FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Simulation and Control of Humanoid Robots for Box Handling in Logistics Environments

Luís Manuel Gaspar Leitão

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Doutor António Paulo Moreira

Second Supervisor: Prof. Doutor António Fernando Ribeiro

July 21, 2026

© Luís Leitão, 2026

Resumo

Robôs humanoides combinam locomoção bípede com manipulação com dois braços numa única plataforma, permitindo-lhes operar em ambientes concebidos para trabalhadores humanos sem necessidade de alterações estruturais significativas. No entanto, a sua implementação enfrenta desafios específicos que não surgem em plataformas robóticas convencionais: a locomoção bípede induz vibrações contínuas que afetam a estabilidade dos sensores, uma câmara montada numa cabeça móvel introduz incerteza contínua na pose do sensor e as restrições de tamanho, peso e consumo energético próprias destas plataformas limitam os recursos computacionais disponíveis a bordo. Estes fatores dificultam a obtenção da precisão e da capacidade de resposta em tempo real exigidas por tarefas autónomas de manipulação.

Esta dissertação propõe uma arquitetura integrada de perceção, localização, navegação e manipulação para o robô humanoide Booster T1, destinada à execução de tarefas de recolha e colocação de caixas num ambiente interior estruturado. A pose planar é estimada continuamente através de um Filtro de Kalman Estendido, que combina *dead-reckoning* com observações de *landmarks* semânticos já existentes no ambiente, detetados por uma rede neuronal convolucional, e de marcadores fiduciais ArUco colocados seletivamente em locais onde é necessária elevada precisão, como as estações de recolha. A arquitetura de software encontra-se distribuída por duas placas de processamento embarcadas, que comunicam através do *Robot Operating System 2* (ROS 2), separando as tarefas de perceção e localização das de controlo do movimento e manipulação. Um orquestrador baseado numa máquina de estados finitos coordena as diferentes fases de navegação, alinhamento visual e captura da caixa com ambos os braços, completando o ciclo de recolha.

O sistema foi validado experimentalmente no robô físico, num ambiente laboratorial. O desempenho da localização foi avaliado com um sistema de captura de movimento como referência, demonstrando uma redução significativa do erro de pose relativamente à utilização exclusiva de odometria. Estudos de simulação de Monte Carlo, desenvolvidos em MATLAB, permitiram caracterizar a sensibilidade do filtro à densidade de *landmarks* e ao ruído das medições. A arquitetura integrada foi ainda avaliada através de ensaios repetidos de recolha de caixas, medindo a taxa de sucesso do ciclo completo de manipulação. Adicionalmente, a adaptação do sistema a um campo de futebol da RoboCup, durante o Festival Nacional de Robótica, demonstrou a portabilidade da solução para novos ambientes.

Palavras-chave: Robótica Humanoide, Localização Visual, Filtro de Kalman Estendido, Deteção de Landmarks Semânticos, Manipulação, ROS 2.

Abstract

Humanoid robots combine bipedal locomotion with dual-arm manipulation in a single platform, enabling them to operate in spaces designed for human workers without significant structural modification. However, their effective deployment faces specific challenges that do not arise on conventional robotic platforms: bipedal locomotion induces continuous vibration that affects sensor stability, a camera mounted on a movable head introduces ongoing uncertainty in sensor pose, and the strict size, weight, and power budgets of humanoid platforms limit the computational resources available onboard. These factors introduce significant challenges for achieving the precision and real-time responsiveness required for autonomous manipulation tasks.

This dissertation proposes an integrated perception, localization, navigation, and manipulation pipeline for the Booster T1 humanoid robot to perform box pick-and-place tasks in a structured indoor environment. An Extended Kalman Filter continuously estimates the robot's planar pose, fusing proprioceptive dead reckoning with observations of semantic landmarks already existing in the environment, detected by a convolutional neural network, and of ArUco fiducial markers selectively placed in high-precision dependent locations such as picking stations. The software architecture is distributed across two onboard processing boards communicating via *Robot Operating System 2* (ROS 2), separating perception and localization from motion control and manipulation. A finite-state-machine orchestrator sequences the phases of navigation, visual alignment, and dual-arm grasping, completing the picking cycle.

The system was experimentally validated on the physical robot in a laboratory environment. Localization was evaluated against motion capture ground truth, demonstrating a substantial reduction in pose error compared to pure dead reckoning. Monte Carlo simulation studies in MATLAB characterized the filter's sensitivity to landmark density and measurement noise. The integrated pipeline was assessed through repeated picking trials, measuring the success rate of the full manipulation cycle. Additionally, the adaptation of the system to a RoboCup soccer field during the Robotics National Festival validated the portability of the framework to new environments.

Keywords: Humanoid Robotics, Visual Localization, Extended Kalman Filter, Semantic Landmark Detection, Dual-Arm Manipulation, ROS 2.

