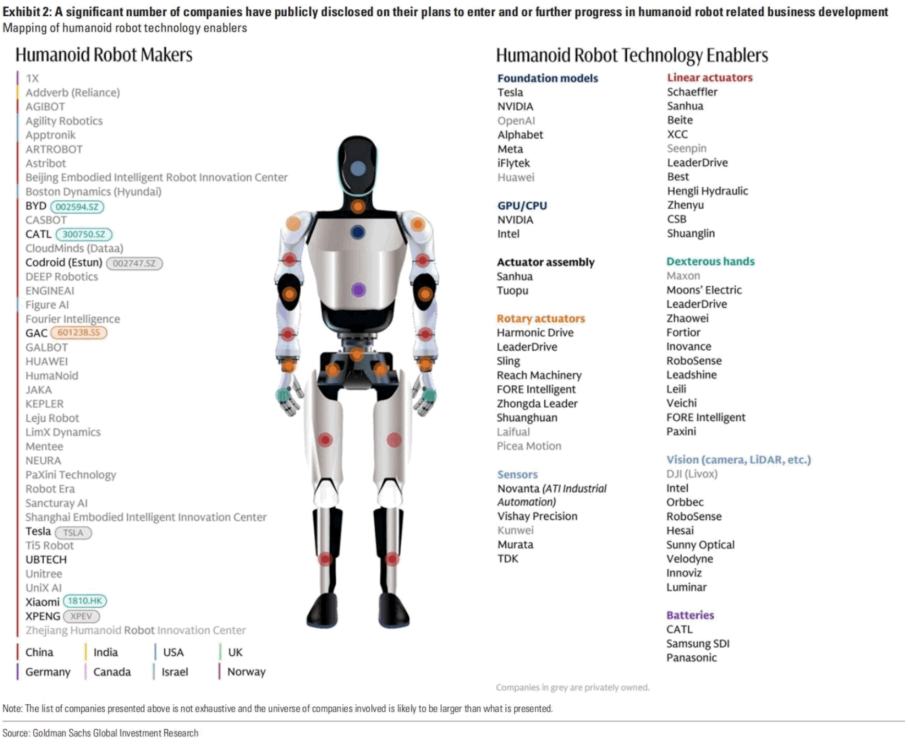


Figure 2.1: Humanoid robot companies, makers, and technology-enabled applications. [25]

From the analysis of Figure 2.1, it can be observed that China plays a leading role in humanoid robot development, with several companies strongly invested in this field, including AgiBot [26],

Fourier Intelligence [27], and Unitree [28]. The United States also shows a significant presence, hosting companies responsible for some of the most advanced humanoid platforms currently under development, such as Figure AI [29], Boston Dynamics [19], Apptronik [30], and Agility Robotics [31].

Beyond robot manufacturers, the advancement of humanoid robotics is strongly supported by a broad ecosystem of technology enablers that supply critical hardware and software components. For instance, NVIDIA [32] and Intel [33] supply Graphics Processing Units (GPUs) and Central Processing Units (CPUs) that enable advanced perception and AI computations, while Sanhua [34], Harmonic Drive [35], and LeaderDrive [36] provide actuators, actuator assemblies, and rotary mechanisms essential for precise joint control. Dexterous hands and grippers are developed by companies such as Maxon [37] and Moons' Electric [38], enabling robots to manipulate objects. Sensor technologies, including cameras, LiDAR, and other perception devices, are supplied by firms such as DJI [39], Intel [33], Orbbec [40], and RoboSense [41]. Battery solutions from CATL [42], Samsung SDI [43], and Panasonic [44] support the operational autonomy of these platforms.

2.2.2 Current Humanoid Models overview

In recent years, advances in robotics, AI, and mechatronics have driven rapid progress, leading to the continuous emergence of new humanoid robots and enabling technologies. These systems are designed with diverse objectives and capabilities, addressing a broad range of applications, from domestic assistance to industrial and construction-related tasks.

Table 2.1 presents a compilation of some of the most well-known and notable humanoid robots currently under development or deployment. The selection was based on an extensive review of existing humanoid platforms, scientific literature, and recent technological developments reported in academic publications and industry sources.

In the domestic domain, humanoid robots such as NEO, developed by 1X Technologies, exemplify the integration of automation into everyday life. NEO is designed to perform household chores, including cleaning, organizing living spaces, managing laundry, and watering plants, while also serving as a personal companion. Its primary goal is to enhance convenience and free human time for other activities.

Table 2.1: Overview and comparison of representative humanoid robots, including their main technical characteristics and application domains.

Robot	Company / Model	Technical Characteristics	Application Field
	Unitree Robotics, H1 (China) [45]	Height: 1.80 m; Weight: 47 kg; Degrees of Freedom (DoF): 19; Speed: 3.3 m/s	General-purpose humanoid for research and industrial automation
	Unitree Robotics, G1 (China) [46]	Height: 1.32 m; Weight: 35 kg; DoF: 23; Speed: 2.0 m/s	Research and general-purpose industrial tasks, including palletizing, handling, assembly, and welding
	AgiBot, AgiBot A2 (China) [47]	Height: 1.69 m; Weight: 69 kg; DoF: 40; Speed: 1.2 m/s	Entertainment and commercial performance in public spaces
	Fourier Intelligence, GR-2 (China) [48]	Height: 1.75 m; Weight: 65 kg; DoF: 53; Speed: 1.39 m/s	General-purpose service and industrial tasks
	Sanctuary AI, Phoenix (Canada) [49]	Height: 1.70 m; Weight: 70 kg; DoF: 75; Speed: 1.34 m/s	General-purpose work across retail, logistics, manufacturing, and service environments
	Boston Dynamics, Atlas (USA) [50]	Height: 1.50 m; Weight: 89 kg; DoF: 28; Speed: 2.5 m/s	Advanced R&D platform for dynamic locomotion, industrial inspection, construction-related trials, and search-and-rescue research

continued on next page

Table 2.1 – continued from previous page

Robot	Company / Model	Technical Characteristics	Application Field
	Figure AI, Figure 03 (USA) [51]	Height: 1.68 m; Weight: 60 kg; DoF: 41; Speed: 1.2 m/s	General-purpose industrial labor in manufacturing, logistics, and domestic environments
	Agility Robotics, Digit (USA) [52]	Height: 1.75 m; Weight: 65 kg; DoF: 28; Speed: 1.5 m/s	Logistics and warehouse automation
	Apptronik, Apollo (USA) [53]	Height: 1.73 m; Weight: 72 kg; DoF: 71; Speed: 0.94 m/s	Industrial manufacturing and intralogistics tasks
	1X, Neo (Norway) [54]	Height: 1.68 m; Weight: 30 kg; DoF: 22; Speed: 1.4 m/s	Domestic assistance
	Tesla, Optimus Gen 2 (USA) [55]	Height: 1.73 m; Weight: 56 kg; DoF: 22; Speed: 2.22 m/s	General-purpose humanoid for repetitive and physically demanding tasks in manufacturing and logistics

In contrast, several humanoid robots are being developed for industrial and labor-intensive applications. Examples include Apollo, from Apptronik, and Atlas, developed by Boston Dynamics. These robots operate in environments such as warehouses, logistics centers, and industrial facilities. They combine human-like mobility and manipulation with robotic endurance and precision, positioning humanoid systems as a potential solution to labor shortages and productivity challenges in physically demanding industries.

Some humanoid robots are designed for multipurpose functions, capable of operating in both domestic and industrial settings. An example is Figure 03, from Figure AI, which demonstrates versatility across multiple applications.

2.2.3 Humanoid Robots as a Solution to Construction Industry Challenges

The introduction of humanoid robots into the construction industry represents a promising approach to addressing long-standing challenges in the sector such as persistent shortages of skilled labor and low productivity levels. Construction projects are often characterised by repetitive, physically demanding, and occasionally hazardous tasks. In this context, humanoid robots offer a viable and innovative solution. Owing to their human-like morphology and ability to perform tasks that typically require human dexterity, these robots can operate in environments originally designed for human workers [56]. Another key advantage of humanoid robots is their multipurpose nature. Unlike many existing robotic solutions that are designed for a single, specialized task, humanoid robots can perform a wide range of activities using tools and equipment already employed by human workers. [3] This versatility allows them to adapt to different phases of construction projects, thereby enhancing their overall utility and cost-effectiveness. Furthermore, as humanoid robot costs continue to decrease while labor costs rise, the price of a humanoid robot can be comparable or even lower than the annual cost of a human worker [5].

In addition to their physical versatility, recent advances in AI have further expanded the capabilities of humanoid robots. Although significant development challenges remain, humanoid robots can become highly capable when supported by embodied AI, which enables rapid decision-making and integrates vision–language–action foundation models to interpret visual cues and follow spoken instructions, thereby enhancing their effectiveness in complex construction environments [3].

Building on these characteristics, humanoid robots present several advantages when applied to the construction industry. According to Ikuabe et al. [56], the main benefits include:

- **Improved safety on site:** Humanoid robots can be deployed in hazardous environments to perform dangerous tasks, thereby reducing the risk of accidents and injuries for human workers.
- **Improved working conditions:** Construction activities are often associated with poor ergonomic conditions and exposure to dust, chemicals, and noise. Assigning such tasks to humanoid robots can significantly reduce physical strain on human workers and improve overall working conditions.

- **Increased quality and accuracy of construction outputs:** As automated systems, humanoid robots can provide high precision and consistent quality when compared to human labor, which may be affected by fatigue and human error. Additionally, unlike humans, robots can continuously monitor task execution without compromising on performance, enabling earlier detection of defects or inconsistencies.

- **Reduced project duration and enhanced efficiency:** Humanoid robots are capable of continuous operation, subject to battery limitations and technical or maintenance-related constraints, without the need for breaks or shift rotations, thereby enabling faster task execution. By assuming repetitive and time-consuming tasks, they enable skilled human workers to dedicate their efforts to more complex activities that require higher levels of expertise. Consequently, project timelines are shortened, operational efficiency is improved, and labor costs are reduced.

2.2.4 Safety Standards

Humanoid robots not only mimic human morphology but are intended to operate within human environments and work alongside humans. For this reason, the existence of appropriate safety standards is of critical importance. Currently, the International Organization for Standardization (ISO) has established several standards addressing robot safety. However, these are primarily focused on industrial robotic arms and collaborative robots. Such standards do not adequately account for the inherent instability and uncertainty associated with autonomous humanoid robots, particularly with respect to their locomotion and dynamic interaction with the environment. To address these limitations, specific safety guidelines for humanoid robots are currently under development, namely ISO 25785-1 [57].

This issue has become particularly relevant as the humanoid robotics market continues to evolve, reinforcing the need for comprehensive safety standardization manufacturers require clear and unified guidelines to ensure that their robotic systems operate safely, while consumers depend on standardized certifications to be confident that the products they acquire are both safe and effective. In the absence of such standards, technological development may either progress without adequate safety assurances or stagnate due to uncertainty and regulatory hesitation [58].

In the absence of dedicated safety standards for humanoid robots, several studies have already addressed this topic. For instance, an Institute of Electrical and Electronics Engineers (IEEE) study group has published a framework for humanoid robot safety [58] with the objective of supporting the development of future standards. This framework mainly focuses on three key aspects: classification, stability, and human–robot interaction. Regarding classification, the study highlights the need for a humanoid classification system based on physical capabilities, such as locomotion, dexterity, and sensory systems, as well as behavioral complexity, levels of autonomy and adaptability, and application domains, since each category is exposed to different types of risks. In the stability section, performance metrics and safety validation methodologies are proposed, as stability is considered one of the most significant barriers to the widespread deployment of humanoid robots.

Lastly, the framework addresses human–robot interaction by analyzing psychosocial, ethical, and functional factors that are essential for guiding the design, testing, and integration of humanoid robots into human-centered environments.

Building on the ongoing efforts toward standardization, several technical studies have proposed methodologies and tools to address humanoid robot safety. An example is SPARK [59], a benchmarking framework developed to enhance the safety of humanoid robots. Featuring a modular architecture, the framework incorporates configurable safe control algorithms, allowing users to adjust safety levels in a manner that balances performance and risk. This makes it possible to adapt to a wide variety of tasks and operational scenarios. In addition, SPARK provides a simulation environment that includes numerous safe control tasks, enabling testing and validation within a controlled environment.

Additionally, Scianca et al. [60] proposed a behavior-based control architecture with particular emphasis on locomotion, defining three categories of safety behaviors: override, temporary override, and proactive. Based on triggers from the robot's sensor system, a state machine manages these behaviors, handling their activation and deactivation. Wang et al. [61] proposed a more recent end-to-end approach that formulates humanoid locomotion as a learning problem, explicitly incorporating safety and human comfort through constraints on collisions, stability, and interpersonal distance.

Together, these works provide practical tools, metrics, and validation methodologies that can help with the creation of upcoming humanoid safety standards.

2.3 Application Areas of Humanoids in Construction

This study explores the use of humanoid robots in the construction sector using simulation, focusing on a task that could particularly benefit from automation. The selection process involved a comprehensive review of typical construction activities, followed by an analysis of the main hazards to which workers are exposed in the field, in order to identify tasks most suitable for robotic automation. The candidate tasks were then evaluated based on the capabilities and limitations of humanoid robots, ensuring that the tasks chosen for simulation are both practical and representative of real-world construction scenarios.

2.3.1 Hazards in Construction Activities

To evaluate potential tasks for humanoid automation, an analysis of the main hazards associated with construction activities was conducted. This was essential to identify tasks that not only have the potential of improving the efficiency and quality of construction processes but also to enhance safety conditions on site by reducing human exposure to hazardous situations. In Ikuabe et al. [56] research on construction safety, the hazards were classified into the following categories.

- **Primary Hazards:** They are caused by the nature of construction sites and are characterised by constantly changing environments and constant movement of people and machinery. This includes risks such as falls from heights, electrocution, struck-by incidents, and slips, trips,

and falls at the ground level and caught-in between incidents. These commonly result in severe injuries or even death.

- **Physical Hazards:** They involve the exposure to forms of energy or environmental conditions that can cause harm to workers. This includes exposure to noise, extreme temperatures, light and visibility and fire. These can lead to health issues or making them more vulnerable.
- **Chemical Hazards:** These arise from exposure to harmful substances through inhalation or skin contact, which can cause both immediate and long-term health problems. Examples include use of hazardous and airborne materials, thermal and chemical burns, and asbestos exposure.
- **Ergonomic and Other Hazards:** They are associated with repetitive movements, improper lifting, handling of materials and equipment, and unsafe postures. They can result in musculoskeletal disorders and other chronic injuries. Additional risks include the improper use of scaffolds, heavy machinery, and temporary structures, which may lead to falls, crush injuries, or other accidents.

2.3.2 Task Selection

Based on the previous analysis, a use case study was carried out by selecting a set of construction tasks that could benefit from humanoid robot automation. These tasks are summarised in Table 2.2. Tasks that exceeded the physical capabilities of the robot such as lifting very heavy objects, performing highly precise manipulations, or requiring extensive mobility were excluded from the selection.

Table 2.2: Construction activities suitable for humanoid robot automation, associated advantages, and challenges.

Construction Activity	**Advantages of Automation**	**Main Challenges**
Steel Rebar Assembly	Reduction of ergonomic strain due to non-ergonomic postures; decreased risk of cuts and abrasions when handling steel bars	Handling rigid materials; accurate positioning of reinforcement elements
Welding	Reduced exposure to fumes, radiation, and intense light; High risk of burns and eye injuries;	High precision requirements
Painting	Minimized exposure to chemical solvents; reduction of repetitive motions and forced postures	Surface quality control; adaptation to uneven or complex surfaces
Vinyl Flooring Installation	Reduced repetitive work in kneeling or bent positions; lower ergonomic stress on joints and back	High precision and alignment requirements; Robot positioning strategy to enable installation
Ceiling Installation	Reduced physical strain from overhead work; lower risk of injury from falling materials	Operation on elevated platforms; balance and reach limitations
Sanding	Reduced exposure to fine dust particles; elimination of prolonged hand–arm vibration	Vibration management; consistent surface finishing
Carpentry (Sawing, Hammering, Drilling)	Reduction of repetitive manual tasks; decreased risk of injuries from sharp tools and impacts	High precision; vibration control; complex tool handling

TThis topic was later discussed with a construction industry expert to validate the candidate tasks and ensure their relevance to real-world construction scenarios. From the feedback received, it was concluded that the activities of most interest were steel rebar assembly, sanding, and vinyl flooring installation. Among these, vinyl flooring installation was selected for further development in simulation. Its workflow can be decomposed into navigation, detection, and manipulation of objects like vinyl planks, allowing a robot to cover all steps of the process. Additionally, it combines challenges in mobility, manipulation, and precision, making it a balanced benchmark for evaluating the capabilities of humanoid robots in construction settings. Finally, the task is performed in an ergonomically unfavourable posture and is highly repetitive, which reinforces the interest in automating it.

2.3.3 Vinyl Flooring Installation

Vinyl flooring installation is a recurrent interior finishing task in the construction sector. The process was studied in detail to enable an accurate simulation of its execution by a humanoid robot. To this end, multiple application manuals [62], [63], [64], [65] were reviewed to identify the required tools and materials, as well as to formalize the sequence of operations involved in the installation procedure. Since different brands and types of vinyl flooring may have specific installation requirements, the study focused on the general steps common to most vinyl flooring products.

2.3.3.1 Tools, Material and Installation Steps

The main tools required for vinyl flooring installation include X-Acto knife, rubber hammer, tapping block and adhesive tape. Materials needed for the installation include vinyl planks, underlayment, and spacers.

The installation of vinyl flooring typically starts with the preparation of the subfloor, which involves cleaning to ensure proper adhesion of the vinyl planks. Next, an underlayment is laid down and secured with tape and the spacers are placed along the walls. The vinyl planks are then placed starting from one corner of the room, ensuring a tight fit between adjacent planks. Due to the need for precise cutting of some vinyl planks to fit in the room dimensions, the use of an X-Acto knife is required. The planks are put in place using a rubber hammer and tapping block to ensure a secure fit. The process continues until the entire floor area is covered. To finish, the skirting boards and transition profiles are placed.

2.4 Simulation Framework

This section presents the simulation framework adopted in this work and the criteria used for its selection. The framework consists of three main components: a physics-based simulation environment, a robotic middleware, and a humanoid robot platform. Together, these components provide the infrastructure required to develop, integrate, and evaluate the perception, locomotion, and manipulation capabilities investigated in this dissertation.

2.4.1 Simulation Environment

The selection of an appropriate simulation environment is crucial for accurately modeling the dynamics and interactions of humanoid robots in construction tasks. For this project, some of the most used robotics simulators were considered, most notably Gazebo and Isaac Sim.

Gazebo is an open-source robotics simulator that offers a robust physics engine, extensive sensor models, and a large community of users. It is widely used in academia and industry for simulating various robotic systems, including humanoid robots. Its integration with the ROS makes it a popular choice for robotics research and development [66].

Isaac Sim, developed by NVIDIA, is a high-fidelity simulation platform that leverages NVIDIA's Omniverse technology. It provides advanced rendering capabilities, realistic physics simulation, and support for AI and machine learning applications. Isaac Sim is particularly well-suited for simulating complex environments and interactions, making it a strong candidate for construction-related tasks [67].