Humanoid Robotics,Visual Localization,Extended Kalman Filter,Semantic Landmark Detection,Dual-Arm Manipulation,ROS2

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals[1] to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation contributes to sustainable industrialization by developing an autonomous perception-driven pipeline for humanoid robots in logistics environments. The system enables flexible automation of box handling tasks using exclusively onboard sensing, eliminating the need for external localization infrastructure and reducing the requirement for extensive facility modifications, promoting resource-efficient production processes and extending the functional lifespan of existing warehouse infrastructure. The passive lifting strategy employed by the custom end-effectors reduces actuator energy demand compared to conventional force-closing grasp approaches, contributing to energy-efficient robotic operations. Furthermore, the reusable, modular software architecture demonstrates how intelligent robotic systems can be deployed with reduced development waste and minimal environmental footprint. By enabling humanoid robots to operate alongside human workers in spaces originally designed for people, the work also supports inclusive and sustainable economic growth, advancing collaborative automation rather than wholesale workforce replacement.

The specific Sustainable Development Goals mentioned have the following names:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 12 Ensure sustainable consumption and production patterns

[1] https://sdgs.un.org/goals

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.2	The proposed system enhances logistic productivity through autonomous robotic handling, reducing reliance on repetitive manual labor and promoting economic growth via technological innovation.	Picking task success rate of 80%; localization error of 14.4 cm with the Extended Kalman Filter (EKF).
	8.4	The modular architecture and exclusive use of onboard sensing avoid costly infrastructure modifications, promoting efficient use of resources in existing warehouse facilities.	No structural modifications required, only the addition of ArUco markers in areas with high-precision requirements
9	9.1	The distributed two-board ROS 2 architecture and platform-agnostic software design provide a reusable framework for deploying humanoid robots in logistics infrastructure.	Validated across two distinct environments (laboratory and RoboCup soccer field); 23-Degrees-Of-Freedom (DoF) platform integration with generic interfaces.
	9.5	The novelty of this work lies in combining an EKF with a Convolutional Neural Network(CNN)-based semantic landmark detector for localization on a humanoid robot, where the head-mounted camera undergoes continuous viewpoint changes induced by bipedal locomotion. The filter incorporates a delayed-update mechanism with state rollback and replay to compensate for neural inference latency.	Real-time EKF localization at 30 Hz fusing delayed CNN observations (91 ms average latency) with proprioceptive dead reckoning on embedded hardware; further analyzed through MATLAB simulation and OptiTrack ground truth comparison.
12	12.2	Passive L-shaped end-effectors eliminate the need for sustained gripping force, reducing mechanical wear and energy consumption during box transport.	Passive grasping strategy requires no active power during hold; custom end-effectors reduce actuator load.
	12.5	The perception pipeline serves dual purposes, localization and object detection, using a single RGB-D camera, maximizing sensor utility and minimizing computational redundancy.	Single camera handles perception for both localization and manipulation tasks; distributed board architecture optimizes computational resources.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. António Paulo Moreira and Prof. António Fernando Ribeiro, for their invaluable guidance and availability throughout the development of this dissertation. Their expertise and constructive feedback were fundamental to the successful completion of this work. I am also grateful for the times they went out of their way to help this project.

My gratitude also extends to my senior members of the laboratory, whose guidance and willingness to help whenever I needed it made a real difference throughout this process.

I am equally grateful to my colleagues at the laboratory, whose brainstorming sessions, shared late hours, and lunch conversations made the daily work not only easier but genuinely enjoyable. Their willingness to help and exchange ideas contributed significantly to this work.

I would also like to thank my family for their unconditional support, encouragement, and belief in me throughout my academic journey. Their motivation has always been one of the main driving forces behind my achievements. This work is as much theirs as it is mine.

Luís Leitão

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
  - 1.1.1 Limitations of Fixed Automation . . . 2
  - 1.1.2 Humanoid Robots as a Flexible Solution . . . 2
  - 1.1.3 Industrial Adoption and Recent Deployments . . . 4
- 1.2 Core Technical Challenges . . . 5
  - 1.2.1 Perception and Localization . . . 5
  - 1.2.2 Bipedal Locomotion and Balance Control . . . 6
  - 1.2.3 Manipulation and Object Interaction . . . 6
  - 1.2.4 Computational Constraints . . . 7
  - 1.2.5 Human-Robot Interaction . . . 7
- 1.3 Scope of Work . . . 7
- 1.4 Document Structure . . . 8

2 Background and State of the Art **11**
- 2.1 Vision-Based Localization for Humanoid Robots . . . 11
  - 2.1.1 Simultaneous Localization and Mapping on Humanoid Robots . . . 11
  - 2.1.2 Landmark-Based Localization with the Extended Kalman Filter . . . 13
  - 2.1.3 EKF Formulation and Implementation on Humanoid Platforms . . . 16
  - 2.1.4 Data Association . . . 17
  - 2.1.5 Temporal Consistency and Delayed Observations . . . 17
  - 2.1.6 Comparative Analysis of Localization Methods . . . 18
- 2.2 Object Perception for Manipulation . . . 19
  - 2.2.1 CNN-Based Object Detection . . . 19
  - 2.2.2 Three-Dimensional Position Estimation from RGB-D . . . 20
  - 2.2.3 Six-Degree-of-Freedom Pose Estimation . . . 21
- 2.3 Navigation and Path Tracking . . . 21
  - 2.3.1 Path Planning . . . 22
  - 2.3.2 Path Tracking over Predefined Waypoints . . . 23
- 2.4 Task-Level Capabilities in Logistics Environments . . . 23
  - 2.4.1 Grasping in Logistics Scenarios . . . 24
  - 2.4.2 Vision-Based Quality Inspection . . . 24
  - 2.4.3 Collision Avoidance with Humans and Mobile Robots . . . 24
- 2.5 Robotic Simulation Platforms . . . 25

3 System Architecture and Proposed Approach **27**
3.1 Robotic Platform and Hardware . . . 27
3.2 Software Tools and Environment . . . 32
3.3 Overall System Architecture . . . 33
3.4 ROS 2 Communication Layer . . . 34
3.5 End-to-End Mission Pipeline . . . 36
3.6 Picking State Machine . . . 37
3.7 Simulation Environment . . . 38

4 Visual Localization **41**
4.1 Localization Problem Formulation . . . 41
4.2 Semantic Map of the Environment . . . 43
4.3 Feature Detection for Localization . . . 44
4.4 Observation Generation from RGB-D Data . . . 47
4.5 Extended Kalman Filter Formulation . . . 48
4.5.1 Motion Model and Prediction . . . 49
4.5.2 Dynamic Camera Pose and Measurement Model . . . 50
4.5.3 EKF Update and Validation . . . 51
4.6 Data Association and Update Validation . . . 51
4.7 Temporal Robustness and Real-Time Architecture . . . 54
4.7.1 System Architecture and Real-Time Performance . . . 54
4.7.2 Implementation Details . . . 54
4.7.3 Temporal Misalignment and Delayed Updates . . . 56
4.8 ArUco Marker Localization . . . 56
4.9 Experimental Results . . . 57
4.9.1 Real-World Evaluation with OptiTrack . . . 58
4.9.2 MATLAB Simulation Study . . . 59
4.10 Adaptation to a New Environment: Robotics National Festival 2026 . . . 65

5 Navigation, Perception and Manipulation **67**
5.1 Navigation and Path Tracking . . . 67
5.2 Visual Object Detection . . . 71
5.3 Visual Alignment and Picking Approach . . . 72
5.4 Grasping and Manipulation . . . 75
5.5 Mission Execution and Orchestration . . . 77
5.6 Picking Experimental Evaluation . . . 78

6 Conclusions **81**
6.1 Summary and Retrospective . . . 81
6.2 Main Contributions . . . 82
6.3 Discussion of Results . . . 83
6.3.1 Localization Performance . . . 83
6.3.2 Insights from MATLAB Simulations . . . 84
6.3.3 Picking Performance . . . 85
6.3.4 Adaptability to New Environments . . . 85
6.4 Limitations . . . 86
6.5 Future Work . . . 87

References **89**

7 Supplementary Material and Visual Documentation **93**
7.1 Simulation Validation (Isaac Sim) 93
7.2 Localization and Navigation 93
7.3 Robotics National Festival 2026 93
7.4 Picking Trials 94
7.5 Complete End-to-End Mission 94

List of Figures

1.1 Examples of humanoids executing locomotion and manipulation tasks. 3
1.2 Digit robot in a warehouse environment. 4

2.1 RGB-D SLAM reconstructed 3D map from a humanoid platform. 12
2.2 Examples of fiducial markers. 13
2.3 Pose estimation through image correspondences using SuperGlue. 14
2.4 CNN-detected semantic landmarks in the target laboratory environment. 15
2.5 YOLO-based detection of the target box. 20
2.6 Paths planned by traditional A* (left) and an improved variant (right). 22
2.7 Exploration using RRT* in a continuous space. 22
2.8 Isaac Sim simulation environment. 25

3.1 Booster T1 humanoid robot. 28
3.2 Orbbec Gemini 2 335L camera mounted on the robot head. 29
3.3 Custom L-shaped end-effector extensions. 29
3.4 Kinematic structure of the Booster T1 humanoid robot. 31
3.5 Onboard computing architecture and communication via ROS 2 topics over the onboard Ethernet network. 35
3.6 Task-level command state machine. 36
3.7 Path tracking in simulation. 38

4.1 Mapped environment including both semantic landmarks ArUco markers. 44
4.2 Example of the detected semantic landmarks used for robot localization. 45
4.3 Data association pipeline, for both natural detected landmarks and ArUco markers. 53
4.4 ArUco marker deployment. 57
4.5 Comparison of EKF, ground-truth, and dead-reckoning trajectories. 58
4.6 Effect of landmark density on EKF localization performance. 60
4.7 Effect of Q/R ratio on EKF convergence behavior. 62
4.8 Effect of fiducial marker noise on EKF precision. 64
4.9 YOLO landmark detection on the RoboCup soccer field. 65