Zwingel et al. [68] conducted a comparative analysis of Gazebo and Isaac Sim for robotics simulation. To evaluate both simulators, the authors considered hardware requirements, the degree of physics simulation, graphics simulation, and connectivity with other frameworks. It was also examined interfaces between ROS, Gazebo, and Isaac Sim, as well as sensor simulation and the simulation of moving objects and humans.

The study concluded that both simulators have distinct strengths and weaknesses. Gazebo is an open-source simulator widely used with ROS for mobile robots, offering accurate physics simulation, but the authors found its graphics quality to be limited compared with Isaac Sim. Isaac Sim excels in high-quality graphics and rendering, although it requires more hardware resources and, at the time, showed limitations in sensor simulation. The authors also emphasized that Isaac Sim is a more modern platform with significant potential for future development. Since the study was published in 2022, Isaac Sim has evolved considerably, addressing previous weaknesses, particularly in sensor simulation.

A more recent article from 2024 in Medium [69] further emphasizes Isaac Sim's photorealistic rendering capabilities. It notes that "for the perception algorithms to work reliably in the real world, the rendered environment needs to closely match reality." While Gazebo prioritizes accurate physics simulation, Isaac Sim focuses on photorealism, which is crucial for perception tasks, as modern autonomous robots rely heavily on computer vision.

Considering the advantages and limitations of each simulator, Isaac Sim was selected as the simulation environment for this project. Its advanced rendering capabilities and realistic physics simulation make it well-suited for accurate modeling. Additionally, being a modern technology that evolves rapidly with new features and improvements, Isaac Sim represents a promising choice for future developments in humanoid robotics simulation.

Additionally, the ROS 2 was adopted as the robotic middleware due to its widespread use in robotics research and industry, extensive community support, and comprehensive ecosystem of tools and libraries. ROS 2 enables communication and coordination between the different software components, providing a common interface for integrating perception, control, planning, and simulation. Furthermore, its native compatibility with NVIDIA Isaac Sim enables seamless interaction between the simulation environment and the robotic system, making it well suited for the framework developed in this work [70].

Several simulation environments and humanoid robot platforms were analyzed and compared to identify the most suitable solution for the proposed construction scenario. The selected simulation environment and humanoid platform are presented in Sections 2.4.1 and 2.4.2, respectively.

2.4.2 Humanoid Robot Model

Building on the simulation and control capabilities provided by ROS and Isaac Sim, a humanoid robot was selected to serve as the platform for simulating the task of vinyl flooring installation.

Among the various humanoid robots available, since Isaac Sim has available assets to be used directly in the simulator, the humanoid robot model selected for this project was chosen among those available in Isaac Sim's asset library. In the decision of the most suitable humanoid robot, factors such as mobility capabilities, manipulation dexterity, physical aspects like weight and height, and available sensors and workload capacity were taken into account. After evaluating the available options, the Unitree H1 humanoid robot was selected.

2.4.2.1 Unitree H1

The H1 is a humanoid robot developed by Unitree, a Chinese robotics company, designed as a full-size bipedal platform for research in dynamic locomotion and whole-body control. Unlike smaller humanoid or legged robots, the H1 is intended to operate at human scale, enabling the evaluation of balance, walking, running, and coordinated limb motion under realistic conditions. Its mechanical and control design focuses on strong movement ability, which is an essential characteristic for humanoid robots intended to operate in unstructured or human-centered environments.



Figure 2.2: H1 humanoid robot from Unitree. [71]

From a physical perspective, the H1 features a human-sized body, with a height and mass comparable to those of an adult person. Standing at approximately 180 cm in height and weighing around 47 kg, its dimensions make it suitable for real-world dynamics and human-robot interaction testing. These characteristics were taken into account, as the robot is intended to operate in environments designed for humans and to interact with them directly. Moreover, its relatively lightweight design contributes to increased safety during physical interaction.

It features a robust mechanical design with multiple Degrees of Freedom (DoF). Each leg has 5 DoF (hip × 3, knee × 1, ankle × 1) and each arm has 4 DoF (expandable) enabling complex locomotion and manipulation tasks. The robot is powered by high-torque joint motors, achieving maximum joint torques up to 360 N·m, and its advanced powertrain allows it to walk and run with a moving speed of over 3.3 m/s, with potential mobility exceeding 5 m/s, reflecting a high level of agility and dynamic performance.

In terms of perception, the H1 is equipped with a 3D LiDAR (Livox Mid-360) and a depth camera (Intel RealSense D435i), which together provide real-time spatial data for environment mapping and navigation. Its onboard computing capabilities typically include an Intel Core i5 or i7 processor, with optional configurations such as NVIDIA Jetson Orin NX for enhanced control and perception processing. The robot's battery capacity is 864 Wh, which is quickly replaceable to support extended operation. These characteristics, combined with its modular hardware and sensor suite, make the H1 a versatile platform for simulation, perception, control, and interaction research in humanoid robotics [71], [72].

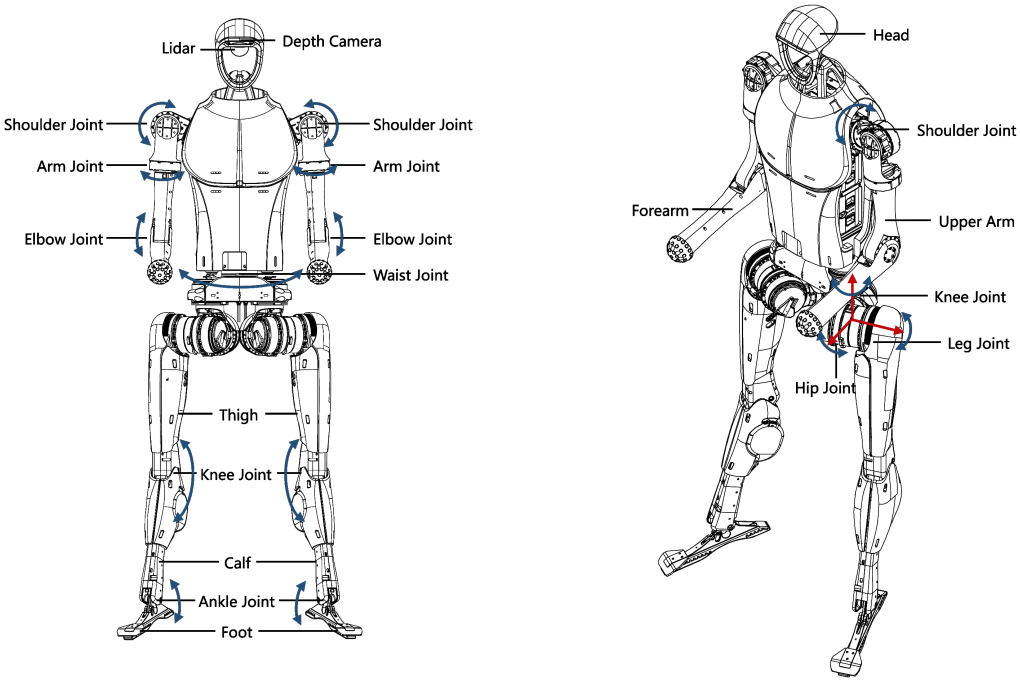


Figure 2.3: H1 main body structure. [73]

Overall, the Unitree H1 provides a suitable humanoid platform for simulation-based studies and control development, combining human-scale morphology, dynamic locomotion capabilities, and a sensor configuration aligned with modern humanoid robotics research [71], [72].

Chapter 3

Methodology

3.1 Task Coordination using the Open Scalable Production System

The process described in Section 2.3.3 represents a manual operation, illustrating how the task is performed by human workers, which comprises several interconnected stages, each involving distinct operations and technical challenges. While certain steps of this process are particularly challenging to automate, the aim of this project is not to replicate the human approach in its entirety. Instead of replicating the Standard Operating Procedure (SOP) for this task, the objective is to leverage the specific capabilities of the robot to perform operations more efficiently and safely. Based on the analysis of the overall process and its constituent activities, a subset of representative tasks was selected for implementation by the humanoid robot. These operations include autonomous locomotion within the construction environment, perception of the materials around the environment and manipulation of vinyl flooring planks. These activities were selected because they represent fundamental capabilities required for humanoid robots operating in construction environments. Furthermore, material handling is a recurring operation across a wide range of construction activities, making the proposed solution applicable beyond the specific case of vinyl flooring installation.

To support the development of modular and reusable robot capabilities, this work adopts the Open Scalable Production System (OSPS). OSPS is a skill-based software framework in which robot functionalities are encapsulated as independent software modules, referred to as skills. Each skill implements a single capability and exposes a standardized interface, allowing different functionalities to be combined into more complex robotic applications.

The OSPS architecture is organized around three main components: the Production Manager, the Task Manager, and the robot skills. The Production Manager assigns tasks to available robotic resources, while the Task Manager coordinates the execution of the corresponding skills according to a predefined SOP. In OSPS, each robot capability is encapsulated as an independent ROS 2 Action server, referred to as a *skill*. This abstraction provides a standardized communication interface between high-level task coordination and the underlying robot controllers while hiding the internal implementation of each capability.

Each skill receives a goal describing the requested operation, executes its corresponding control

algorithm, periodically publishes execution feedback, and returns a result upon completion. This communication mechanism allows different robot capabilities to share a common interface regardless of their internal implementation. Following this architecture, the functionalities developed were implemented as independent OSPS skills, in particular, locomotion and object detection.

This system was chosen since this modular design offers several advantages. First, each capability can be developed and tested independently from the remaining system. Second, software reuse is promoted, as the same skill can be incorporated into different robotic applications without modification. Finally, the standardized interface provided by the OSPS framework facilitates the future integration of additional robot capabilities and enables these modules to be incorporated into complete task-oriented workflows.

3.2 Software Development Environment

The proposed framework was developed using NVIDIA Isaac Sim 4.5.0 and 5.1 (for locomotion and navigation testing), NVIDIA Isaac Lab (main branch), and ROS 2 Humble, which together provide the simulation, robot control, and communication infrastructure required for the implementation of the proposed system.

NVIDIA Isaac Sim was used as the main simulation platform, providing a high-fidelity physics engine, realistic sensor simulation, and an environment for modeling and testing the humanoid robot and the construction scenario. The simulator also enables seamless integration with ROS 2, facilitating the exchange of information between the simulated robot and the external software modules.

NVIDIA Isaac Lab was employed for the development and training of reinforcement learning policies, providing a framework for creating robotic tasks, defining reward functions, and training manipulation controllers within Isaac Sim.

ROS 2 Humble was adopted as the middleware responsible for communication between the different software components. It provides the infrastructure for data exchange, coordination of robot functionalities, and integration of the navigation, perception, manipulation, and task execution modules developed throughout this work.

The interaction between these technologies forms the software foundation of the proposed robotic framework. The specific implementation of each component is described in the following sections.

3.3 Simulation Environment

3.3.1 Environment Development

The simulation environment was designed to replicate a realistic construction scenario. To create the environment, Autodesk Revit, a BIM software used by architects, engineers, and construction professionals, was used to model a residential building during its construction phase. The building

was designed to represent a house before the installation of vinyl flooring, including the structural elements required for the task, such as walls, door openings, and the finished floor surface.

The model, shown in figure 3.1, comprises two connected rooms, measuring approximately 14.8 m by 15.3 m combined. This layout was intentionally designed to replicate a realistic residential environment in which one room requires vinyl flooring installation, allowing the robot to navigate between different areas of the house Furthermore, the two-room configuration enables one room to represent the installation area, which remains free of unnecessary obstacles, while the second room serves as a storage area for construction materials and tools.

Once completed, the BIM model was exported from Revit in the GL Transmission Format (glTF) through the Leia extension, an exporter from Autodesk Revit, and imported into NVIDIA Isaac Sim, where it became the basis of the simulation environment.

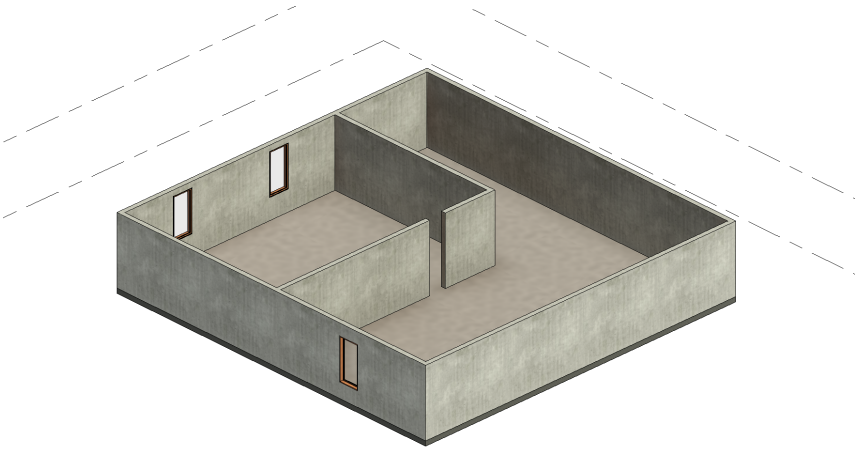


Figure 3.1: Environment model in Autodesk Revit.

3.3.2 Environment Assets

To increase the realism of the construction scenario, objects commonly found on construction sites were incorporated into the environment, including pallets, storage boxes, and other construction equipment. These assets were obtained from the NVIDIA Isaac Sim asset library as well as publicly available Universal Scene Description (USD) models.

In addition to these generic construction assets, task-specific objects were developed to support the vinyl flooring installation scenario. Since these objects were not available in the default Isaac Sim asset library, they were modeled using Blender. The developed models include vinyl flooring planks, vinyl flooring box, and a foam underlay roll.

Among these objects, the vinyl flooring plank is the primary object manipulated by the humanoid robot. The model was designed to represent a click-lock Luxury Vinyl Tile (LVT) plank with dimensions of approximately 1220 mm $\times$ 180 mm $\times$ 5 mm, as presented in figure 3.2. Click-lock planks feature a tongue-and-groove locking system that enables adjacent planks to interlock without

adhesives. The flooring boxes and foam roll contribute to the realism of the construction scenario while also acting as obstacles during navigation. The flooring boxes were modeled with dimensions of 1250 mm × 190 mm × 100 mm, corresponding to commercially available packages containing 8-10 planks, covering approximately 1.76 m^2 to 2.0 m^2 and weighing between 10 kg to 12 kg. All custom objects were created with dimensions and physical characteristics similar to their real counterparts, providing a more representative and realistic simulation environment.

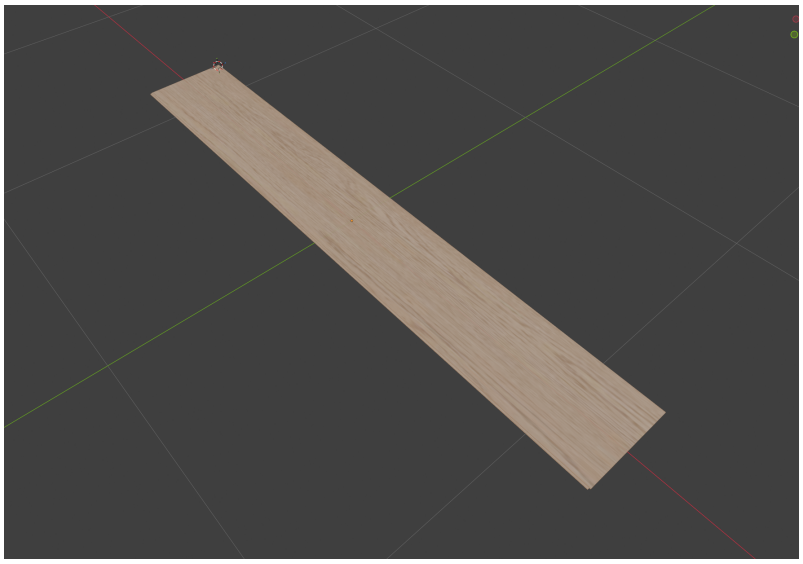


Figure 3.2: Custom vinyl plank model in Blender.

The custom models were exported from Blender in USD format and subsequently imported into NVIDIA Isaac Sim.

3.3.3 Final Simulation Scenario

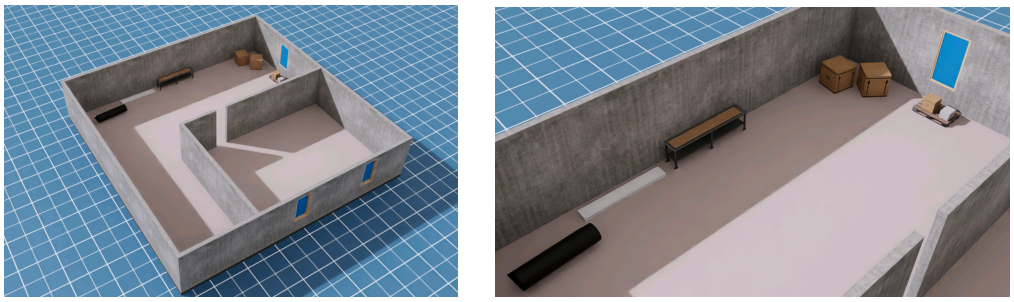
The final simulation environment, displayed in figure 3.3a was obtained by combining the building model imported from Revit with both the standard construction assets and the custom-designed objects placed as shown in figure 3.3b, resulting in a scenario that serves as the experimental environment for evaluating the proposed robotic framework. It provides the necessary conditions to assess the robot's navigation, perception, manipulation under realistic construction site conditions.

3.4 Robot Configuration

3.4.1 Robot Platform and Sensor Configuration

As introduced in Section 2.4.2, the Unitree H1 humanoid robot was selected as the platform for this work for its human-like morphology and its suitability for construction tasks. The simulation uses the USD model provided in the NVIDIA Isaac Sim asset library.

The robot was equipped with a Red, Green, Blue and Depth (RGB-D) camera, an Intel RealSense D455, and a rotating 2D LiDAR sensor, both from the NVIDIA Isaac Sim asset library. On



(a) Overview of the environment. (b) Construction objects.

Figure 3.3: Environment in NVIDIA Isaac Sim.

the physical H1 both sensors are mounted on the head, but the USD model available in Isaac Sim has no dedicated head link to attach them to. Both sensors were therefore fixed to the torso link and positioned as close as possible to their location on the real robot.

The RGB-D camera supplies the color and depth data used for object detection and scene perception. The LiDAR supports mapping, localization and navigation. Both sensors were configured with the sensor models available in Isaac Sim and connected to ROS 2 through Action Graphs, which publish the sensor data to the corresponding ROS 2 topics for use by the developed skills.

3.4.2 Gripper Selection

The default Unitree H1 model available in the NVIDIA Isaac Sim asset library can be configured with or without dexterous robotic hands on both arms. These hands allow anthropomorphic manipulation, but alternative end-effector configurations were evaluated against the requirements of the proposed material handling task.

The task requires manipulation of vinyl flooring planks, which are thin and flat and can be difficult to grasp reliably with the dexterous fingers alone, particularly when resting on a flat surface. Nevertheless, later extensions of the framework may also require the robot to handle construction tools at different stages of the installation, for instance to cut the planks or to position the foam underlay.