5.1 Finite-state machine for path tracking. 68
5.2 Target boxes used in the pick-and-place experiments. 71
5.3 YOLO box detection when multiple candidates are available. 72
5.4 Two-phase picking strategy: lateral alignment, followed by forward approach. . . 73
5.5 Dual-arm grasp sequence positions. 75
5.6 End-effector extensions positioned beneath the box edges. 76

List of Tables

2.1 Comparison of vision-based localization methods for humanoid robots. 19

3.1 Software tools and frameworks used in the system 32
3.2 Module distribution across the two onboard boards 33
3.3 Key ROS topics in the mission pipeline 35

4.1 YOLOv8 detector comparison 46
4.2 Real-world evaluation results (mean $\pm$ std over ten runs) 58

5.1 Path tracker limits and parameters 70
5.2 Picking-stage parameters 74

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
A*	A-Star (search algorithm)
AMR	Autonomous Mobile Robot
ArUco	Fiducial Marker System (ArUco)
AS/RS	Automated Storage and Retrieval System
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DoF	Degrees-of-Freedom
EKF	Extended Kalman Filter
FOV	Field of View
GPU	Graphics Processing Unit
ICNN	Individual Compatibility Nearest Neighbor
IMU	Inertial Measurement Unit
InEKF	Invariant Extended Kalman Filter
JCBB	Joint Compatibility Branch and Bound
LiDAR	Light Detection and Ranging
mAP	mean Average Precision
NIS	Normalized Innovation Squared
ORB	Oriented FAST and Rotated BRIEF
PnP	Perspective-n-Point
PRM	Probabilistic Roadmap
RGB-D	Red Green Blue - Depth
RMSE	Root Mean Square Error
ROS2	Robot Operating System 2
RRT	Rapidly-exploring Random Tree
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded-Up Robust Features
UKF	Unscented Kalman Filter
YOLO	You Only Look Once
TOML	Tom's Obvious, Minimal Language
RTAB-Map	Real-Time Appearance-Based Mapping
JSON	JavaScript Object Notation
YAML	Yet Another Markup Language

Chapter 1

Introduction

The objective of this work is to design, implement, and validate a perception-driven pipeline for a humanoid robot performing box manipulation tasks in a structured logistics-inspired environment. The experimental validation focuses on box pick-and-place operations from fixed support structures in a warehouse setting, using the T1 humanoid robot from Booster Robotics[1] as the reference platform. The proposed pipeline includes vision-based localization using semantic landmarks and fiducial markers, three-dimensional object position estimation from Red, Green, Blue-Depth (RGB-D) data, path tracking over predefined waypoints, visual alignment, dual-arm manipulation, and mission orchestration, integrated into a distributed two-board architecture. The system relies exclusively on the robot's onboard RGB-D camera as the exteroceptive sensor, avoiding dependence on external instrumentation and extensive infrastructure modification.

This chapter contextualizes the work within the broader challenges facing warehouse automation, identifies the core technical problems that humanoid robots must address to operate effectively in logistics environments, defines the scope and objectives of this dissertation, and outlines the structure of the document.

1.1 Context and Motivation

The logistics sector faces increasing pressure to handle greater product variety, shorter delivery cycles, and fluctuating demand, driven largely by the growth of e-commerce [1, 2]. Fixed automation systems such as conveyor belts, automated storage and retrieval systems (AS/RS), and high-throughput industrial manipulators are well-established for high-volume, low-variety operations [3, 4]. However, their effectiveness depends on stable product portfolios and dedicated infrastructure; they are not easily reconfigured when layouts, product types, or throughput requirements change [1]. In parallel, many logistics facilities were originally built for human operators and still contain a substantial share of tasks that resist full fixed automation, either because the task variety demands cognitive flexibility or because the cost of retrofitting the facility is prohibitive [2]. These conditions create a persistent gap between what fully automated systems can handle and what still

[1] https://www.booster.tech/

relies on human labor. Humanoid robots, which combine bipedal mobility with dual-arm manipulation and integrated exteroceptor sensing in a single platform, have been proposed as a means of addressing this gap: they are designed to operate in spaces built for people, adapt to changing task configurations, and perform physical work without requiring extensive infrastructure modification [5].

1.1.1 Limitations of Fixed Automation

Traditional warehouse automation relies on fixed systems such as conveyor belts, automated storage and retrieval systems (AS/RS), and industrial manipulators [3]. These were engineered for stable, flow-line production scenarios like assembly lines, where product specifications and sequences remain predetermined [3, 4].

Logistics warehouses, however, operate under fundamentally different conditions. Order composition, product variety, and facility layouts change frequently in response to fluctuating demand and short product life cycles [1]. Fixed automation systems are inherently rigid by design: this rigidity is what enables high throughput and repeatability in stable production scenarios, but it makes reconfiguration expensive and time-consuming[2, 4]. This inflexibility is particularly problematic in dynamic picking and handling tasks, where boxes vary in size and placement, and where robots must operate alongside humans and Autonomous Mobile Robots (AMRs) [1]. Traditional automation is designed around the principle that the environment should be structured to suit the robotic system, which is economically effective for stable production but becomes a limitation when flexibility is required [6].

In existing warehouse facilities, the physical modifications required to integrate fixed automation, such as reinstalling conveyor systems, reconfiguring shelving, and redesigning transport routes, can be impractical or economically unjustifiable, particularly for facilities handling diverse product types or facing uncertain demand profiles [1]. This constraint leaves many warehouses reliant on manual labor or semi-automated solutions that provide limited operational flexibility.

These limitations have motivated the search for robotic platforms that can operate flexibly in human-centered environments. Within this context, humanoid robots have emerged as a promising option [5].

1.1.2 Humanoid Robots as a Flexible Solution

As shown in Figure 1.1, humanoid robots are robotic systems that combine bipedal locomotion with dual-arm manipulation in a single integrated platform, enabling them to work effectively in spaces originally designed for human operators [5, 7]. Unlike wheeled platforms, humanoids can navigate across uneven terrain and access areas constrained by stairs or narrow passages, which are common in real warehouse facilities [5]. This mobility allows humanoids to operate across multiple zones of a warehouse using the same pathways as human operators, without requiring extensive infrastructure modifications[8].

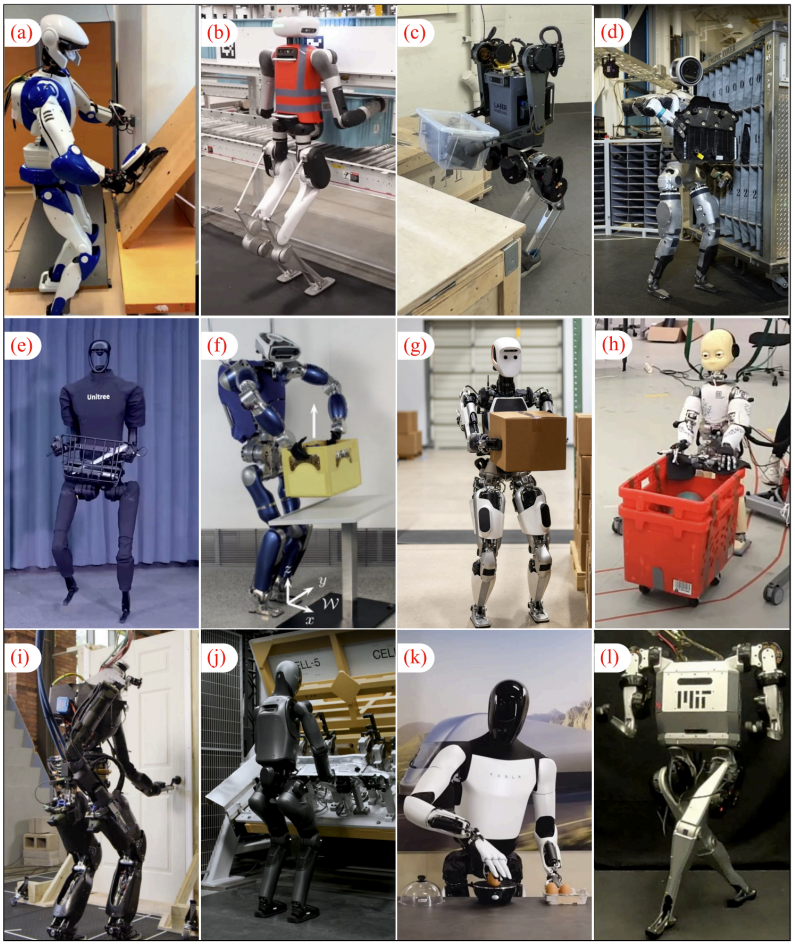

Figure 1.1: Examples of humanoids executing locomotion and manipulation tasks [8].

Beyond mobility, humanoid robots offer advanced manipulation capabilities through articulated upper limbs with multiple Degrees-of-Freedom (DoF). These enable them to perform human-like tasks such as grasping, lifting, and placing objects at varying heights and orientations. Unlike fixed industrial manipulators that are confined to a single workstation or a limited workspace, humanoids can reposition themselves while manipulating, allowing them to work with a higher range of movement. This combination of locomotion and dexterous manipulation, achievable by a single platform without structural reconfiguration, is particularly relevant for logistics scenarios involving shelves, racks, and mobile platforms [7].

Humanoid robots also typically have a movable head mounted with vision sensors, which provides an adjustable perception perspective that enables localization, detection, and interaction with objects in the environment [8].

When compared to other robotic solutions, these capabilities show why humanoid robots are useful in logistics. Standard AMRs excel at efficient material transport on flat surfaces, but most have a hard time moving in uneven terrain and lack the dexterity and reach required for direct manipulation of objects [2]. Mobile robots that mount robotic arms on top of wheeled bases partially overcome this limitation, but remain constrained to planar workspaces. Industrial manipulators

provide high precision and payload capacity, but they are confined to fixed workspaces and require structured environments [2]. Humanoid robots bridge this gap by combining mobility and manipulation, enabling adaptability in logistics scenarios [5].