A hybrid gripper configuration was therefore adopted. The right arm keeps the dexterous hand, so that the robot can still manipulate tools and interact with the environment in other ways. On the left arm, the default hand was replaced by the suction gripper. The model used for the suction gripper is also available in the NVIDIA Isaac Sim asset library `short_gripper.usd`. An image of this configuration is presented in figure 3.4.

Suction was chosen because it holds flat surfaces more reliably than finger-based grasping in this application. The resulting configuration handles the vinyl planks with the left arm while keeping the dexterous hand available on the right for the task extensions described above.

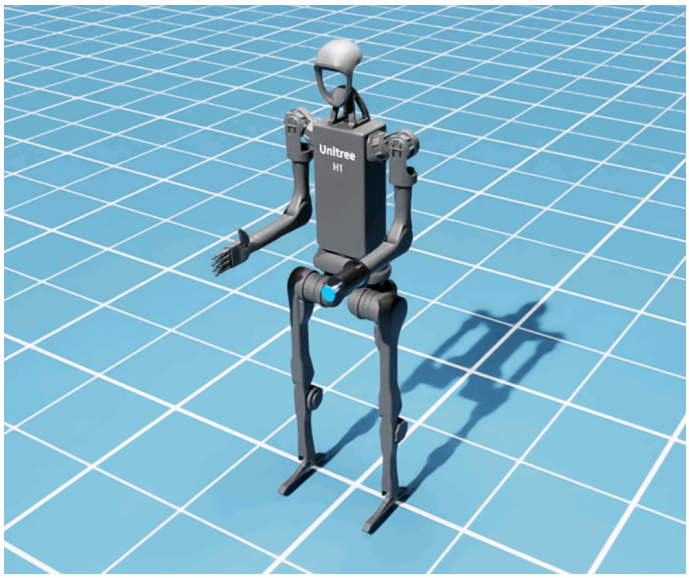


Figure 3.4: Hybrid gripper configuration in H1 model.

3.5 Object Detection Using YOLO

Autonomous manipulation requires the robot to perceive its surroundings and identify the objects involved in the task. To provide this capability, a vision-based perception module based on the YOLOv8 object detector was developed to recognize construction materials and workstation elements required during the simulated flooring installation task. Since the proposed framework operates entirely in simulation, synthetic data generation was adopted instead of collecting and manually annotating real-world images.

The perception pipeline consists of four main stages: synthetic dataset generation using NVIDIA Isaac Sim Replicator, automatic annotation of object instances, conversion of the generated annotations into the format required by YOLO, and fine-tuning of a pretrained YOLO model. The resulting detector is subsequently integrated into the ROS 2 perception framework, where it processes Red, Green and Blue (RGB) images acquired by the Intel RealSense D455 camera mounted on the humanoid robot.

3.5.1 Synthetic Dataset Generation with Isaac Sim Replicator

The development of robust object detectors typically requires large quantities of annotated training data. In construction environments, however, collecting and manually annotating such datasets is both time-consuming and expensive, while ensuring sufficient variability in viewpoints, object arrangements, and illumination conditions remains particularly challenging. To overcome these limitations, the NVIDIA Isaac Sim Replicator framework was utilized. It is a synthetic data generation framework that uses domain randomization to automatically generate annotated datasets for training computer vision models and was employed to generate a synthetic dataset directly from the simulation environment developed for this dissertation.

The simulated workspace reproduces the flooring installation environment described throughout this work and includes the construction objects required during task execution. Four semantic classes were considered during dataset generation:

- `pavement`: vinyl flooring plank;
- `box_pav`: flooring package;
- `roll_foam`: foam underlay roll;
- `table_station`: work table.

These objects were selected for the synthetic dataset because they represent the materials and workstation elements involved in the flooring installation task, and are therefore expected to be recognized and manipulated by the robot during future autonomous task execution. Each object was assigned with a semantic label using the semantic annotation functionality provided by Isaac Sim, allowing Replicator to automatically generate object annotations for every rendered image.

To increase the diversity of the generated dataset and improve the generalization capability of the detector, domain randomization techniques were applied during image generation. For each rendered frame, the following scene properties were randomly modified:

- object positions within predefined regions;
- object orientations;
- camera pose;
- illumination intensity and direction.

The virtual RGB camera was randomly positioned around the workspace to emulate the viewpoints that may be encountered by the humanoid robot during operation. This process increases the visual diversity of the dataset while reducing the dependency of the detector on a single scene configuration.

A total of 499 RGB images with a resolution of 640×640 pixels were generated. For each image, Replicator automatically exported the corresponding two-dimensional tight bounding boxes in NumPy [74] format together with the semantic identifier associated with each visible object.

3.5.2 Automatic Annotation and Dataset Conversion

Although Isaac Sim Replicator automatically generates object annotations, these are not directly compatible with the format required by YOLO. Therefore, a custom Python script was developed to convert the exported annotations into both the annotation format and directory structure expected by the Ultralytics framework.

Each detected object is associated with a semantic identifier (`semanticId`), which was mapped to the corresponding object class used during training.

Bounding boxes smaller than five pixels were discarded to eliminate annotations containing negligible visual information, and duplicate annotations corresponding to the same object were removed before generating the final label files.

The dataset was subsequently divided into training, validation, and testing subsets using an 80:10:10 split, resulting in 399 training images, 49 validation images, and 51 testing images. Frames without visible objects were excluded, as they do not provide useful localization information for detector training.

Finally, the conversion script automatically generated the `data.yaml` configuration file required by the Ultralytics framework, containing the dataset paths and class definitions. The correctness of this conversion was verified by inspecting a batch of training images reconstructed by the Ultralytics framework directly from the converted label files, shown in Figure 3.5. The bounding boxes overlaid on each image are correctly aligned with the corresponding objects, confirming that the coordinate conversion and the class mapping were performed without error prior to training.

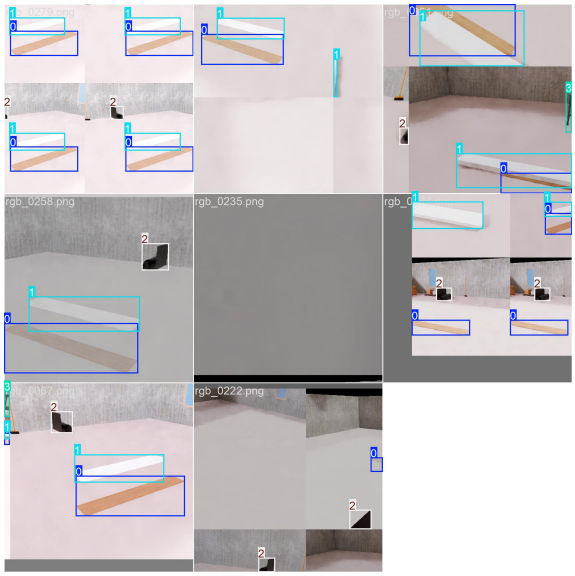


Figure 3.5: Batch of training images reconstructed by the Ultralytics framework from the converted YOLO format annotations.

3.5.3 YOLO Training Configuration

The object detector adopted in this work is based on the YOLO architecture developed by Ultralytics. Among the available model variants, YOLOv8s was selected because it provides a suitable compromise between detection accuracy and computational efficiency, making it well suited for real-time robotic perception applications. As a one-stage object detector, YOLOv8s simultaneously predicts the location and class of objects within an input image, producing a set of bounding boxes, confidence scores, and class labels in a single forward pass through the network.

Rather than training the network from scratch, transfer learning was employed by fine-tuning the pretrained `yolov8s.pt` model using the synthetic dataset generated. The pretrained model

was originally trained on the Common Objects in Context (COCO) dataset, which contains a large collection of annotated images covering everyday object categories. Through this pretraining process, the network learns generic visual features, such as edges, textures, and object shapes, that can be transferred to new detection tasks. Fine-tuning then adapts these learned representations to the four construction-related object classes considered in this work, enabling accurate object detection while significantly reducing both the amount of training data and the training time required.

Training was performed for 100 epochs using an input image resolution of 640×640 pixels and a batch size of eight images. Early stopping with a patience of 20 epochs was employed to prevent unnecessary training once convergence had been achieved. To further improve the robustness of the detector, the data augmentation techniques provided by the Ultralytics framework were enabled, including Mosaic augmentation, random rotations of up to $\pm 15^{\circ}$, horizontal flipping, and vertical flipping. These transformations increase the visual diversity of the training samples, reducing overfitting and improving the detector's ability to generalize to unseen viewpoints and object configurations.

The resulting model was exported as a custom set of weights, which are subsequently employed by the perception module during real-time object detection in the simulated robotic environment.

3.5.4 Object Detection Skill

The trained YOLO detector was integrated into the robotic architecture as a dedicated ROS 2 skill, following the action based structure adopted throughout this work for every capability exposed to the task manager. The skill is implemented as a ROS 2 action server built on the standard skill template used across the architecture, in which the goal, feedback and result messages are defined by a custom action interface named `FindobjectSkill`. The goal accepts an optional `object` field identifying the class to be searched for, while the result reports the completion percentage, a textual status and an outcome flag among `succeeded`, `not_found` and `aborted`. The fields defined by this action interface are summarised in Table 3.1.

Table 3.1: Fields of the `FindobjectSkill` action.

Section	Field	Meaning
Goal	`object`	Name of the object class to detect (optional)
Result	`percentage`	Completion percentage attained
	`skill_status`	Human-readable termination message
	`outcome`	`succeeded`, `not_found` or `aborted`
Feedback	`percentage`	Detection progress during execution
	`skill_status`	Currently detected object classes

Since perception is required to operate continuously throughout the execution of the manipulation task, rather than only during the brief interval in which a single goal is active, the skill decouples the image processing pipeline from the lifecycle of individual action goals. The RGB

image stream published by the camera mounted on the humanoid robot is consumed through a subscription to the `/rgb` topic, illustrated in Figure 3.6, which shows the corresponding node and topic connectivity as reported by the ROS 2 computation graph.

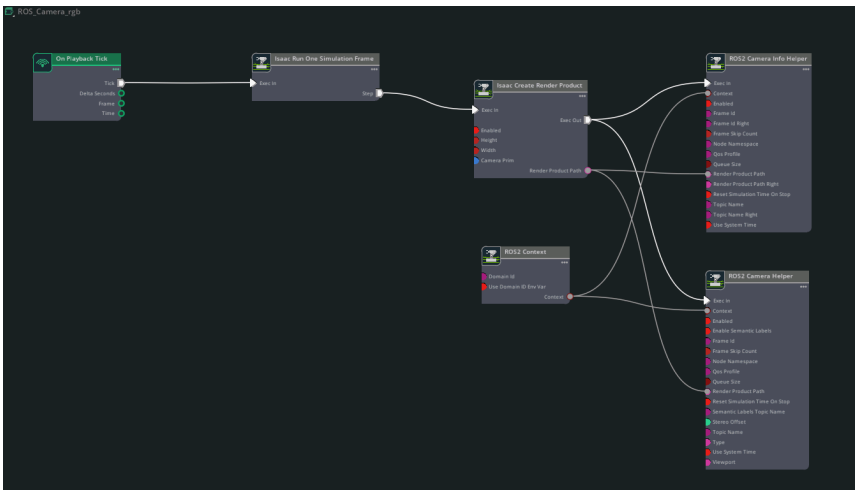


Figure 3.6: ROS 2 computation graph for `/rgb` camera topic publishment.

On receiving the first goal, the server loads the trained network and creates the subscription to the camera topic together with two publishers, one for the annotated image and one for a `Detection2DArray` message containing the class, confidence and bounding box of every detected object. From that point onward, every incoming frame is processed by a dedicated image callback that runs independently of any active goal, so that detections and their visualization remain available for the entire duration of the simulation, including through Rviz, and are not tied to the acceptance of a specific request.

The role of an individual action goal is therefore reduced to querying the state maintained by this continuous perception process. Once a goal is received, the server monitors the most recent detections and periodically publishes feedback messages describing the classes currently visible. A goal is considered fulfilled, and the action succeeds, once the requested object, or any object when the field is left empty, has been observed with a confidence above a configurable threshold in three consecutive frames, which prevents an isolated false positive from being reported as a s uccessful detection. If no matching detection occurs within a configurable time limit of 120 seconds, the action terminates with the outcome `not_found` rather than being left indefinitely pending, although this limit can also be disabled to allow the skill to keep monitoring the scene for the entire duration of the simulation. Preemption is supported through the cancellation mechanism provided by the action server template, allowing an active goal to be interrupted by the task manager at any point during its execution.

The camera image, published in the `rgb8` encoding, together with the model weights and the class list, are the only elements specific to this deployment. The confidence threshold, the number of consecutive detections required and the goal timeout are exposed as ROS 2 parameters, which allows the behavior of the skill to be adjusted without modifying its implementation, for instance

when tuning the trade off between responsiveness and robustness to spurious detections discussed in Section 4.1.3.

3.6 Reinforcement Learning for Humanoid Manipulation

The manipulation task developed in this dissertation aims to enable the Unitree H1 humanoid robot to autonomously grasp a vinyl flooring panel using a suction gripper attached to its left arm and lift it while maintaining a stable horizontal orientation. This manipulation stage represents a key component of the overall construction workflow, as it allows the robot to securely handle flooring materials before transportation and placement.

Unlike conventional trajectory-based approaches, Reinforcement Learning (RL) was selected due to the complexity of the interaction between the suction gripper and the object. Successful attachment depends simultaneously on the relative position, orientation, and contact distance between the suction cup and the panel surface. These relationships are highly nonlinear and difficult to model analytically, making RL a suitable solution for learning the manipulation behavior directly through interaction with the environment.

The policy was trained using NVIDIA Isaac Lab, which provides a GPU-accelerated reinforcement learning framework built on top of Isaac Sim. The ability to execute thousands of simulation environments in parallel significantly reduces training time while enabling robust policy optimization.

The scope of the manipulation policy was intentionally restricted to grasping and lifting the flooring panel.

3.6.1 Simulation Environment

The manipulation environment was implemented using the `DirectRLEnv` interface provided by Isaac Lab. This interface provides the framework required to develop reinforcement learning environments by managing the interaction between the simulator and the learning algorithm, while allowing the task-specific observations, rewards, actions, and termination conditions to be defined by the user. This approach offers greater flexibility than the manager-based framework and was therefore adopted because the suction attachment mechanism could not be conveniently implemented using the default manager architecture.

The simulated scene consists of three main components:

- Unitree H1 humanoid robot equipped with a suction gripper mounted on the left arm;
- A workbench supporting the flooring panel;
- A rigid vinyl flooring panel representing the manipulated object.

Instead of manually defining the workbench dimensions, its height and bounding box are obtained directly from the simulation model during runtime. Consequently, the initial pose of the

flooring panel is computed automatically from the detected workbench geometry, preventing inconsistencies when modifying Computer-Aided Design (CAD) models or scaling the environment.

The simulation operates at a physics frequency of 200 Hz, while the control policy executes at 50 Hz through a decimation factor of four. Each training episode lasts 10 seconds, corresponding to 500 control steps, with 2048 environments simulated simultaneously to maximize GPU utilization.

During training, the humanoid base remains fixed to the world frame. This design choice isolates the manipulation problem from locomotion, preventing balance failures from dominating the learning process.

3.6.2 Observation Space

The reinforcement learning policy receives a 96-dimensional observation vector containing robot proprioception together with task-specific information. The observation includes:

- Joint positions and joint velocities;
- Suction gripper position;
- Right hand position;
- Flooring panel position and orientation;
- Relative vectors between the robot end-effectors and the object;
- Projected gravity vector;
- Base linear and angular velocities;
- Binary grasp state indicators.

All geometric quantities are represented relative to the robot base frame rather than the world frame. This representation improves generalization by allowing the learned policy to operate correctly regardless of where the navigation system positions the robot within the environment.

3.6.3 Action Space

The policy outputs a continuous action vector composed of 33 joint commands together with one suction activation command. Rather than directly predicting absolute joint positions, the policy generates joint offsets relative to a predefined reference posture,

$$q_{target} = q_{default} + \alpha \Delta q, \tag{3.1}$$

where α represents the action scaling factor, with a value of $\alpha = 2.5$.

The commanded joint targets are further filtered using an exponential moving average to reduce oscillatory behavior and improve trajectory smoothness.

3.6.4 Suction Grasp Model

The suction mechanism is represented as a kinematic attachment rather than a full vacuum simulation. An attachment is created only when several conditions are simultaneously satisfied:

- suction command is active;
- suction cup orientation is approximately perpendicular to the panel surface;
- the distance between the suction cup and the panel lies within an admissible tolerance;
- the suction cup contacts the plank surface rather than empty space.

Once these conditions are met, the relative pose between the suction cup and the flooring panel is fixed throughout the manipulation phase. This simplified model provides stable grasp behavior while avoiding the computational cost of simulating detailed vacuum dynamics.

3.6.5 Reward Function

The reward function combines multiple objectives that progressively guide the robot towards successful manipulation. The main reward components encourage:

- approaching the flooring panel;
- aligning the suction cup with the panel surface;
- establishing suction contact;
- lifting the panel to the desired height;
- maintaining a stable horizontal orientation after grasping.

Additional penalty terms discourage excessive joint motion, collisions with the workbench, loss of suction attachment and unstable object motion. The final reward corresponds to the weighted sum of all individual components.

3.6.6 Adaptive Curriculum Learning

To facilitate learning, the suction attachment conditions are progressively tightened throughout training. Initially, relatively permissive alignment and distance thresholds allow the robot to discover successful grasping behaviors. As the grasp success rate improves, these thresholds gradually converge towards realistic values, producing a curriculum that automatically adapts to the policy performance.

Unlike a fixed curriculum schedule, the progression is directly linked to the observed grasp success rate. If performance deteriorates, the curriculum relaxes the task difficulty, preventing the policy from becoming trapped in unrecoverable learning states.

3.6.7 Policy Training

The manipulation policy was trained using the Proximal Policy Optimization (PPO) algorithm implemented in the RSL-RL framework. Training was performed using 2048 parallel environments over 5000 learning iterations.

The most relevant training parameters are summarised in Table 3.2.

Table 3.2: Main PPO training parameters.

Parameter	Value
Algorithm	PPO
Parallel environments	2048
Episode length	500 control steps
Learning rate	1×10^{-4}
Entropy coefficient	0.001
Initial action standard deviation	0.40
Control frequency	50 Hz
Physics frequency	200 Hz
Training iterations	5000

3.7 Locomotion and Navigation

This section presents the methodology adopted to enable autonomous locomotion of the Unitree H1 humanoid robot within the simulated environment. The objective is not to develop a new locomotion controller, but to integrate an existing learned locomotion policy with higher level navigation and task execution layers, so that the robot can be commanded in terms of destinations rather than gaits.

The locomotion controller accepts desired body velocities as `geometry_msgs/msg/Twist` messages on the `/cmd_vel` topic, and every component above that interface reasons in terms of body velocity rather than joint kinematics or gait. This abstraction allows navigation algorithms developed for wheeled mobile robots to command a humanoid platform without modification, since their output is already a velocity command. It also allows the navigation implementation to be replaced without affecting the task execution layer, provided the replacement respects the same interface. Two navigation strategies were developed on this basis and are described in Section 3.7.3.