The potential for intuitive human-robot collaboration is another important advantage of humanoid platforms. While collaborative robots, typically industrial manipulators, are explicitly designed to share workspaces with human operators through force-limiting and speed-limiting safety features, their interaction with humans remains largely constrained to their fixed operational envelope [6]. Humanoid robots, by contrast, leverage their anthropomorphic appearance and motion patterns to foster more natural and intuitive interaction. Their body structure and movement dynamics can improve predictability and safety in shared workspaces, reducing the cognitive load on human operators who must interpret and anticipate robot behavior [6]. This advantage extends the potential for human-robot collaboration beyond safety compliance to genuine collaborative task execution. In logistics environments, where robots and humans frequently operate in close proximity and share tasks such as picking, sorting, and material handling, the ability of humanoids to communicate intent through natural body language and spatial awareness capabilities that are difficult to achieve with traditional robotic arms becomes increasingly valuable [6].

1.1.3 Industrial Adoption and Recent Deployments

Recent years have witnessed the first commercial deployments of humanoid robots in logistics and manufacturing environments, signaling early-stage interest from the industry in transitioning humanoid platforms beyond research environments. GXO Logistics became the first company to deploy humanoid robots commercially in a warehouse setting, using the Digit robot[2], shown in Figure 1.2, from Agility Robotics, to lift and carry tote bins between conveyor belts [9]. This milestone demonstrated that, at least in a controlled warehouse setting with repetitive tasks, the Digit robot can perform material handling alongside human workers, performing repetitive material handling tasks.

Figure 1.2: Digit robot in a warehouse environment.

Several companies have since advanced beyond pilot testing. Figure AI delivered its Figure 02 humanoid for logistics triage operations on moving conveyor systems and has established manufacturing capacity for large-scale production [10]. Tesla has deployed its Optimus humanoid internally for factory material handling and transport tasks, with commercial availability expected in the near term [11]. These deployments, while concentrated in semi-structured environments with repetitive tasks, suggest the emerging feasibility of humanoid robots for limited logistics applications and provide valuable operational insights.

Early operational results from these commercial pilots provide preliminary evidence that humanoid robots can achieve acceptable reliability in repetitive material handling tasks, although these deployments remain limited in scope and scale. Agility Robotics reported that Digit has moved over 100,000 totes at GXO's facility, demonstrating reliability across thousands of repetitive cycles under real warehouse conditions [9]. Similarly, Figure AI's pilot operations handled approximately 90,000 sheet-metal components, achieving manipulation speeds comparable to human performance [10]. Beyond throughput, deployment efficiency has improved significantly, with setup times reduced from weeks to days due to advances in AI integration. These milestones suggest that humanoid platforms are starting to transition from experimental prototypes to initial practical solutions.

1.2 Core Technical Challenges

Despite their potential, several fundamental challenges limit the deployment of humanoid robots in logistics environments. These challenges span perception, locomotion, manipulation, human-robot interaction, and system integration, each imposing distinct requirements on the design and operation of humanoid platforms.

1.2.1 Perception and Localization

Perception and localization are critical capabilities for autonomous operation in logistics scenarios, enabling a robot to interpret its surroundings and maintain an estimate of its own state within that environment [12]. Localization refers to the process of estimating the robot's pose (position and orientation) relative to a map, while perception provides the semantic and geometric information about objects, obstacles, and work areas that the robot must interact with [12]. To navigate and interact effectively, humanoid robots must continuously estimate their own position in relation to the environment; without localization, a robot cannot move reliably, manipulate objects, or perform any meaningful autonomous task [12, 13].

The challenge is that the camera mounted on a movable head may introduce continuous uncertainty. As the robot walks, manipulates objects, or moves its head, the camera's position and orientation change constantly. Accurate localization requires precise knowledge of the robot's kinematics and careful handling of the relationships between different coordinate frames using

[2] https://www.youtube.com/watch?v=tMIZlzX8HTI

homogeneous transformation matrices [7]. This geometric complexity adds substantial difficulty compared to wheeled robots with fixed sensor configurations, where sensor pose relative to the robot base remains constant [7]. An additional challenge arises from the computational demands of visual perception. Modern neural-network-based detectors can extract rich semantic information from camera images, but their inference time introduces a non-negligible delay between image acquisition and the moment the resulting observations are available to the localization filter. By the time a visual measurement is ready, the robot has already moved, and directly applying a stale observation to the current pose estimate introduces temporal inconsistency. Addressing this requires mechanisms that either compensate for the delay or apply corrections at the correct past time instant. The specific approaches to address these challenges are reviewed in Section 2.1 and further developed in Chapter 4.

1.2.2 Bipedal Locomotion and Balance Control

Achieving robust bipedal locomotion under industrial conditions remains a primary challenge for humanoid robots. Logistics facilities present uneven floors, dynamic obstacles, and frequent interactions with humans and mobile robots, which impose rigid requirements on balance control and motion stability [8, 14]. While the details of humanoid-specific motion planning and whole-body control fall outside the scope of this work, as the T1 platform provides dedicated balance modules, the requirements they impose on perception, localization, and path planning are critical.

Long-term operation poses additional concerns, as humanoid robots are mechanically complex systems with many DoF, which increases the likelihood of component wear, calibration drift, and failure over extended periods [8, 14].

1.2.3 Manipulation and Object Interaction

Humanoid manipulation in logistics environments requires reliable detection and three-dimensional localization of target objects relative to the robot [8]. Unlike industrial manipulators operating in fixed, structured workspaces, humanoids must contend with variable object positions, constraints imposed by bipedal balance, and the limited field of view of head-mounted sensors.

Detecting and spatially locating box-shaped objects is particularly challenging in warehouse settings, where boxes may lack rich visual texture and frequently appear partially hidden by other items or infrastructure [8]. In scenarios where boxes are placed on standardized support structures with predictable orientation, the estimation problem can be simplified to recovering the three-dimensional position of the object rather than its full 6-DoF pose. However, depth estimation from RGB-D sensors introduces its own sources of uncertainty, particularly at close range or on textureless surfaces, which must be accounted for in the manipulation pipeline. Grasping strategies must also be compatible with the robot's end-effector design and payload limitations, trade-offs that are particularly acute on compact humanoid platforms with limited arm actuator strength.

1.2.4 Computational Constraints

Vision-based perception, localization, and object detection pipelines rely on computationally intensive algorithms, including deep neural networks and probabilistic state estimation methods. Ensuring real-time performance on onboard hardware while maintaining robustness and accuracy remains a significant challenge [5, 8]. Humanoid platforms have strict limits on computational capacity due to size, weight, and power constraints, necessitating efficient algorithms and careful system-level optimization. One practical strategy to address these constraints is to distribute the computational load across dedicated processing boards, for example, offloading Graphics Processing Unit (GPU)-accelerated neural inference to a dedicated vision processor while reserving a separate board for low-latency motion control, but this introduces inter-board communication requirements that must be managed without compromising real-time performance.

1.2.5 Human-Robot Interaction

Safe operation in shared warehouse environments where humans and robots work in close proximity requires robust perception of human presence and responsive safety mechanisms [6]. While human-robot interaction is a critical concern for real-world logistics deployment, it falls outside the scope of this work, which operates under the assumption of a static environment without dynamic obstacles or human co-workers.

1.3 Scope of Work

This dissertation develops and integrates a perception-driven pipeline for the Booster T1 humanoid robot to perform box pick-and-place tasks in a structured indoor environment. The system relies exclusively on the robot's onboard RGB-D camera as the exteroceptive sensor, and the work is intentionally scoped to address the perception, localization, and task-sequencing challenges that enable autonomous operation, while delegating lower-level locomotion and balance control to the platform's existing modules.

The primary focus of this work is visual localization. The robot must continuously estimate its planar pose within a known semantic map of the environment. The localization solution fuses two complementary sources of visual information within an Extended Kalman Filter (EKF): semantic landmarks detected by a You Only Look Once (YOLO)-based neural network, and fiducial ArUco markers placed in precision-critical zones. The EKF combines visual updates from the RGB-D camera with proprioceptive dead reckoning from the robot's motion controller. To address the temporal misalignment caused by neural inference latency, the filter employs a delayed-update mechanism with state rollback and replay, ensuring that observations are applied at the correct past time instant. The localization pipeline is validated against an OptiTrack motion capture system on the real robot, and its intrinsic behavior is further characterized through controlled Monte Carlo simulations in MATLAB.

The secondary focus is object detection and three-dimensional position estimation. A second YOLO-based detector identifies the target box in the camera stream and estimates its 3D position by back-projecting depth data into the robot trunk frame. The box orientation is assumed to be known *a priori*, as the target object is placed on a standardized support structure parallel to the robot's approach direction. This simplification reduces the estimation problem from full 6-DoF pose recovery to robust depth-based position estimation, a pragmatic choice given the textureless nature of the target box and the limited field of view of the head-mounted camera at close range.

Path tracking over predefined waypoints is employed for navigation between workstations. Rather than computing paths online, the system follows static waypoint sequences loaded from configuration files, using the EKF pose estimate as feedback. This approach is sufficient for the laboratory environment, where the positions of picking stations are known in advance. A dedicated visual alignment module uses the lateral component of the detected box position to centre the robot relative to the target before the forward approach, decoupling lateral correction from depth-axis motion and simplifying the control problem.

Dual-arm manipulation is implemented as a predefined grasp sequence: reach, close, up, and home, executed via the robot's Software Development Kit (SDK). The stock end-effectors are replaced by custom L-shaped extensions that slide under the box edges and lift it from below, avoiding the need for sustained gripping force. This passive lifting strategy is a practical response to the limited torque available from the compact arm actuators.

All modules are integrated under a mission orchestrator that implements a finite-state machine sequencing navigation, visual alignment, grasping, and retreat. The software architecture is distributed across two onboard processing boards: a Jetson board handles GPU-accelerated neural inference and EKF localization, while a dedicated motion board runs the path tracker, alignment controller, and arm executor with low latency and direct access to the robot SDK. The two boards communicate over the onboard Ethernet network via ROS 2 topics.