The framework is consequently organized into three layers. At the lowest level, NVIDIA Isaac Sim provides the physics simulation of the humanoid and its environment. Above it, a reinforcement learning locomotion policy converts desired body velocities into joint position commands, and a navigation layer, for which two alternative implementations were developed, produces those velocity commands from a requested destination. At the top of the stack, a skill exposes navigation as a single ROS 2 action, so that a movement can be requested and its outcome observed without knowledge of how it is executed.

All experiments platform specifications are summarised in Table 3.3.

Table 3.3: Computational platform used for the experiments.

Component	**Specification**
Processor	Intel Core i5-13600KF (13th generation), 20 threads
System memory	16 GB (15 GB allocated to WSL)
Graphics	NVIDIA GeForce RTX 4070
Graphics driver	32.0.15.9186
Host system	Windows 11 (build 26200)
Simulator	NVIDIA Isaac Sim 5.1.0 (Windows)
Guest system	Ubuntu 22.04.5 Long Term Support (LTS) under WSL2 (kernel 6.18.33.2)
Middleware	ROS 2 Humble
Real-time factor	0.24–0.50, depending on the active navigation stack

3.7.1 Reinforcement learning locomotion controller

The locomotion controller used in this work is the flat terrain reinforcement learning policy provided by NVIDIA within the Isaac Lab framework. It corresponds to the `Isaac-Velocity-Flat-H1-v0` environment and was trained to control the Unitree H1 humanoid on flat surfaces. The policy supports forward walking and rotation in place, and does not support backward or lateral locomotion, a restriction that later constrains the recovery maneuvers available to the navigation layer.

A pre-trained policy was selected rather than developing a new controller because the contribution of this work lies in the integration of locomotion, navigation and task execution. A policy that is documented and distributed with the simulator provides a reproducible foundation for that integration.

The policy requires the robot model to be rigged, that is, to expose an articulation whose joint ordering and drive configuration match those used during training. The USD asset adopted in this work is already rigged accordingly, which removes a source of silent failure and avoids the need to reproduce the rigging procedure.

The policy runs an independent ROS 2 node that subscribes to the robot state topics, constructs the observation vector, evaluates the neural network and publishes the resulting joint commands. The observation vector combines inertial measurements taken at the pelvis link, the relative joint positions and velocities reported by the simulated articulation, the commanded body velocity and the action applied in the previous iteration. The presence of this last component means that the policy is not memoryless with respect to its own output, so a discontinuous command sequence perturbs the observation in a way that a smooth sequence does not. This property motivates the velocity profiling described in Section 3.7.6.2.

The node must be active before the simulation starts. Without joint targets the articulation has no mechanism to maintain balance and collapses under gravity. When the node is active and no motion command is received, it holds the robot in a standing posture.

3.7.2 Isaac Sim and ROS 2 integration

Communication between the simulated humanoid and the autonomy stack is implemented through the ROS 2 interfaces provided by Isaac Sim, which exposes data through OmniGraph Action Graphs evaluated during the simulation loop. Data related to control and proprioception is published once per physics step rather than once per rendered frame, so that the control and sensing rates remain independent of rendering performance. This distinction matters because physics and rendering advance at different rates, and a control loop driven by rendering would inherit the variable frame rate of the graphics pipeline.

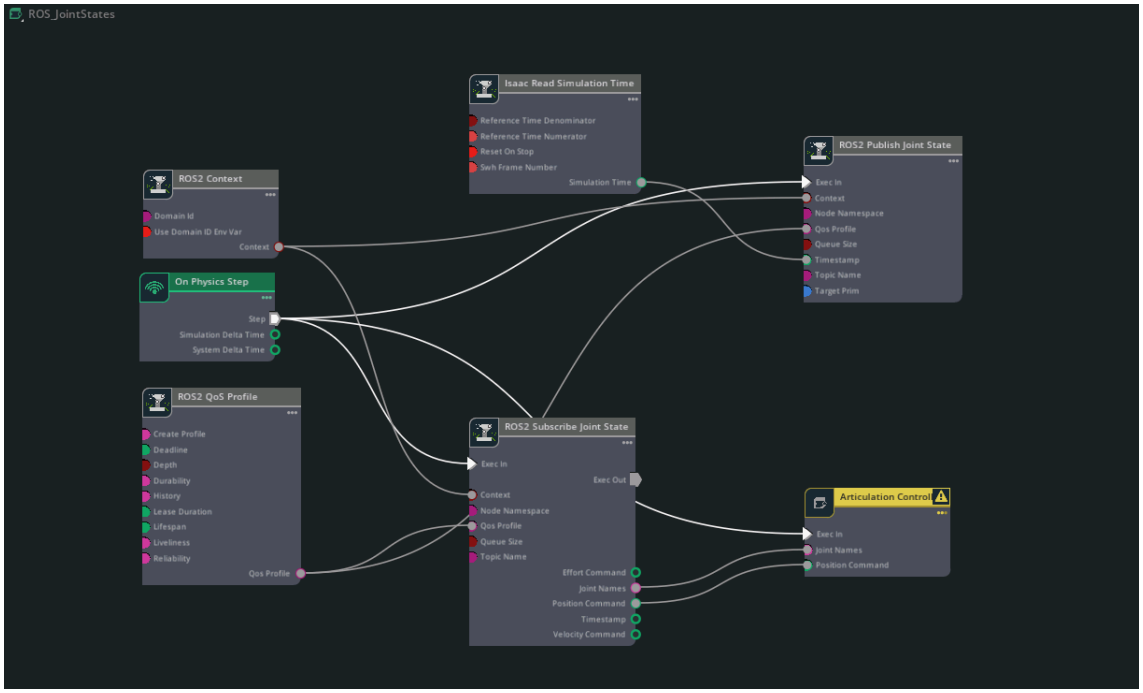


Figure 3.7: Action Graph responsible for the joint state interface.

3.7.2.1 Robot state and control interfaces

Communication between the simulator and the locomotion stack is established through a set of ROS 2 topics, listed in Table 3.4. Three carry the robot state published by Isaac Sim, and two carry the commands that act upon it, one at the level of individual joints and the other at the level of the body velocity.

Table 3.4: Main ROS 2 interfaces between Isaac Sim and the locomotion stack.

Topic	Message type	Source	Purpose
`/joint_states`	`sensor_msgs/JointState`	Isaac Sim	Robot joint feedback
`/imu`	`sensor_msgs/Imu`	Isaac Sim	Body acceleration and orientation
`/joint_command`	`sensor_msgs/JointState`	Policy node	Desired joint positions
`/cmd_vel`	`geometry_msgs/Twist`	Navigation layer	Desired body velocity
`/odom`	`nav_msgs/Odometry`	Isaac Sim	Robot pose feedback

The joint state interface closes the control loop between the simulator and the policy, as illustrated in Figure 3.7. The robot configuration is published by Isaac Sim, processed by the locomotion controller and returned as desired joint positions through the command interface. The inertial measurements are associated with the pelvis frame because that is the reference frame in which the observation space was defined during training.

3.7.2.2 Odometry and coordinate transformations

Navigation requires the robot to be located with respect to the environment, which in ROS 2 is expressed through a tree of coordinate frames. The convention adopted by the navigation stack defines three frames of interest and organizes them into the chain

$$\texttt{map} \longrightarrow \texttt{odom} \longrightarrow \texttt{base} \tag{3.2}$$

where the arrow denotes the parent to child relationship.

The frame `base_link` is rigidly attached to the body of the robot and moves with it, and in this work it is located at the pelvis. Quantities expressed in this frame, such as the range measurements of a sensor, are relative to the robot rather than to the environment. The frame `odom` is fixed in the world and its origin coincides with the pose of the robot at the instant the simulation begins, so that the transformation `odom` to `base_link` describes the displacement accumulated since that instant. The frame `map` is likewise fixed in the world, with its origin anchored to the occupancy grid of the environment, and it therefore provides the global reference against which navigation goals are expressed.

The pose of the robot in the map frame is obtained by applying the transformation from `map` to `odom` followed by the transformation from `odom` to `base_link`, so that a displacement reported by the odometry is interpreted relative to the point at which the robot started, and that point is in turn expressed in map coordinates.

The `odom` frame is interposed rather than relating `map` to `base_link` directly for two reasons. The first is structural, since a transformation tree admits only one parent per frame, and the odometry source already publishes the transformation to `base_link`. A localization algorithm therefore publishes the correction, expressed from `map` to `odom`, rather than a second transformation to the same frame. The second reason is functional. Odometry is continuous but drifts over time, whereas localization is accurate but changes discontinuously whenever an estimate is corrected. Keeping the two in distinct transformations allows the local controller to operate in the smooth `odom` frame while the global planner operates in the accurate `map` frame, so that a correction of the pose estimate does not introduce a discontinuity in the velocity command.

In the present system the transformation from `odom` to `base_link` is obtained directly from the simulated rigid body state through the graph shown in Figure 3.8. Unlike a physical robot, where odometry is estimated by integrating proprioceptive measurements, Isaac Sim reports the pose of the robot exactly, so this transformation does not accumulate drift. This property is the basis of the navigation strategy described in Section 3.7.6.

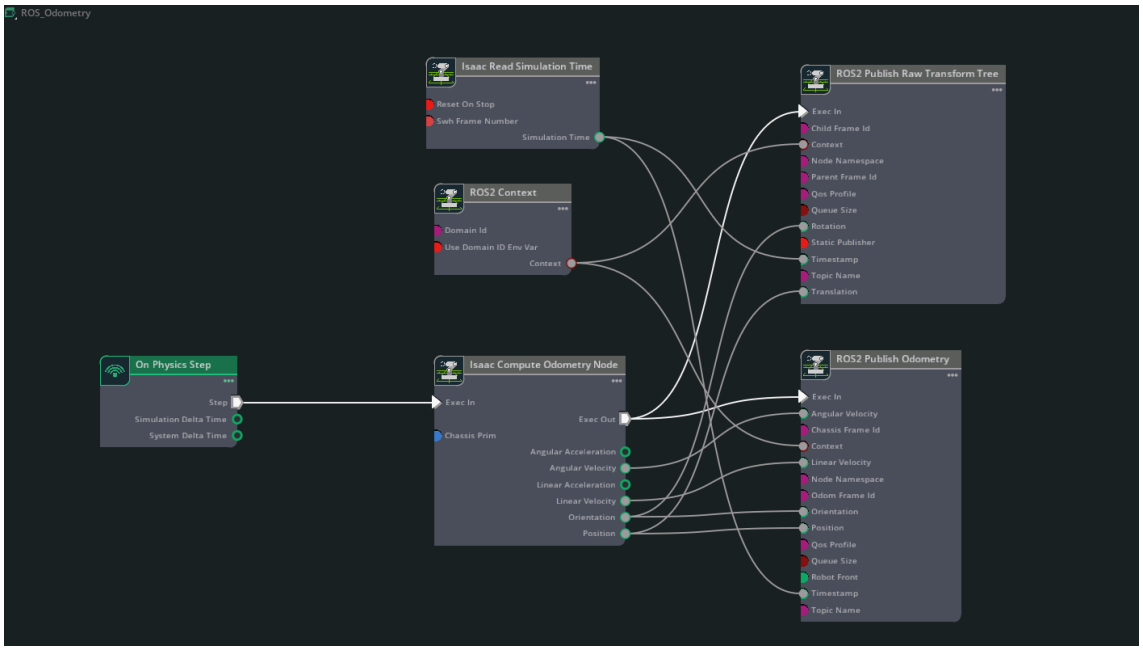


Figure 3.8: Action Graph responsible for odometry. The pose of the robot is computed from the simulated rigid body state and published both as a `nav_msgs/Odometry` message and as the transformation from `odom` to `base_link`.

The transformation from `map` to `odom` is published as a static transform, since the platform carries no exteroceptive sensor from which localization could be estimated. This is admissible precisely because the odometry does not drift, which means that the correction the transformation would otherwise provide is constant throughout a run. Its value is the pose at which the robot is spawned, expressed in map coordinates.

Obtaining that value requires reconciling two independent conventions. The occupancy grid is generated centered on the geometry of the environment, so the origin of the `map` frame coincides with the centroid of that geometry, whereas the origin of the Isaac Sim world frame is arbitrary with respect to it. The offset between the two was determined by comparing the axis aligned bounding box of the environment geometry in the simulator with the bounding box of the occupied cells in the occupancy grid, and it enters the static transform as a constant correction.

3.7.3 Navigation strategies

Two navigation strategies were developed behind a common task level interface. The first delegates planning and path following to the ROS 2 Navigation Stack, producing autonomous motion towards a pose specified in the map frame. The second executes movements defined relative to the current pose of the robot, rotating by a given angle and then advancing a given distance, without reference to a map.

The development of both approaches was motivated by the different requirements expected during humanoid robot operation. A fully autonomous system requires the ability to plan motions according to a representation of the environment and to adapt the trajectory when the initial

conditions change, and Nav2 provides a general framework capable of planning paths, following trajectories and integrating perception based obstacle avoidance once an exteroceptive sensor becomes available.

Humanoid robots nevertheless introduce constraints that conventional mobile robots do not present. The accuracy of navigation depends on the interaction between localization, locomotion dynamics and the ability of the controller to arrest the motion precisely, and each of these contributes an error that is difficult to separate from the others. The second strategy was therefore developed for scenarios in which the desired trajectory is known in advance, where movements are defined relative to the current robot pose and execution becomes deterministic and repeatable.

The two approaches are complementary rather than alternative. Nav2 offers a flexible solution for autonomous operation in unknown or partially known environments, while the relative motion strategy offers a controlled solution for predefined routes in which repeatability matters more than autonomous decision making.

3.7.4 The WalkSkill action interface

Navigation is exposed to the task manager through a single ROS 2 action interface, `WalkSkill`, which is shared by both navigation strategies.Its fields are presented in Table 3.5. The goal, feedback and result message definitions are identical across implementations, so the task manager can invoke either strategy without any change to the calling code. The semantics of the goal fields, however, are strategy-specific: in the Nav2-based implementation they specify an absolute target pose in the `map` frame, whereas in the relative-motion implementation `goal_yaw` specifies an in-place rotation and `goal_x` a forward distance, with `goal_y` unused.

Table 3.5: Fields of the `WalkSkill` action.

Section	**Field**	**Nav2 strategy**	**Relative strategy**
Goal	`goal_x`	Target position, x (map frame)	Distance to walk forward
	`goal_y`	Target position, y (map frame)	Final rotation, after walking
	`goal_yaw`	Orientation to adopt on arrival	Initial rotation, before walking
Section	**Field**	**Meaning (both strategies)**	
Result	`percentage`	Completion percentage attained	
	`skill_status`	Human readable termination message	
	`outcome`	`succeeded`, `aborted` or `canceled`	
Feedback	`percentage`	Progress during execution	
	`skill_status`	Current phase and remaining distance	

The `outcome` field is restricted to three values, which allows the task manager to distinguish a completed movement from one that failed and from one that was interrupted by a higher level decision. The interpretation of the three goal fields differs between the two strategies and is described in the corresponding sections.

The skill server is implemented as a ROS 2 action server executing in a multi threaded executor with a reentrant callback group. This combination is necessary because the execution callback

blocks while the movement is carried out, and a mutually exclusive callback group would prevent cancellation requests and feedback callbacks from being serviced during that period.

3.7.5 Autonomous navigation using Nav2

3.7.5.1 Framework and integration

Nav2 is the standard navigation framework of the ROS 2 ecosystem and provides a modular architecture composed of global planning, local trajectory control, costmap management and recovery behaviors. The global planner computes a feasible path between the current robot pose and a desired goal location, while the local controller converts that path into velocity commands that are continuously updated during execution. Both components operate over costmaps, occupancy representations in which each cell carries a traversal cost derived from obstacles and their surrounding clearance.

Nav2 was selected because of its modular architecture and its compatibility with the existing locomotion framework. The navigation layer does not control the humanoid joints directly but instead publishes velocity commands through the `/cmd_vel` topic, which the reinforcement learning locomotion controller then converts into the joint positions required to achieve the requested motion. This separation is what allows navigation algorithms originally developed for wheeled mobile robots to be applied to a humanoid platform without modifying the underlying locomotion controller.

3.7.5.2 Environment representation and map generation

The simulated environment is represented as a static occupancy grid. Rather than performing a mapping procedure with the robot, the grid was generated directly from the scene geometry using the occupancy map generator provided by Isaac Sim, which traces the collision geometry of the stage at a specified height and produces the corresponding grid together with its metadata file.

The resulting map, shown in Figure 3.9, has a resolution of 0.05 m per cell and covers approximately 15.2 m by 14.5 m, with its origin placed at $(-7.575, -7.225)$ in the map frame. Cells are classified as occupied above a threshold of 0.65 and as free below 0.196, with intermediate values treated as unknown.

Generating the map from the simulation scene rather than from an external representation of the building has a consequence for the accuracy of navigation. The grid and the simulated world share the same geometry by construction, so the occupancy of each cell corresponds exactly to the collision geometry the robot encounters, and no error is introduced by the conversion. What remains is the alignment between the origin of the grid and the origin of the simulation world, which is expressed through the `map` to `odom` transformation described in Section 3.7.2.2 and is the principal source of positional uncertainty in this strategy.

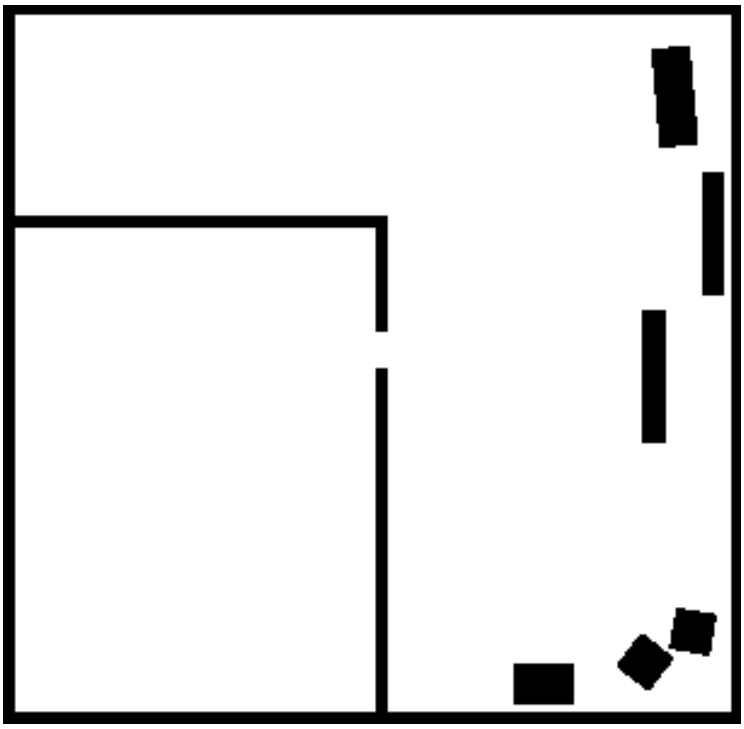


Figure 3.9: Occupancy grid used for navigation, obtained from Isaac Sim.

3.7.5.3 Configuration

Several Nav2 parameters were adapted to account for the characteristics of humanoid locomotion. Global planning uses NavFn, a grid based planner that computes a navigation function over the costmap and extracts a path by gradient descent. Path following uses the Regulated Pure Pursuit (RPP) controller, which selects a lookahead point on the path and derives a velocity command from the curvature of the arc connecting the robot to that point.

The default robot footprint was modified to account for the substantial geometric differences between a humanoid robot and a conventional wheeled robot. The Unitree H1 measures 0.57 m across the shoulders and 0.22 m in depth, resulting in an elongated horizontal cross-section. Nav2 supports two types of footprint representations: a circular footprint defined by a radius and a polygonal footprint with arbitrary geometry. It was decided to use a rectangular footprint measuring 0.40 m by 0.60 m. This configuration reduces the inscribed radius compared to a circular footprint, thereby increasing the available clearance in narrow corridors. The selected dimensions do not exactly reproduce the robot's nominal geometry. The footprint is deeper than the robot in the direction of travel, providing additional clearance for the lateral motion of the arms during walking and the trunk oscillations observed in the simulation.

The completion tolerances were also adjusted to the behavior of the locomotion policy. Since stopping is not instantaneous and the robot continues to move for a short distance after receiving a zero velocity command, a tolerance comparable in magnitude to that displacement cannot be satisfied reliably, and an unsatisfiable tolerance causes the controller to circle the goal indefinitely.

The progress checker allowance was extended for a related reason, since it declares failure when the robot does not displace itself within a given time window, a condition that occurs during every rotation in place.

Finally, all velocity limits are bounded by the operating envelope of the locomotion policy, which becomes unstable for linear or angular commands above 0.75 in the respective units.

Table 3.6: Main Nav2 parameters adapted for humanoid locomotion.

Parameter	**Value**	**Purpose**
Robot footprint	$0.40\,\mathrm{m} \times 0.60\,\mathrm{m}$	Approximate the humanoid geometry and reduce the inscribed radius
Inflation radius	$0.12\,\mathrm{m}$	Clearance around obstacles.
Local costmap layers	obstacle, inflation	Static layer removed; cells inherited from the map are never cleared
Goal position tolerance	$0.05\,\mathrm{m}$	Completion criterion in position
Goal orientation tolerance	$0.20\,\mathrm{rad}$	Completion criterion in orientation; superseded by the alignment phase
Desired linear velocity	$0.50\,\mathrm{m\,s^{-1}}$	Within the stability envelope of the policy
Minimum approach velocity	$0.15\,\mathrm{m\,s^{-1}}$	Floor of the deceleration profile near the goal
Approach scaling distance	$0.80\,\mathrm{m}$	Distance from the goal at which deceleration begins
Maximum angular velocity	$0.25\,\mathrm{rad\,s^{-1}}$	Bounds the curvature the controller may request
Lookahead distance	$0.80\,\mathrm{m}$ ($0.50\,\mathrm{m}$ to $1.20\,\mathrm{m}$)	Nominal and admissible range of the pursuit point
Progress checker radius	$0.15\,\mathrm{m}$	Avoids declaring failure during rotation in place
Progress checker allowance	$45\,\mathrm{s}$	Accounts for the settling time of the policy

3.7.5.4 WalkSkill implementation

In this strategy the goal fields specify an absolute pose in the map frame, with `goal_x` and `goal_y` defining the destination and `goal_yaw` the orientation to adopt on arrival. The implementation combines several mechanisms, each addressing a distinct characteristic of the platform.

Path decomposition. Segmentation into bounded-length segments serves a purpose beyond geometric validity. The locomotion policy does not track commanded velocities exactly, and continuous walking over longer distances allows positional error to accumulate before it can be corrected by the next localization update. Executing a single long-range goal therefore risks the controller acting on a stale pose estimate. Bounding the length of each segment limits the distance travelled between corrections and keeps the accumulated error within a range the controller can tolerate.

Proximity-based waypoint acceptance. Waypoints are accepted based on positional proximity rather than full pose matching. Since the orientation of an intermediate waypoint has no task-level significance, enforcing an orientation constraint at each waypoint would introduce unnecessary

rotations and potentially induce oscillatory behaviour. The same criterion is applied to the final waypoint, allowing the navigation goal to be released once the robot reaches the vicinity of the destination. The orientation requirement is subsequently addressed during the alignment phase described below.

Releasing the navigation goal based on position alone is necessary on this platform. The Nav2 goal checker requires both position and orientation to satisfy their respective tolerances and, being stateful, switches to testing orientation alone once the positional criterion has been met. The yaw oscillation of the biped during the gait cycle is of the same order of magnitude as any orientation tolerance that would be useful for the task, so this criterion is not reliably satisfiable on the platform. Early release is therefore adopted, and the positional error it introduces is the subject of the compensation described next.

A complementary criterion detects when a waypoint has been passed rather than reached. The minimum distance attained during each segment is recorded, and the waypoint is accepted if the robot approached it closely and subsequently recedes by a predefined margin. The criterion exists because the acceptance radius admits no tolerance in the opposite direction: a waypoint approached to slightly more than that radius satisfies no condition, and the pursuit controller continues to act on a target the robot is no longer approaching.

Compensation of the stopping displacement. The distance to the waypoint is evaluated while the robot is mid-stride, with the trunk inclined forward, so the base is further advanced than it will be once the gait terminates and the standing posture is recovered. The final waypoint is therefore offset beyond the requested goal along the arrival tangent, by a constant calibrated from the displacement observed between goal release and rest. Being a single constant rather than a model of the stopping dynamics, the offset compensates the mean displacement rather than that of any individual movement.

Orientation control in closed loop. Requests that involve a change of orientation are not handled by `NavigateToPose`. That action is defined in terms of a path to be followed, and when the requested destination coincides with the current position the global planner returns a degenerate path consisting of a single point. The RPP controller derives its command from the curvature of an arc towards a lookahead point, so a degenerate path leaves that formulation without a well-conditioned solution.

Nav2 provides a dedicated primitive for this case, the `Spin` behavior, which commands angular velocity alone. Before each step of the rotation it projects the footprint forward in time and tests it against the local costmap, and this check proved not to be satisfiable on the present platform: the primitive reports a collision with the robot standing in open space, for reasons examined in Section 4.3.6.2. The primitive assumes a mobile base whose footprint remains in a fixed plane, whereas the pose of a biped oscillates in roll and pitch throughout a rotation, so the projected footprint sweeps regions the checker does not anticipate. Rotation is therefore performed by the skill itself.

The rotation is executed in closed loop against the transform tree. The angular error is evaluated at each cycle and the commanded angular velocity is modulated accordingly, at cruise speed while the error is large and decreasing proportionally within an approach band, with a floor below which the locomotion policy does not initiate a step. The sign of the command follows the current error rather than the initial direction, so that a rotation which passes the target is corrected rather than continued. A stall detector terminates the rotation when the error ceases to improve, and velocity is driven to zero in a guarded block so that the robot stops even if the loop exits through an exception.

The same routine serves both cases. When the requested destination lies within a threshold of the current position the request is treated as a pure rotation; and once the final waypoint has been accepted, the heading is compared with the requested orientation and corrected if the difference exceeds tolerance. Separating the two objectives avoids the coupled regulation of position and orientation that produces oscillation in the pursuit controller, since the navigation phase is only required to reach a position and the alignment phase only an orientation.

Closed-loop verification. Both the pure rotation and the final alignment include a settling interval, after which the heading is measured and the correction repeated if the angular error remains above tolerance. The heading estimate is obtained from the circular mean of a series of samples rather than from a single measurement, since an individual sample reflects the instantaneous attitude of the gait cycle rather than the resting attitude.

This verification is necessary because the internal termination criterion of the rotation loop and the attitude ultimately adopted do not coincide: part of the executed rotation is recovered as the robot settles, for reasons quantified in Section 4.3.5.2. The outcome reported to the task manager is therefore based on the measured final error rather than on the internal criterion, so that a movement which reaches the destination but does not satisfy the orientation tolerance is reported as such.

3.7.6 Relative motion strategy

3.7.6.1 Motivation and operating principle

The second navigation strategy was developed as a deterministic alternative to autonomous navigation. Instead of computing a path, it executes a predefined sequence of movements expressed with respect to the current robot pose.

The reasoning follows from a property established in Section 3.7.2.2. Because the odometry is derived from the simulated rigid body state rather than integrated from sensor measurements, the transformation between `odom` and `base_link` is exact. A movement specified relative to the current pose therefore inherits only the tracking error of the locomotion policy, whereas a movement specified in the map frame additionally inherits the calibration error of the transformation between `map` and `odom`. Removing the map from the control loop removes that entire class of error, at the cost of losing the ability to plan around obstacles and to correct deviations against a global reference. The principal advantage of this approach is repeatability, since each movement is executed through the same sequence of commands.

3.7.6.2 Implementation

The same `WalkSkill` interface is used in order to maintain compatibility with the task manager, but the interpretation of the input parameters changes from an absolute goal pose to a relative motion command. The three goal fields specify a sequence of movements: `goal_yaw` a rotation to perform relative to the current heading, `goal_x` a distance to walk forward afterwards, and `goal_y` a further rotation to perform on arrival. Arbitrary routes are expressed as sequences of such goals.

The final rotation exists so that a relative command can express the same movement as an absolute goal in the Nav2 strategy, which specifies both a destination and the orientation to adopt upon reaching it. Without a third phase, the relative command would leave the robot facing along the direction of travel, and the two formulations would not describe the same movement.

Execution proceeds through these phases in sequence, separated by settling intervals during which zero velocity is commanded. Each rotation phase commands an angular velocity proportional to the heading error until that error falls below tolerance, holding the linear velocity at zero. The translation phase commands a forward velocity until the displacement measured from the pose at which the phase began reaches the requested distance, with a proportional term on the heading error correcting lateral deviation. Separating the phases avoids the coupled regulation of position and orientation and, for the same reason discussed in Section 3.7.5, prevents the oscillatory behavior that arises when both objectives compete within a single control law.

Two characteristics of the locomotion policy shape the velocity profile applied during these phases. Acceleration and braking are limited at different rates, with braking substantially slower, because the displacement produced while the robot comes to rest is a function of the momentum that the policy must dissipate. The commanded velocity is also bounded below as well as above, since reducing the output towards zero on approach would enter the region in which the policy does not initiate a gait and the robot would stall short of the destination. The profile therefore decreases linearly from the maximum to a non zero floor rather than to zero. The value of this floor was established experimentally, and its influence on the accuracy of the strategy is examined in Section 4.3.5.1.

A further mechanism compensates for the displacement that occurs after the control loop terminates. Because the robot continues to move while the policy brings it to rest, the translation phase is wrapped in an outer loop that commands zero velocity, waits for the settling interval, measures the displacement again and repeats the inner loop if the remaining distance still exceeds tolerance. Each pass measures from the current pose, so the residual distance of one pass becomes the target of the next, and the mechanism compensates the stopping displacement without requiring it to be modelled explicitly.

All timing within the controller is measured on the simulation clock rather than on the wall clock. The simulation does not advance in real time, so a control loop timed with the wall clock would produce acceleration ramps considerably more aggressive in simulated time than configured, and execution timeouts would expire prematurely.

Table 3.7: Main parameters of the relative motion controller.

Parameter	**Value**	**Purpose**
Maximum linear velocity	$0.50\,\mathrm{m\,s^{-1}}$	Within the stability envelope of the policy
Minimum linear velocity	$0.30\,\mathrm{m\,s^{-1}}$	Floor below which the gait is not initiated
Maximum angular velocity	$0.30\,\mathrm{rad\,s^{-1}}$	Bounds the rotation rate to the tracking capability of the policy
Minimum angular velocity	$0.30\,\mathrm{rad\,s^{-1}}$	Floor below which rotation is not initiated
Linear acceleration limit	$0.40\,\mathrm{m\,s^{-2}}$	Smooths the command sequence seen by the policy
Linear braking limit	$0.15\,\mathrm{m\,s^{-2}}$	Reduces the displacement produced while stopping
Deceleration distance	0.80 m	Distance from the target at which the linear profile begins to decrease
Distance tolerance	0.10 m	Completion criterion of the translation phase
Orientation tolerance	0.20 rad	Completion criterion of the rotation phase
Settling interval	1.5 s	Allows the policy to come to rest before measuring
Correction passes	3	Upper bound on the closed loop correction loop
Acceptance radius	0.40 m	Residual distance within which a stalled translation is treated as arrival

The controller additionally monitors the displacement achieved since the last improvement and classifies a movement as stalled when it does not increase by a minimum increment within a fixed interval. During rotation, or when a stalled translation occurs far from the destination, the condition is reported within a few seconds rather than persisting until the execution timeout expires. A translation that stalls within a wider acceptance radius of the destination is instead treated as arrival, since the residual displacement is comparable to the imprecision already tolerated by the completion criterion.

Chapter 4

Results and Discussion

This chapter presents the experimental results obtained from the developed humanoid robotics framework. The objective of this evaluation is to assess the performance of the proposed approaches and analyze the challenges associated with deploying a humanoid robot in a simulated construction environment. The results are analyzed considering both quantitative metrics and qualitative observations regarding the behavior of the robot.

4.1 Perception Results

This section reports the outcome of the perception module in three stages. Section 4.1.1 presents the training of the YOLOv8 detector on the synthetic dataset generated with Isaac Sim Replicator. Section 4.1.2 evaluates the behavior of the detector once integrated into the ROS 2 skill and executed during simulation.

4.1.1 Training Outcome

The evolution of the training process over the 100 epochs is presented in Figure 4.1.

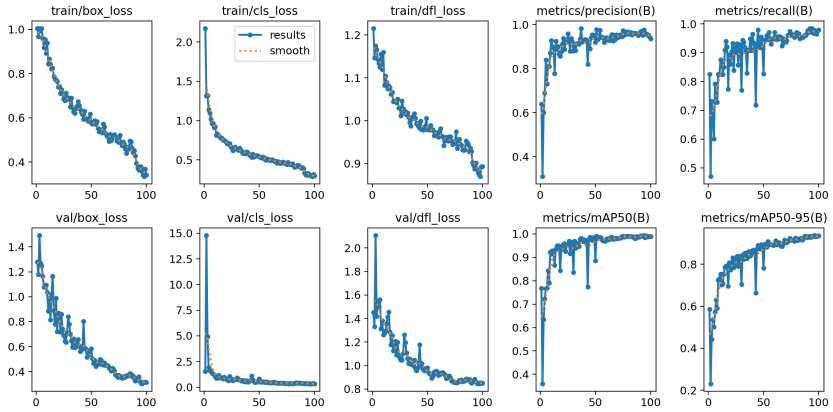


Figure 4.1: Evolution of the loss components and of the evaluation metrics during the 100 training epochs. The top row corresponds to the training subset and the bottom row to the validation subset.

Three loss components are optimized simultaneously, each addressing a distinct aspect of the detection problem. The box loss quantifies the geometric disagreement between predicted and annotated bounding boxes and therefore governs localization accuracy. The classification loss quantifies the error committed when assigning a semantic class to a detected object. The distribution focal loss refines the estimation of the box boundaries by treating each coordinate as a probability distribution rather than as a single regressed value, which sharpens the predicted edges.

All three components decrease consistently on both subsets. The decisive observation is that the validation curves track the training curves throughout, without diverging at any point. Overfitting would manifest itself as validation losses that plateau and subsequently rise while the training losses continue to fall, and no such behavior occurs. The early stopping mechanism, configured with a patience of twenty epochs, was consequently never triggered.

The pronounced peak in the validation classification loss during the initial epochs corresponds to the warm up phase, in which the detection head remains close to its random initialization and produces class scores with no discriminative value. It resolves within approximately ten epochs and does not recur.

The detector reaches a precision of approximately 0.95, a recall of approximately 0.97, a mean average precision at an intersection over union threshold of 0.5 of approximately 0.99, and a mean average precision averaged over thresholds from 0.5 to 0.95 of approximately 0.95. Precision expresses the proportion of the produced detections that are correct, whereas recall expresses the proportion of the annotated objects that the detector succeeds in finding. The mAP50 metric accepts a detection whenever it overlaps the corresponding annotation by at least half of their union, which is a comparatively permissive criterion. The mAP50 to 95 metric averages the same quantity over ten progressively stricter overlap thresholds and is therefore sensitive not only to correct classification but also to the tightness of the predicted boxes. The value of 0.95 obtained under this stricter criterion indicates that the predicted boxes are closely aligned with the annotations rather than merely coincident with the correct object.

The normalized confusion matrix computed on the validation subset is presented in Figure 4.2.

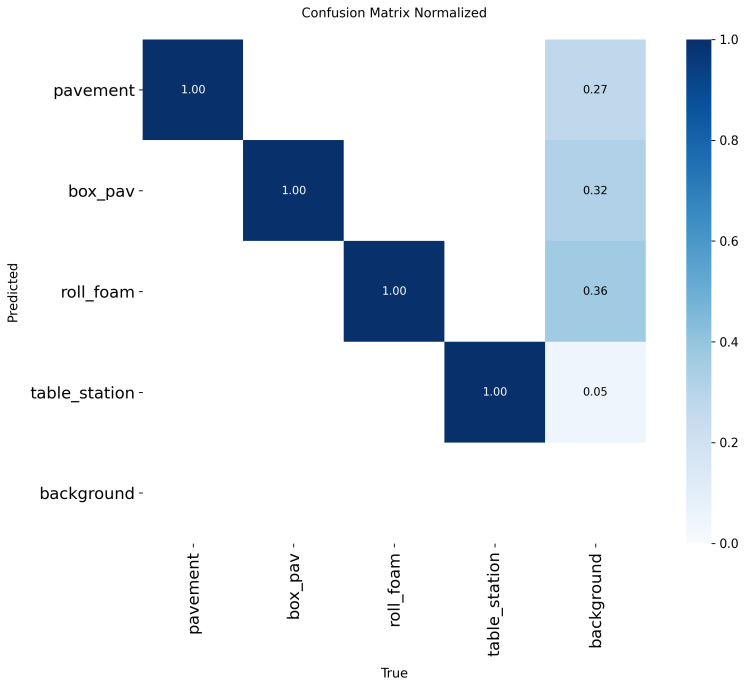


Figure 4.2: Normalized confusion matrix obtained on the validation subset. Columns correspond to the annotated class and rows to the predicted class.

The interpretation of this representation requires attention to the normalization convention, which is performed by column. Each column corresponds to a ground truth category and its entries sum to unity, so every value expresses the proportion of the instances of that category that received a given prediction.

The four columns associated with the object classes present a value of 1.00 on the diagonal and zero elsewhere. Every annotated instance of the validation subset was therefore assigned to its correct class, without a single confusion between the four categories. The absence of entries in the row labeled as background further indicates that no annotated object was missed.

The column labeled as background follows a different logic. It does not express how frequently the background is mistaken for an object in absolute terms. It expresses how the false positives, meaning the detections produced in image regions containing no annotated object, distribute themselves across the four classes. Of these spurious detections, 36 per cent were labeled as `roll_foam`, 32 per cent as `box_pav`, 27 per cent as `pavement` and 5 per cent as `table_station`. The work table is thus the class least prone to generating false detections, by a substantial margin, whereas the foam roll is the class most frequently hallucinated.

4.1.2 Detection Performance in Simulation

The detector was subsequently evaluated online through the ROS 2 skill described in Section 3.5. The skill subscribes to the RGB stream published by the simulated camera at a resolution of 1280×720 pixels and an approximate rate of eight frames per second, performs inference on every

received frame, and publishes both the annotated image and the corresponding list of detections. A confidence threshold of 0.5 was adopted, and a detection was only reported as a successful outcome after being observed in three consecutive frames, which prevents isolated spurious responses from propagating to the task manager.

The evaluation consisted in placing the robot at progressively shorter distances from the working area and recording the detections obtained in each configuration. Three representative situations are reported and the corresponding confidence values are collected in Table 4.1.

In the first configuration the robot was positioned at the far end of the working area, with the objects required by the task simultaneously within the field of view, as shown in Figure 4.3. Three of the four classes were recognized with confidence values above 0.80. The vinyl plank resting on the work table was not detected, despite being within the field of view and free from occlusion.

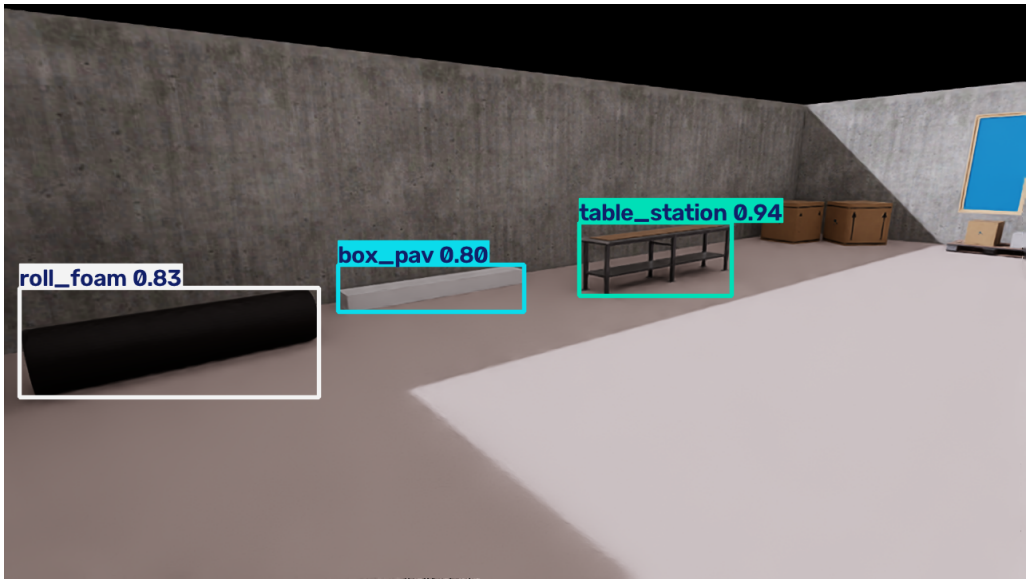


Figure 4.3: Detections obtained with the robot positioned at the far end of the working area.

In the second configuration the robot was moved closer and oriented directly towards the work table to better evaluate the perception of the flooring plank, as shown in Figure 4.4. The confidence attributed to the work table increased relative to the previous configuration, which confirms that proximity improves the quality of the detection. The vinyl plank remained undetected.

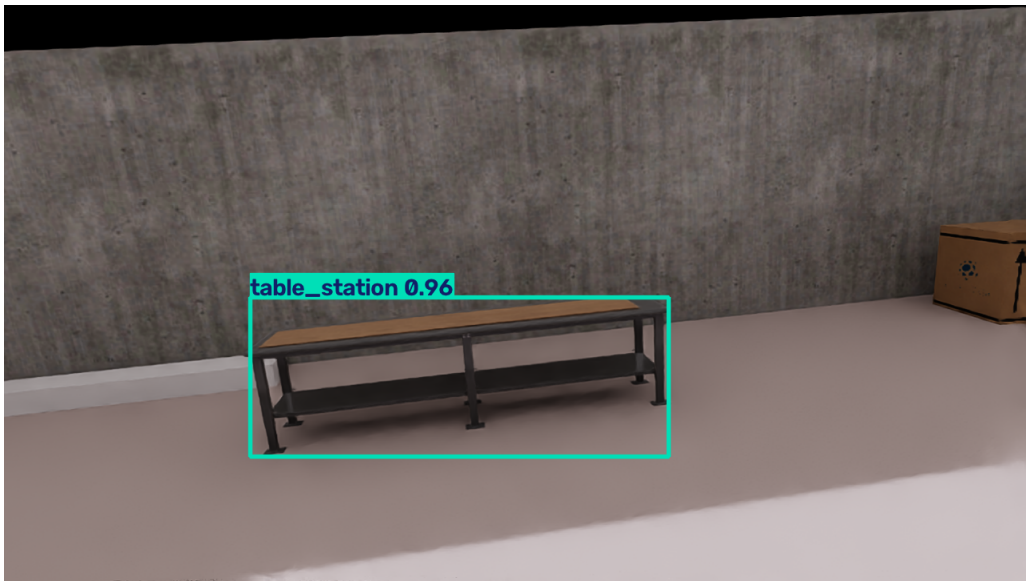


Figure 4.4: Detection of the work table with the robot positioned at an intermediate distance and facing the object directly.

In the third configuration the robot was placed immediately in front of the work table, at the distance at which a manipulation action would be attempted, as shown in Figure 4.5. The vinyl plank was detected at this range, with a predicted box that encloses the plank together with the upper surface of the table on which it rests. The associated confidence remains appreciably lower than the values obtained for the remaining classes, even under these favorable viewing conditions.



Figure 4.5: Detection of the vinyl plank resting on the work table with the robot positioned at close range.

Table 4.1: Confidence values obtained for each class as a function of the robot position.

Class	**Far**	**Intermediate**	**Close**
`table_station`	0.94	0.96	detected
`roll_foam`	0.83	out of view	out of view
`box_pav`	0.80	out of view	out of view
`pavement`	absent	absent	0.68

The pattern is consistent across the three configurations. The volumetric objects are recognized reliably at every distance at which they remain within the field of view, whereas the vinyl plank only becomes detectable once the robot has reached its final working position.

4.1.3 Discussion

The online evaluation confirms that the detector operates reliably on the objects that govern the early stages of the task, while exhibiting a single systematic failure mode that admits a coherent explanation.

Two complementary mechanisms account for the behavior of the vinyl plank. The first concerns the geometry of the observation. The plank is a thin and flat element resting on the horizontal surface of the work table. When observed from the height of the robot head, the line of sight is almost parallel to the supporting plane, so the projection of the plank onto the image collapses into a narrow band spanning very few pixels in height. The remaining objects preserve a substantial apparent height at the same distance because their extent is predominantly vertical, and they consequently retain sufficient visual evidence for the detector to operate on.

The second mechanism concerns the region of the input space that the training data actually covers. The plank viewed from several meters away falls well below that band, so the detector is required to operate at object scales for which it received no supervision. The failure is therefore not attributable to an inherent difficulty of the class, but to a mismatch between the sampling strategy adopted during dataset generation and the viewpoints that the robot actually assumes during execution. The confidence of 0.68 obtained even at close range supports this reading, since it indicates that the plank remains the least confidently recognized class under conditions that are otherwise favorable. By contrast, larger volumetric objects with more distinctive visual features remain reliably detectable over a much wider range of distances and are less susceptible to false detections.

This finding carries a direct implication for the design of the task execution strategy, and it can be converted into a design principle rather than treated as a deficiency. The work table is simultaneously the class detected most reliably at long range and the class least prone to false positives, which makes it a suitable visual landmark. Under a coarse to fine perception scheme, the materials required by the task are staged on the work table, the robot first detects and approaches the table using long range perception, and the individual materials are only resolved once the working

position has been reached. The perception requirements at each stage are thereby matched to what the detector is capable of delivering at the corresponding distance, and the limitation identified above ceases to constrain the execution of the task.

The detections produced by the perception module provide the semantic information required by subsequent stages of the architecture. However, these detections do not directly provide the three-dimensional position of the objects in the robot workspace. Estimating the metric position required by the manipulation module requires additional information, which can be obtained either from the depth channel of the Intel RealSense D455 camera registered with the RGB image, or by assuming that the object rests on a supporting plane of known height and combining this assumption with the camera intrinsic parameters and the transformation between the camera and robot base frames.

4.1.4 Limitations

The conditions under which the detector was evaluated delimit the scope of the conclusions that may be drawn from the reported metrics, and are stated here explicitly.

The objects observed during the online evaluation are the same USD assets employed to generate the training dataset, rendered by the same engine and with identical material definitions. The reported precision, recall and mean average precision therefore quantify the ability of the detector to generalize within the distribution produced by Replicator, and not its ability to generalize to the physical world. The complete absence of confusion between classes is partly attributable to the four objects being geometrically and photometrically distinct and identical across every rendered instance, a condition that no physical environment reproduces.

In a real construction environment the same materials would exhibit variation in texture, wear, soiling, illumination and partial occlusion, none of which was represented during training, and a degradation in performance should be anticipated. Narrowing this gap would require extending the randomization applied by Replicator to the material and texture properties of the objects, and subsequently fine tuning the detector on a small set of manually annotated images acquired with the physical camera.

Finally, the training data were generated at a resolution of 640×640 pixels, whereas inference is performed on frames of 1280×720 pixels published by the simulated camera. The inference framework preserves the aspect ratio through letterboxing, so no geometric distortion is introduced, but the apparent scale of the objects and the presence of padding regions differ from the conditions observed during training. Generating the dataset at the aspect ratio of the operational camera would remove this discrepancy.

4.2 Grasping Manipulation Results

The trained reinforcement learning policy was evaluated through rollout under the conditions described in Section 3.6. Task success was defined as the percentage of control steps in which the plank remained attached to the suction gripper, was lifted above the target height, maintained an

approximately horizontal orientation, and exhibited stable motion below the predefined velocity threshold. Since grasping was performed exclusively through the suction gripper, the success criterion did not require contact with the right hand, although the corresponding observations remained part of the policy input.

Table 4.2: Manipulation performance over 2048 parallel simulation environments and 5000 iterations.

Metric	Value
Success rate (%)	98.0
Suction attachment rate (%)	98.4
Plank levelness	0.982 ($\approx 11^\circ$ tilt)
Mean table intersection (mm)	0.0
Maximum table intersection (mm)	0.0

Table 4.2 shows that the learned policy performs the grasping task with a high degree of reliability. The suction gripper successfully establishes contact with the plank in 98.4% of the evaluated control steps, while the overall task success reaches 98.0%. The small difference between these values corresponds primarily to the transient interval between attachment and the plank reaching the required lifting height while satisfying the orientation and steadiness constraints.

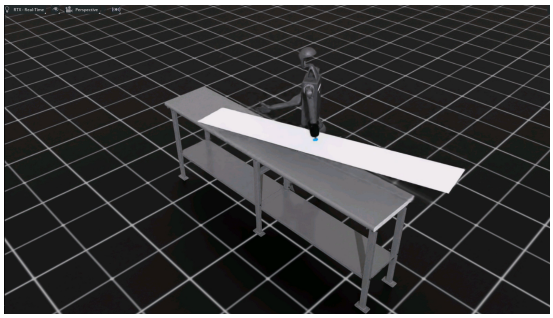
The levelness metric of 0.982 indicates that the plank is generally transported in a nearly horizontal configuration, corresponding to an average tilt of approximately 11°. Furthermore, no significant penetration between the plank and the supporting table was recorded across the evaluated environments, indicating that the learned policy consistently avoids excessive collisions during lifting.

4.2.1 Qualitative observations

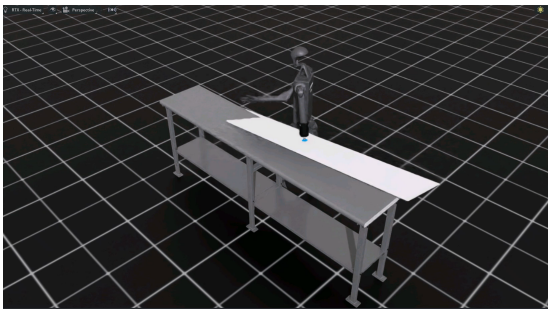
Visual inspection of the policy rollouts confirms the quantitative evaluation. The manipulator approaches the plank with the suction cup oriented downwards, establishes contact with minimal alignment error, and performs a predominantly vertical lifting motion. Throughout most executions, the plank remains stable and close to horizontal, demonstrating that the learned policy successfully coordinates arm motion and torso posture to achieve a reliable grasp.

Nevertheless, occasional rollouts revealed brief intersections between one edge of the plank and the table immediately after lift-off, as illustrated in Figure 4.6. These events do not represent exactly the aggregate metrics reported in Table 4.2. This behavior is consistent with the reward formulation presented in Section 3.6, where table penetration is penalized continuously and only terminates an episode when it exceeds the predefined threshold. Consequently, minor contacts may occur during the initial lifting transient without significantly affecting the overall performance statistics.

The tilt of the plank further contributes to this behavior. Since the penetration metric is computed using the plank's oriented geometry, even a small angular deviation may temporarily bring one edge closer to the table surface during the initial lifting motion.



(a) Plank lifted with a slight residual tilt.



(b) Temporary table intersection during lift-off.

Figure 4.6: Execution of the learned grasping policy in Isaac Lab.

4.2.2 Discussion

An important observation during development concerned the relative position of the robot with respect to the table. Initially, the robot approached the plank from the front, requiring the manipulator to reach both forwards and downwards. Owing to the limited number of actuated degrees of freedom available in the arm, this configuration constrained the orientation of the suction cup and consistently produced tilted grasps.

To address this limitation, the robot was repositioned to approach the table laterally, allowing the end-effector to descend almost vertically onto the plank. This modification substantially improved both the orientation of the suction cup and the resulting grasp quality, as shown in Table 4.3.

Table 4.3: Influence of robot stance on grasp quality.

Metric	Frontal stance	Lateral stance
Cup axis verticality	0.718	0.980
Plank levelness	0.909	0.9840
Maximum table intersection (mm)	282	0

These results indicate that grasp quality depends not only on the learned policy but also on the kinematic feasibility of the approach configuration. By improving the robot's initial stance, the policy was able to exploit a significantly more favorable workspace, resulting in flatter grasps and a substantial reduction in undesired table contact.

Another important finding concerns the action scaling factor α introduced in Section 3.6. Using $\alpha = 1.0$, the policy consistently learned to establish suction attachment but failed to lift the plank. Increasing the scaling factor to $\alpha = 2.5$ enabled the manipulator to reach the required workspace and successfully complete the lifting task without modifying the reward function.

This observation highlights that reinforcement learning performance depends not only on reward design but also on the available control authority. If the action range is insufficient to reach feasible grasping configurations, the learning algorithm cannot compensate through exploration alone.

4.2.3 Limitations

Some limitations remain in the current manipulation framework. First, suction attachment is modeled kinematically and therefore does not transmit reaction forces or object inertia to the manipulator. Consequently, the learned policy never experiences the dynamic effects associated with carrying the plank, which would become relevant for deployment on physical hardware.

Also, the policy was trained using a fixed-base robot and therefore does not account for locomotion or balance during manipulation.

4.3 Locomotion and Navigation Results

This section presents the evaluation of the humanoid robot's locomotion capabilities and the performance of the navigation strategies introduced in Section 3.7.

4.3.1 Experimental protocol

The two navigation strategies were evaluated over a common set of routes. Under the Nav2 strategy the pose of the robot was read immediately before each trial and the goal computed relative to it, so that every trial covers the same distance regardless of the drift accumulated since the simulation was started. Under the relative strategy no such precaution is required, since the goal is expressed with respect to the current pose by construction.

Four routes were used. The first consisted of a straight-line translation of 2 m along the initial heading, assessing the ability to travel accurately in a straight line and stop at a predefined target position. The second consisted of an in-place rotation of 90°, evaluating orientation control without translational motion. The third and fourth consisted of a diagonal displacement of 3.54 m, equivalent to 2.5 m along each axis, requiring a change of heading of 45° followed by realignment on arrival. The third route curves to the right and the fourth to the left. The two senses were separated because the platform does not behave symmetrically, as discussed in Section 4.3.5.3.

Each route was executed five times per strategy, with the exception of the left-hand diagonal, which was executed once for the reason given in Section 4.3.6.1. For each trial the completion status, execution time and final pose were recorded automatically by the skill server. The position error is the Euclidean distance between the final and target positions, and the orientation error the normalised angular difference between the final and target orientations.

The simulation does not advance in real time and the ratio is not constant: the real-time factor recorded during the simulation ranged from 0.24 to 0.37 under Nav2 and from 0.32 to 0.46 under the relative strategy, the difference reflecting the computational load of the navigation stack. Therefore, all durations are reported in simulated time, obtained from the ROS clock with `use_sim_time` enabled.

Both strategies were configured with the parameters listed in Tables 3.6 and 3.7.

Table 4.4: Straight translation of 2 m. Mean ± standard deviation over five trials, with the observed range in brackets.

	Nav2	**Relative**
Position error (m)	0.163 ± 0.028 [0.143, 0.213]	0.093 ± 0.118 [0.014, 0.301]
Orientation error (deg)	9.6 ± 1.1 [7.9, 10.5]	5.4 ± 1.9 [3.2, 8.3]
Walking time (sim, s)	6.5 ± 0.2	9.4 ± 0.2
Alignment time (sim, s)	3.8 ± 0.8	—
Total time (sim, s)	10.3	11.5 ± 0.2

4.3.2 Straight line

The results of the straight-line translation, presented in Table 4.4, show that both strategies completed every trial in comparable total time, and the relative controller attained the lower error in both position and orientation.

The two strategies differ markedly in dispersion. The Nav2 error is tightly grouped, at 0.028 m against a mean of 0.163 m, and decomposing it shows both components to be systematically negative: the robot finishes short of the target and displaced to one side in every trial, which is the signature of a compensation that is applied uniformly and matches the incurred displacement only approximately.

The distribution of the relative controller's error is markedly different. Four of the five trials fall between 0.014 m and 0.069 m, while the remaining trial reached 0.301 m, almost entirely as a lateral deviation of 0.295 m. The standard deviation consequently exceeds the mean, and the aggregate figure describes neither the typical trial nor the exceptional one. Considered alone, the four consistent trials yield 0.041 ± 0.023 m.

The outlying trial is retained in the reported statistics, as its occurrence is itself informative. The closed-loop correction described in Section 3.7.6.2 acts upon the distance travelled along the commanded heading and therefore recovers longitudinal error, but it provides no equivalent mechanism for lateral deviation, which is corrected only by the proportional heading term during the translation itself. A deviation that develops faster than that term can compensate is consequently carried to the end of the movement, since the controller has no global reference against which it could be detected and removed.

4.3.3 Rotation in place

Table 4.5 shows the results of in-place rotations of 90°. A biped rotating in place does not pivot about a fixed point: the base frame describes a small arc with each step of the rotation. The displacement of 0.057 m under Nav2 and 0.146 m under the relative strategy quantifies this effect directly, and accounts for part of the position error observed on routes that terminate with an alignment phase.

The displacement is approximately two and a half times larger under the relative strategy. The difference is attributable to the closed-loop verification described in Section 3.7.5.4: the

Table 4.5: In-place rotation of 90°. Mean ± standard deviation over five trials, with the observed range in brackets.

	Nav2	**Relative**
Displacement (m)	0.057 ± 0.022 [0.018, 0.072]	0.146 ± 0.028 [0.123, 0.194]
Orientation error (deg)	9.8 ± 1.3 [8.2, 11.4]	10.2 ± 1.9 [7.1, 12.2]
Total time (sim, s)	8.2 ± 1.0	8.0 ± 0.6
Correction passes	2.0 ± 0.7	1

Nav2 implementation measures the attitude after settling and repeats the correction if the residual error exceeds tolerance, typically converging in two passes, whereas the relative implementation terminates on the internal criterion of its rotation loop without remeasuring. Each additional pass under Nav2 is a short rotation that displaces the robot less than the original one, so the net displacement is smaller even though more rotations are performed.

The orientation errors are comparable and both lie close to 10°, which is the accuracy limit discussed in Section 4.3.5.2.

4.3.4 Diagonal route

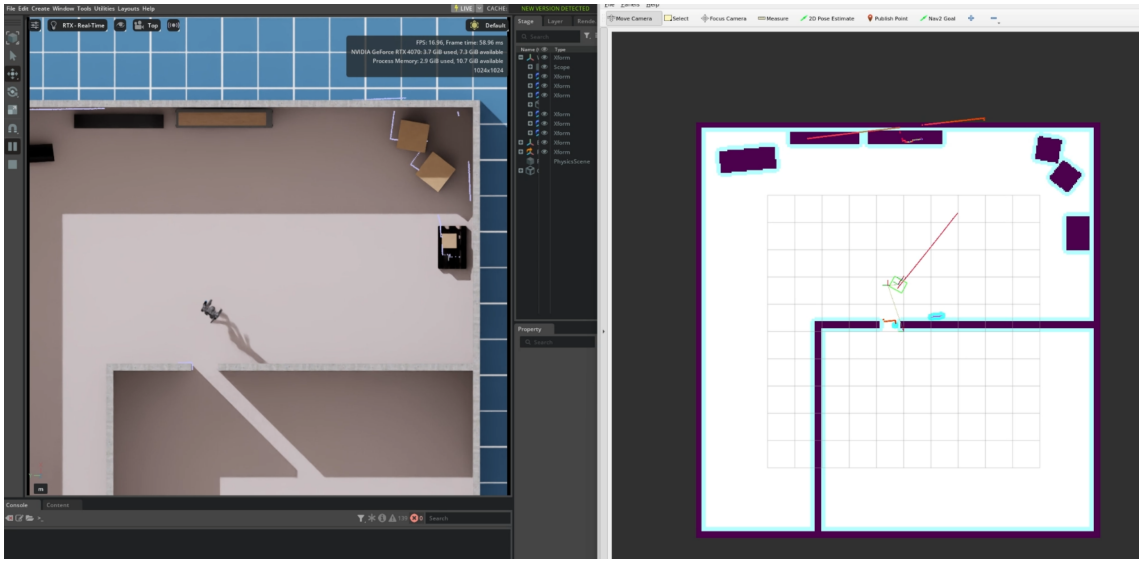


Figure 4.7: The diagonal route being executed under Nav2. The Isaac Sim viewport shows the robot in the simulated environment, while RViz shows the same situation as represented by the navigation stack.

Figure 4.7 shows the system executing this route with Nav2. The two windows present the same situation from the perspective of the simulation and from that of the navigation stack, the latter being the representation upon which the planner and the controller act.

Table 4.6: Diagonal displacement of 3.54 m with curvature to the right. Mean ± standard deviation over five trials, with the observed range in brackets.

	Nav2	**Relative**
Position error (m)	0.055 ± 0.028 [0.022, 0.095]	0.371 ± 0.034 [0.342, 0.428]
Orientation error (deg)	10.2 ± 0.9 [9.1, 11.2]	7.4 ± 4.2 [5.0, 14.8]
Walking time (sim, s)	11.2 ± 1.0	12.5 ± 0.4
Alignment time (sim, s)	4.5 ± 0.3	2.9 ± 0.2
Total time (sim, s)	15.7	21.4 ± 0.2

The results of the diagonal route are presented in Table 4.6. On this route Nav2 attained a position error of 0.055 m, lower than on the straight route of 2 m despite the greater distance and the change of heading. The relative controller attained 0.371 m, approximately seven times larger, and required a third more time to complete the route.

The orientation errors bracket the same value from opposite sides. Under Nav2 the error is tightly grouped around 10.2°, which corresponds to the residual left by the alignment phase. Under the relative controller the mean is lower but the dispersion four times larger, reflecting that the orientation attained is whatever the final rotation happens to leave rather than the outcome of a verified correction.

Table 4.7: Diagonal displacement of 3.54 m with curvature to the left. Single trial per strategy.

	Nav2	**Relative**
Outcome	succeeded	succeeded
Position error (m)	0.673	0.301
Orientation error (deg)	9.3	5.6
Heading error on arrival (deg)	102.6	—
Closest approach (m)	0.400	—
Total time (sim, s)	19.5	22.3

The same displacement was then executed with the curvature reversed to the left, and the results are shown in Table 4.7. Nav2 terminated 0.673 m from the destination, thirteen times the error obtained on the mirror-image route, and never came closer than 0.400 m to the final waypoint, against a typical closest approach of 0.09 m on the right-hand route. The robot arrived 102.6° from the requested heading, having described a curvature rather than converging onto the path. The relative controller produced 0.301 m, within the range observed on the right-hand route, and showed no comparable degradation.

4.3.5 Discussion

4.3.5.1 Position accuracy

The position errors of the two strategies are not produced by the same mechanism, and the comparison depends on the route.

Under Nav2 the error is bounded by the acceptance criterion and by the compensation applied to it. The navigation goal is released when the robot enters a radius of 0.10 m of the final waypoint, and that waypoint is itself offset beyond the destination by 0.14 m along the arrival tangent, to absorb the displacement that occurs between release and rest. The resulting error reflects how well that offset matches the displacement actually incurred, rather than the tracking performance of the controller. Because the offset is a single calibrated constant applied uniformly, the error it leaves is small and consistent but does not tend to zero.

Under the relative controller the error arises from execution, and its magnitude depends on the velocity floor. The floor exists because the locomotion policy does not initiate a gait below a certain command magnitude, and its value proved decisive: at $0.15\,\mathrm{m\,s^{-1}}$ the robot ceased to advance during the final approach and stopped short of the destination, producing errors of 0.265 ± 0.051 m; at $0.30\,\mathrm{m\,s^{-1}}$ the approach completes and the closed-loop correction passes converge, reducing the typical error to a few centimetres. The value reported in Table 3.7 is the latter, and the figures in Table 4.4 correspond to it.

With the floor correctly set, the relative controller is the more accurate of the two on a straight route, since it measures the displacement actually achieved and corrects it, whereas Nav2 applies a fixed compensation. The ordering reverses on routes with curvature, for the reason examined in Section 4.3.5.3: there the dominant error is not the stopping displacement but the heading error accumulated while turning, which the relative controller does not correct against any global reference.

4.3.5.2 Orientation accuracy

Orientation errors cluster around 10° in both strategies and do not fall below that value regardless of the number of correction passes applied.

The cause is visible in the instrumentation. The rotation loop terminates when the error measured during the rotation falls below tolerance, typically at 11°, but the attitude measured after the robot settles is consistently worse, between 10° and 14°. The robot recovers part of the rotation as the feet come to rest, so the attitude adopted and the attitude at which the loop terminated do not coincide. Repeating the correction does not converge below this value: in one trial three successive passes produced residual errors of 13.1°, 13.1° and 11.6°, each pass rotating the robot to within tolerance and each settling back to approximately the same attitude.

The orientation tolerance was therefore set to 0.20 rad, or 11.5°. A tighter value is not merely difficult to attain but unattainable, and an unattainable criterion causes every movement to be reported as a failure even when the destination has been reached to within a few centimetres.

4.3.5.3 Directional asymmetry

The most pronounced result of the test is that Nav2 executes a diagonal displacement accurately in one sense and not in the other, while the relative controller executes both equivalently.

The discrepancy is not attributable to the platform alone. Commanding angular velocity directly, the robot rotates in both senses, though measurably faster in one than the other; and the relative controller, which issues such commands without any path-following law, produced 0.371 m to the right against 0.301 m to the left, a difference within the dispersion of either. The asymmetry of the platform is therefore present in both strategies, but only one of them amplifies it by an order of magnitude.

In the unfavourable sense the robot does not follow the planned path but describes an arc of approximately constant curvature, since the pursuit controller issues a persistent angular velocity alongside the linear command. The trajectory is circular rather than straight, and the robot approaches the destination only in so far as the arc happens to pass near it. The lookahead point was recorded at up to 86° from the heading of the robot while the planned path ran directly towards the destination, which indicates a controller pursuing a target it is not converging upon.

On a route of the length tested this degrades the outcome without preventing it. With the angular velocity bounded at $0.25\,\mathrm{rad\,s^{-1}}$ and a linear velocity of $0.50\,\mathrm{m\,s^{-1}}$, the arc has a radius of approximately 2 m, so a displacement of 3.54 m traverses barely a quarter of its circumference. The robot deviates from the path and arrives 102.6° from the requested heading, but it does arrive, and the alignment phase recovers the orientation afterwards. The reported error is the accumulated deviation of that arc rather than a failure to complete the movement.

The behaviour does not scale with distance. A longer route would traverse a correspondingly larger fraction of the arc, and once the accumulated rotation carries the robot away from the destination the lateral offset of the lookahead point grows rather than diminishes, sustaining the same curvature command. The route length at which the trajectory would close upon itself was not determined experimentally, although nothing in the loop acts to reduce the offset that sustains it.

The cause was not established. The lookahead distance was varied across four values and the angular velocity limit, the inflation radius, the composition of the local costmap layers, the search window used to locate the robot on the path and the segmentation of the route into shorter segments were all changed without altering the outcome.

The observations do establish that neither layer accounts for the behaviour alone. The locomotion policy executes rotations in both senses when commanded directly, and the relative controller, which issues such commands without a path-following law, shows no comparable difference between them. The pursuit controller, conversely, produces an accurate trajectory in one sense on the same platform. The behaviour arises from the combination, and identifying the mechanism would require instrumenting the controller internals rather than observing its output.

4.3.5.4 Execution time

The time comparison does not favour either strategy uniformly. The two are equivalent on the straight route and on the rotation, while on the diagonal Nav2 is approximately a quarter faster.

The relative controller pays a fixed overhead per phase. Each rotation and each translation is followed by a settling interval and, in the case of translation, by a closed-loop correction pass. A route composed of three phases therefore carries three such intervals. Nav2 carries a fixed cost in planning and in the alignment phase, but executes the translation as a single continuous movement.

The pattern that emerges is that the relative controller is efficient for single primitives and accumulates overhead when primitives are chained, while Nav2 amortises its fixed cost over longer routes.

4.3.6 Limitations

4.3.6.1 Limitations of the experimental campaign

The reactive component of the framework was not evaluated. The range sensor and the obstacle layer were operational throughout the simulation, but the routes tested contained no obstacles absent from the static map, so the ability of the system to avoid an unmapped obstacle was never exercised. Establishing that capability requires a dedicated test with obstacles introduced along the planned path.

4.3.6.2 Limitations of the system

The transformation between the `map` and `odom` frames is a calibrated constant rather than a corrected estimate. Adaptive Monte Carlo Localization (AMCL) was configured and tested but did not converge reliably, and was consequently not used. Any residual error in that calibration appears as a systematic offset in every measurement expressed in the map frame, and it is not separable from the execution error of the Nav2 strategy with the data collected. The relative strategy is not affected, since it does not reference the map frame.

The static layer was removed from the local costmap and the inflation radius reduced to 0.12 m, both to prevent the collision checker from reporting obstacles where none exist. These changes make the local representation less conservative than the Nav2 defaults, and the experiments were conducted in an environment free of obstacles along the routes tested; the configuration has not been validated in the presence of unmapped obstacles.

4.3.6.3 Limitations inherited from the locomotion policy

The locomotion policy supports neither reverse nor lateral motion, which removes an entire class of recovery manoeuvres and constrains the behaviour tree configuration. It imposes a lower bound on the commanded velocity, below which no gait is initiated, and this bound is what prevents the relative controller from decelerating continuously to rest. It tracks approximately a quarter of the commanded velocity in both translation and rotation, and does so unequally in the two senses of

rotation, which is the origin of the asymmetry examined in Section 4.3.5.3. Finally, it recovers part of any rotation as the feet settle, which bounds the attainable orientation accuracy at approximately 10°.

The simulation ran at between a quarter and a half of real time depending on the strategy under evaluation. Beyond lengthening the campaign, this factor motivated deriving the timing of both controllers from the simulation clock rather than from the wall clock, as described in Section 3.7.6.2.

Chapter 5

Conclusions and Future Work

This dissertation presented the development of a modular robotic framework that integrates perception, locomotion, and manipulation capabilities to enable a humanoid robot to perform representative construction tasks in simulation. Vinyl flooring installation was adopted as the application scenario because it requires the coordinated execution of autonomous navigation, object perception, material handling, and task sequencing within a realistic construction environment.

A complete simulation environment was developed in NVIDIA Isaac Sim to reproduce an indoor flooring installation scenario and to provide a common platform for perception, navigation, and manipulation experiments. The same environment was also employed to generate a synthetic dataset for object detection, allowing the perception system to be trained without manual image annotation.

Object perception was implemented through a YOLOv8-based detector trained on synthetic data generated with Isaac Sim Replicator and integrated into the robotic architecture as a ROS 2 perception skill. Manipulation was formulated as a reinforcement learning problem in Isaac Lab, resulting in a policy capable of grasping and lifting vinyl flooring planks using a suction gripper. Autonomous locomotion was achieved by combining a learned locomotion policy with two navigation strategies, enabling the humanoid robot to execute goal-directed movements within the simulated environment.

Finally, the locomotion and perception components were integrated into a modular skill-based architecture coordinated by the Open Scalable Production System Task Manager. This architecture provides a unified execution framework in which perception, locomotion, and manipulation remain independent yet interoperable components, facilitating future extensions and the incorporation of additional robotic capabilities.

5.1 Answering the Research Questions

This section revisits the research questions introduced in Section 1.4 and discusses how the proposed framework and the experimental evaluation address each of them.

RQ1: How can autonomous locomotion and navigation be integrated to enable a humanoid robot to safely navigate a realistic construction environment?

The proposed framework addresses this question by decoupling locomotion from high-level navigation. A learned locomotion policy generates stable walking motions from commanded linear and angular velocities, while navigation is performed by standard ROS 2 components operating upon that velocity interface. This abstraction permits the humanoid to be controlled as a mobile platform, and navigation algorithms developed for wheeled robots to be applied without modification to the underlying locomotion controller.

Two navigation strategies were developed and evaluated within this architecture. The first employs the Nav2 framework operating upon a global map; the second executes translation and rotation primitives defined relative to the current pose, in the odometry frame. Neither proved superior across all conditions. Each attained the lower error in a distinct regime, and in each case the advantage follows from how the strategy addresses the error source that dominates there. On a straight translation the relative controller was the more accurate, since it measures the displacement actually achieved and corrects it in closed loop, whereas Nav2 compensates the stopping displacement through a fixed calibrated offset. On a route requiring a change of heading the ordering reversed, the dominant error being no longer the stopping displacement but the heading deviation accumulated while turning, which only a controller referenced to a global frame is able to correct.

Taken together, these results indicate that map-based navigation is the more appropriate solution for general navigation tasks in construction environments, provided that this coupling is characterised for the platform in question. The relative controller retains its utility for short-range positioning operations, such as the final alignment preceding manipulation, where per-movement accuracy is the governing requirement and the absence of a global reference presents no limitation.

Safety, in the sense of avoiding obstacles absent from the map, was not demonstrated. The reactive component of the framework is integrated but was not exercised, the routes tested having contained no such obstacles. What the experiments do establish is that the robot traverses a mapped environment without collision, the footprint and inflation configuration having been adapted to the geometry of the humanoid to that end.

RQ2: How can reinforcement learning be used to enable reliable manipulation of thin vinyl flooring planks?

The manipulation problem was formulated as a reinforcement learning task in Isaac Lab, where the robot learned to approach, grasp, and lift a vinyl flooring plank using a suction gripper. Instead of relying solely on a terminal success signal, the reward function was designed to encourage successive stages of the manipulation process, including approaching the target, establishing suction attachment, lifting the plank, maintaining an approximately horizontal orientation, and avoiding collisions with the supporting table.

The trained policy demonstrated that reinforcement learning provides an effective solution for this manipulation task, achieving a high grasp success rate under the evaluation conditions considered in this work. More importantly, the experimental analysis revealed that policy performance

depends not only on reward design but also on the physical configuration of the task.

Two observations were found to be particularly significant. First, the relative position of the robot with respect to the work table strongly influences the feasibility of the grasping motion. A lateral approach enables the end-effector to descend almost vertically towards the plank, substantially reducing collisions with the table and improving grasp quality. Second, increasing the action scaling factor expanded the reachable action space and enabled successful lifting without requiring modifications to the reward function.

These findings indicate that successful reinforcement learning depends not only on reward engineering but also on appropriate task design and action-space definition. The learning algorithm can only discover successful behaviors when physically feasible solutions are accessible within the exploration space.

The proposed manipulation policy was evaluated under simplified simulation assumptions. The robot base remained fixed throughout training, suction attachment was modeled kinematically, and the subsequent transport and placement of the plank were not considered. Extending the policy to a mobile humanoid platform therefore remains an important direction for integrating this capability in the simulation environment.

RQ3: How can vision-based object detection be integrated into a humanoid robotic platform?

The proposed perception pipeline demonstrates that synthetic data generated directly from the simulation environment can be successfully used to train an object detector for robotic applications. NVIDIA Isaac Sim Replicator was employed to automatically generate annotated training data, which was subsequently used to fine-tune a YOLOv8 detector. The trained model was deployed as a ROS 2 perception node that continuously processes RGB images acquired by the onboard camera and publishes object detections to the remaining components of the framework.

The offline evaluation showed that the detector achieves excellent performance on the validation dataset, demonstrating the effectiveness of synthetic data for training the selected object classes. However, the online experiments highlighted an important limitation. Whereas larger objects remained reliably detectable over a wide range of distances, the thin vinyl flooring plank was only consistently detected when the robot approached the working area. This behavior results from both the limited apparent size of the plank in distant viewpoints and the characteristics of the generated synthetic dataset.

These observations indicate that successful integration of vision-based perception depends not only on the detector itself but also on ensuring that the synthetic training data accurately represent the viewpoints and object scales encountered during robot operation.

RQ4: How can locomotion, perception, and manipulation capabilities be combined to support representative construction tasks in a simulated environment?

The proposed framework combines the developed capabilities through a modular ROS 2 skill-based architecture coordinated by the OSPS. Each capability is implemented as an independent skill with a standardized interface, allowing these components to be executed sequentially without requiring direct dependencies between their internal implementations.

The experimental work demonstrated that this architecture successfully integrates heterogeneous technologies, including reinforcement learning policies, vision-based perception, and the ROS 2 navigation stack, within a unified execution framework. This modular design also facilitates future system extensions, as individual skills can be replaced or improved without affecting the remaining components.

Although the proposed architecture establishes the foundation for autonomous flooring installation, complete end-to-end task execution has not yet been achieved. Metric localization of detected objects, integration of the manipulation policy with the mobile platform, and autonomous plank placement remain necessary before the complete construction workflow can be executed autonomously.

Overall, the proposed framework demonstrates that combining perception, locomotion, and manipulation through a modular architecture provides a practical and extensible foundation for humanoid robotic applications in construction environments.

5.2 Future Work

Although the proposed framework demonstrates the feasibility of integrating perception, locomotion, and manipulation capabilities for humanoid robots in simulated construction environments, several challenges remain before the complete flooring installation task can be executed autonomously. The following directions represent the most relevant extensions of the work presented in this dissertation.

5.2.1 Consolidation of the Experimental Evaluation

The experimental evaluation presented in this dissertation provides an initial validation of the proposed framework, however, a more comprehensive experimental campaign is required to strengthen the statistical significance of the reported results. Navigation experiments should be repeated multiple times under identical conditions to compute performance indicators such as mean error, standard deviation, and confidence intervals, allowing a more rigorous comparison between navigation strategies.

Similarly, the perception module should be evaluated on the held-out test dataset in addition to the validation set used during model development, providing an unbiased assessment of the detector's generalization capability.

Evaluating the obstacle avoidance capability of the framework is a natural next step. With the range sensor and the obstacle layer already integrated, such an evaluation would require only the design of a suitable experimental campaign, in which obstacles absent from the static map are introduced along the planned path so that the reactive component of the navigation stack may be characterised alongside the results reported here. Likewise, improving the robustness of the localization system would eliminate the need for manually calibrated map-to-odometry transformations and provide a more reliable estimate of navigation accuracy.

5.2.2 From Two-Dimensional Detection to Metric Localization

The current perception pipeline provides object identity together with image-space localization through two-dimensional bounding boxes. However, autonomous manipulation requires the three-dimensional position of each detected object to be estimated in the robot reference frame.

A natural extension consists of combining RGB detections with depth information acquired by the Intel RealSense D455 camera. By projecting the detected image coordinates into three-dimensional space using the camera intrinsic parameters, the perception module would be able to provide metric object positions that could be directly consumed by the manipulation system. Alternatively, object positions may be estimated by exploiting prior knowledge of the supporting plane on which the construction materials are placed. Both approaches provide a practical solution for connecting perception with autonomous manipulation.

The synthetic dataset should also be extended to better represent the operating conditions encountered during robot execution. Increasing the variability of viewing distances, camera poses, illumination conditions, and material textures would improve detector robustness. Furthermore, fine-tuning the model using a small number of manually annotated images acquired from the physical robot would help reduce the domain gap between simulation and real-world operation.

5.2.3 Integration of Manipulation with Mobile Locomotion

Although locomotion and manipulation were successfully developed independently, they have not yet been executed simultaneously on the humanoid robot. Integrating the trained grasping policy into the skill-based architecture constitutes the next step towards fully autonomous operation.

Once integrated, the robot will be able to navigate towards the work table, identify the required material, execute the grasping policy, and transport the plank to the installation area. This scenario introduces additional challenges that were not considered during policy training, including the interaction between locomotion and manipulation, payload-induced disturbances, and the dynamic behavior of the transported object.

Future work should therefore incorporate a physically realistic suction model capable of transmitting contact forces and object inertia. In addition, the manipulation policy should be extended to perform the reorientation required to transport the plank safely before placement.

5.2.4 Autonomous Vinyl Flooring Installation

To fully evaluate the capability of a humanoid robot in performing the selected autonomous task, the complete autonomous execution of the vinyl flooring installation must be achieved. Beyond transporting the material, the robot must learn to position the plank accurately on the floor, align it with previously installed elements, and release the suction gripper while maintaining the required placement accuracy.

This problem constitutes a natural extension of the reinforcement learning approach adopted for grasping. Similar design considerations are expected to remain relevant, particularly regarding robot positioning, action-space definition, and task parameterization.

Once all capabilities are integrated, the complete workflow can be executed through the OSPS, coordinating autonomous navigation, object perception, manipulation, transport, and installation within a unified execution pipeline. Such an end-to-end demonstration would provide the final validation of the proposed framework and enable performance indicators directly related to construction applications, including task completion time, placement accuracy, and overall system robustness.

References

[1] "Labor markets: Tight in advanced economies | McKinsey," Accessed: Jan. 2, 2026. [Online]. Available: https://www.mckinsey.com/mgi/our-research/help-wanted-charting-the-challenge-of-tight-labor-markets-in-advanced-economies

[2] "Improving construction productivity is the new imperative | McKinsey," Accessed: Jan. 2, 2026. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/delivering-on-construction-productivity-is-no-longer-optional

[3] "Humanoid robots in the construction industry | McKinsey," Accessed: Jan. 2, 2026. [Online]. Available: https://www.mckinsey.com/industries/engineering-construction-and-building-materials/our-insights/humanoid-robots-in-the-construction-industry-a-future-vision

[4] "Can Humanoid Robotics Beat The Construction Labour Crisis?" Default, Accessed: Jan. 2, 2026. [Online]. Available: https://www.verdantix.com/client-portal/blog/can-humanoid-robotics-beat-the-construction-labour-crisis

[5] P. Hanbury, A. Dutt, and E. Veratti. "Humanoid Robots at Work: What Executives Need to Know," Bain, Accessed: Jan. 2, 2026. [Online]. Available: https://www.bain.com/insights/humanoid-robots-at-work-what-executives-need-to-know/

[6] L. Pall. "Portugal's First 3D Printed House Demonstrates Speed and Cost Efficiency of 3D Construction Printing," COBOD, Accessed: Jan. 4, 2026. [Online]. Available: https://cobod.com/portugals-first-3d-printed-house-demonstrates-speed-and-cost-efficiency-of-3d-construction-printing/

[7] "Hadrian® | Automated Construction Robot from FBR," FBR, Accessed: Jan. 9, 2026. [Online]. Available: https://www.fbr.com.au/view/hadrian

[8] "TyBOT | The Rebar Tying Robot | by Advanced Construction Robotics," Construction Robots, Accessed: Jan. 4, 2026. [Online]. Available: https://www.constructionrobots.com/tybot

[9] "BIM-Driven Layout for Construction | Dusty Robotics," Accessed: Jan. 4, 2026. [Online]. Available: https://www.dustyrobotics.com

[10] C. Brosque and M. Fischer, "Safety, quality, schedule, and cost impacts of ten construction robots," *Construction Robotics*, vol. 6, no. 2, pp. 163–186, Jun. 2022, ISSN: 2509-811X, 2509-8780. DOI: 10.1007/s41693-022-00072-5 Accessed: Jan. 4, 2026. [Online]. Available: https://link.springer.com/10.1007/s41693-022-00072-5

[11] H. Portugal. "Robot de perfuração semiautónomo Jaibot," Accessed: Jan. 4, 2026. [Online]. Available: https://www.hilti.pt/content/hilti/E2/PT/pt/business/business/trends/jaibot-semi-autonomous-drilling-robot.html

[12] "BauBot - fischer international," Accessed: Jan. 4, 2026. [Online]. Available: https://www.fischer-international.com/en/products/innovations/baubot

[13] "Schindler R.I.S.E," Fabricante de elevadores, escadas e tapetes rolantes, Accessed: Jan. 4, 2026. [Online]. Available: https://www.schindler.pt/pt/inovacoes/schindler-rise

[14] "Canvas – Drywall Construction Robotics," Canvas, Accessed: Jul. 31, 2026. [Online]. Available: https://canvas.build/

[15] "OKIBO-Smart Robots For Construction Sites," OKIBO, Accessed: Jan. 4, 2026. [Online]. Available: https://okibo.com/

[16] "Bzl-en," Accessed: Jan. 5, 2026. [Online]. Available: https://www.bzlrobot.com/en/index.html

[17] S. Zhao et al., "Application and Development of Autonomous Robots in Concrete Construction: Challenges and Opportunities," *Drones*, vol. 6, no. 12, Dec. 16, 2022, ISSN: 2504-446X. DOI: 10.3390/drones6120424 Accessed: Dec. 1, 2025. [Online]. Available: https://www.mdpi.com/2504-446X/6/12/424

[18] "Robots that Build the World," Built Robotics, Accessed: Jan. 4, 2026. [Online]. Available: https://www.builtrobotics.com/

[19] "Spot," Boston Dynamics, Accessed: Jan. 4, 2026. [Online]. Available: https://bostondynamics.com/products/spot/

[20] Institution of Structural Engineers, "An introduction to building information modelling (bim)," Institution of Structural Engineers, London, Tech. Rep., 2021, Accessed: 11 January 2026. [Online]. Available: https://www.istructe.org/IStructE/media/Public/An_introduction_to-Building_Information_Modelling_BIM.pdf

[21] H. Yue, Q. Wang, Z. Zhao, S. Lai, and G. Huang, "Interactions between BIM and robotics: Towards intelligent construction engineering and management," *Computers in Industry*, vol. 169, p. 104 299, Aug. 2025, ISSN: 01663615. DOI: 10.1016/j.compind.2025.104299 Accessed: Jan. 8, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0166361525000648

[22] K. Kim and M. Peavy, "BIM-based semantic building world modeling for robot task planning and execution in built environments," *Automation in Construction*, vol. 138, p. 104 247, Jun. 2022, ISSN: 09265805. DOI: 10.1016/j.autcon.2022.104247 Accessed: Jan. 11, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0926580522001200

[23] S. Kim, M. Peavy, P.-C. Huang, and K. Kim, "Development of BIM-integrated construction robot task planning and simulation system," *Automation in Construction*, vol. 127, p. 103 720, Jul. 2021, ISSN: 09265805. DOI: 10.1016/j.autcon.2021.103720 Accessed: Jan. 11, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0926580521001710

[24] Y. Tong, H. Liu, and Z. Zhang, "Advancements in Humanoid Robots: A Comprehensive Review and Future Prospects," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 2, pp. 301–328, Feb. 2024, ISSN: 2329-9274. DOI: 10.1109/JAS.2023.124140 Accessed: Jan. 5, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10415857/

[25] M. Zetter. "Humanoid robots: Key Players in the Humanoid Value Chain." Source: Goldman Sachs Global Investment Research, LinkedIn, Accessed: Jul. 31, 2026. [Online]. Available: https://www.linkedin.com/posts/mzetter_humanoid-robots-key-players-in-humanoid-activity-7379878639707975680-5uBU/

[26] "AgiBot – Official Website," AgiBot, Accessed: Jul. 31, 2026. [Online]. Available: https://www.agibot.com/

[27] "Fourier Intelligence – Official Website," Fourier Intelligence, Accessed: Jul. 31, 2026. [Online]. Available: https://www.fftai.com/

[28] "Unitree Robotics – Official Website," Unitree Robotics, Accessed: Jul. 31, 2026. [Online]. Available: https://www.unitree.com/

[29] "Figure AI – Official Website," Figure AI, Accessed: Jul. 31, 2026. [Online]. Available: https://www.figure.ai/

[30] "Apptronik – Official Website," Apptronik, Accessed: Jul. 31, 2026. [Online]. Available: https://apptronik.com/

[31] "Agility Robotics – Official Website," Agility Robotics, Accessed: Jul. 31, 2026. [Online]. Available: https://www.agilityrobotics.com/

[32] "NVIDIA – Official Website," NVIDIA, Accessed: Jul. 31, 2026. [Online]. Available: https://www.nvidia.com/

[33] "Intel – Official Website," Intel, Accessed: Jul. 31, 2026. [Online]. Available: https://www.intel.com/

[34] "Sanhua – Official Website," Sanhua, Accessed: Jul. 31, 2026. [Online]. Available: https://www.sanhuagroup.com/en/

[35] "Harmonic Drive – Official Website," Harmonic Drive, Accessed: Jul. 31, 2026. [Online]. Available: https://www.harmonicdrive.net/

[36] "LeaderDrive – Official Website," LeaderDrive, Accessed: Jul. 31, 2026. [Online]. Available: https://www.leaderdrive.com/

[37] "Maxon – Official Website," Maxon, Accessed: Jul. 31, 2026. [Online]. Available: https://www.maxongroup.com/

[38] "MOONS' – Official Website," MOONS' Industries, Accessed: Jul. 31, 2026. [Online]. Available: https://www.moonsindustries.com/

[39] "DJI – Official Website," DJI, Accessed: Jul. 31, 2026. [Online]. Available: https://www.dji.com/

[40] "Orbbec – Official Website," Orbbec, Accessed: Jul. 31, 2026. [Online]. Available: https://www.orbbec.com/

[41] "RoboSense – Official Website," RoboSense, Accessed: Jul. 31, 2026. [Online]. Available: https://www.robosense.ai/en

[42] "CATL – Official Website," CATL, Accessed: Jul. 31, 2026. [Online]. Available: https://www.catl.com/en/

[43] "Samsung SDI – Official Website," Samsung SDI, Accessed: Jul. 31, 2026. [Online]. Available: https://www.samsungsdi.com/

[44] "Panasonic – Official Website," Panasonic, Accessed: Jul. 31, 2026. [Online]. Available: https://www.panasonic.com/

[45] "Unitree H1 Humanoid Robot," RoboWorks, Accessed: Jul. 31, 2026. [Online]. Available: https://www.roboworks.net/store/p/store/?itemId=unitree-h1-humanoid-robot

[46] "Unitree G1," Unitree Robotics, Accessed: Jul. 31, 2026. [Online]. Available: https://www.unitree.com/g1

[47] "AgiBot A2," AgiBot, Accessed: Jul. 31, 2026. [Online]. Available: https://www.agibot.com/products/A2

[48] "Fourier GR-2," Fourier Intelligence, Accessed: Jul. 31, 2026. [Online]. Available: https://www.fftai.com/products-gr2

[49] "Sanctuary AI – Phoenix," Sanctuary AI, Accessed: Jul. 31, 2026. [Online]. Available: https://www.sanctuary.ai/

[50] "Boston Dynamics – Atlas," Boston Dynamics, Accessed: Jul. 31, 2026. [Online]. Available: https://www.bostondynamics.com/

[51] "Figure AI Unveils Figure 03," TIME, Accessed: Jul. 31, 2026. [Online]. Available: https://time.com/7324233/figure-03-robot-humanoid-reveal/

[52] "Agility Robotics – Digit," Agility Robotics, Accessed: Jul. 31, 2026. [Online]. Available: https://www.agilityrobotics.com/solution

[53] "Apptronik – Apollo," Apptronik, Accessed: Jul. 31, 2026. [Online]. Available: https://apptronik.com/apollo

[54] "1X – Neo," 1X Technologies, Accessed: Jul. 31, 2026. [Online]. Available: https://www.1x.tech/order

[55] "Tesla Optimus," Robots Guide, Accessed: Jul. 31, 2026. [Online]. Available: https://robotsguide.com/robots/optimus

[56] M. Ikuabe, C. Aigbavboa, and E. Kissi, "Potential applications and benefits of humanoids in the construction industry: A South African perspective," *International Journal of Building Pathology and Adaptation*, vol. 41, no. 6, pp. 254–268, Dec. 18, 2023, ISSN: 2398-4708. DOI: 10.1108/IJBPA-04-2023-0042 Accessed: Jan. 6, 2026. [Online]. Available: http://www.emerald.com/ijbpa/article/41/6/254-268/110174

[57] "Humanoid robots: From concept to reality | McKinsey," Accessed: Jan. 11, 2026. [Online]. Available: https://www.mckinsey.com/industries/industrials/our-insights/humanoid-robots-crossing-the-chasm-from-concept-to-commercial-reality

[58] "A Pathway Study For Future Humanoid Standards,"

[59] Y. Sun et al. "SPARK: A Modular Benchmark for Humanoid Robot Safety." arXiv: 2502.03132 [cs], Accessed: Jan. 11, 2026. [Online]. Available: http://arxiv.org/abs/2502.03132 pre-published.

[60] N. Scianca, P. Ferrari, D. De Simone, L. Lanari, and G. Oriolo, "A behavior-based framework for safe deployment of humanoid robots," *Autonomous Robots*, vol. 45, no. 4, pp. 435–456, May 2021, ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-021-09978-5 Accessed: Jan. 11, 2026. [Online]. Available: https://link.springer.com/10.1007/s10514-021-09978-5

[61] Z. Wang et al. "End-to-End Humanoid Robot Safe and Comfortable Locomotion Policy." arXiv: 2508.07611 [cs], Accessed: Jan. 11, 2026. [Online]. Available: http://arxiv.org/abs/2508.07611 pre-published.

[62] Hanley Importadora e Exportadora Ltda., *Manual de instruções piso laminado hanley*, Versão 16, Hanley, Porto Alegre, Brasil, 2023.

[63] Corkart SA, *Installation and maintenance instructions: Corkart floating floor*, Corkart, Vendas Novas, Portugal, 2022.

[64] ADO Floor, *Manual de instalação de pavimento composto spc rígido*, ADO Floor, Antalya, Turkey, 2021.

[65] Geometrik Target Flooring Solutions, *Manual de instalação e manutenção pavimento vinílico click*, Acedido em jan. 2026, Geometrik, Ovar, Portugal, 2022. [Online]. Available: https://geometriktarget.com/manual/instalacao-manutencao-vinilico-click

[66] "Gazebo," Accessed: Jan. 8, 2026. [Online]. Available: https://gazebosim.org/home

[67] "Isaac Sim," NVIDIA Developer, Accessed: Jan. 8, 2026. [Online]. Available: https://developer.nvidia.com/isaac/sim

[68] M. Zwingel, C. May, M. Kalenberg, and J. Franke, "Robotics simulation – A comparison of two state-of-the-art solutions," in *ASIM SST 2022 Proceedings Langbeiträge*, ARGESIM Publisher Vienna, 2022, pp. 171–178, ISBN: 978-3-901608-97-1. DOI: 10.11128/arep.20.a2033 Accessed: Jan. 5, 2026. [Online]. Available: https://www.asim-gi.org/a2033

[69] G. Gupta. "Isaac Sim: Photorealistic Rendering for Next-Gen Robot Development," Black Coffee Robotics, Accessed: Jan. 5, 2026. [Online]. Available: https://medium.com/black-coffee-robotics/isaac-sim-photorealistic-rendering-for-next-gen-robot-development-37ca8186c1d9

[70] "ROS 2 Documentation — ROS 2 Documentation: Humble documentation," Accessed: Jan. 9, 2026. [Online]. Available: https://docs.ros.org/en/humble/index.html

[71] "Universal humanoid robot H1_Bipedal Robot_Humanoid Intelligent Robot Company | Unitree Robotics," Accessed: Jan. 5, 2026. [Online]. Available: https://www.unitree.com/h1

[72] "Document," Accessed: Jan. 8, 2026. [Online]. Available: https://marketing.unitree.com/article/en/H1/User_Manual.html

[73] QUADRUPED Robotics. "H1 overview | unitree h1." Accessed: 2026-01-12. [Online]. Available: https://www.docs.quadruped.de/projects/h1/html/h1_overview.html

[74] NumPy Developers. "Numpy," NumPy, Accessed: Jul. 31, 2026. [Online]. Available: https://numpy.org/