In summary, this dissertation covers the full pipeline from visual perception to physical manipulation, with the core research contributions concentrated in the localization and sensor-fusion components. The remaining modules: path tracking, visual alignment, grasping, and orchestration, are designed with practical simplicity to support the end-to-end demonstration of the system on the physical robot in the laboratory environment. The locomotion and path tracking modules were first validated in the Isaac Sim physics simulator before deployment on the real platform.

1.4 Document Structure

This document is organized as follows.

Chapter 2 reviews the state of the art in the technical areas central to this work. The review examines vision-based localization methods for humanoid robots. It then discusses object detection and three-dimensional position estimation from RGB-D data. Path-tracking and path-planning strategies for mobile robots are reviewed, followed by grasp-planning approaches suited to box-shaped objects. The chapter also covers task-level capabilities in logistics scenarios, including

grasping strategies and collision avoidance considerations, and briefly reviews robotic simulation platforms used in the development workflow.

Chapter 3 presents the robotic platform and the overall system architecture. It describes the Booster T1 hardware, the onboard computation split across two processing boards, the ROS 2 communication layer, and the end-to-end mission pipeline. The high-level orchestrator state machine that sequences the mission phases is detailed, together with the simulation environment used for locomotion validation and the operating assumptions that define the scope of the system.

Chapter 4 presents the visual localization system. It formulates the EKF for planar pose estimation, describes the semantic map of the environment, and details the YOLO-based landmark detector. The observation generation pipeline from RGB-D data, the data association strategy, and the delayed-update mechanism for temporal consistency are then presented. The complementary ArUco marker system is described, and the chapter concludes with the experimental evaluation against OptiTrack ground truth and the MATLAB simulation studies.

Chapter 5 describes the navigation, perception, and manipulation modules that execute the box-handling tasks. It covers the path tracking controller, the box detection and 3-dimensional (3D) position estimation pipeline, the visual alignment and picking approach strategy, and the dual-arm grasping module with the custom L-shaped end-effectors. The mission execution and orchestration layer is then presented, followed by the qualitative results of the end-to-end task demonstration in the laboratory environment.

Chapter 6 summarizes the main conclusions of the work, articulates the contributions of the developed system, discusses its limitations, and identifies directions for future research.

Chapter 2

Background and State of the Art

This chapter reviews the state of the art in the key technical areas required for deploying a humanoid robot in logistics environments: vision-based localization, object perception for manipulation, navigation and path tracking, and task-level capabilities. For each area, representative methods from the literature are presented and their trade-offs in terms of accuracy, computational efficiency, and applicability to humanoid platforms are discussed. The chapter also briefly covers robotic simulation platforms used in the development workflow.

2.1 Vision-Based Localization for Humanoid Robots

Estimating the robot's pose relative to a known map is a prerequisite for autonomous operation in structured indoor environments. A humanoid robot must continuously track its own position and orientation to plan motion, approach objects, and execute manipulation tasks [12, 13]. Localization in mobile robots can exploit a range of sensor modalities, including Light Detection and Ranging (LiDAR), radar, inertial measurement units (IMUs), and cameras. The Booster T1 platform used in this dissertation is equipped with an RGB-D camera as its sole exteroceptive sensor, making vision-based localization the natural choice [8]. Moreover, humanoid platforms introduce challenges absent from wheeled or fixed-base robots: the camera is mounted on a movable head, its viewpoint shifts continuously during walking, and the kinematic chain linking the camera to the ground introduces additional geometric complexity [7, 15]. This section therefore reviews vision-based localization methods with particular attention to their use on humanoid robots, covering Simultaneous Localization and Mapping (SLAM), landmark-based approaches, Convolutional Neural Network (CNN)-based landmark detection, the underlying EKF formulation, data association strategies, and temporal consistency mechanisms.

2.1.1 Simultaneous Localization and Mapping on Humanoid Robots

SLAM addresses autonomous navigation by building an environmental map while simultaneously estimating the robot's pose. The evolution of visual SLAM has been characterized by a shift from filter-based to graph-based formulations [16]. Modern feature-based systems such as Oriented

FAST and Rotated BRIEF (ORB)-SLAM3 achieve real-time performance with monocular, stereo, and RGB-D sensors, supporting visual-inertial fusion and loop closure on embedded platforms [17]. However, these systems are designed and evaluated primarily on wheeled robots or handheld sensors with smooth trajectories, and they do not explicitly account for the motion modalities of bipedal locomotion [17].

Bipedal humanoid locomotion introduces periodic oscillations in the camera trajectory and intermittent motion blur during the swing phases of each step, which degrade feature tracking and can destabilize pose estimation in algorithms that assume smooth, continuous motion [15]. Stability failures such as slipping or falling produce abrupt camera discontinuities that conventional tracking algorithms are unable to handle. Piperakis et al. [15] addressed these challenges by proposing a dense RGB-D SLAM system that integrates IMU measurements, kinematic information from the robot's joints, and foot contact sensors, as shown in Figure 2.1. Their approach exploits the structure of bipedal walking: foot contact events provide ground-truth pose anchors during the stance phase, while inertial measurements constrain the camera trajectory during the swing phase.

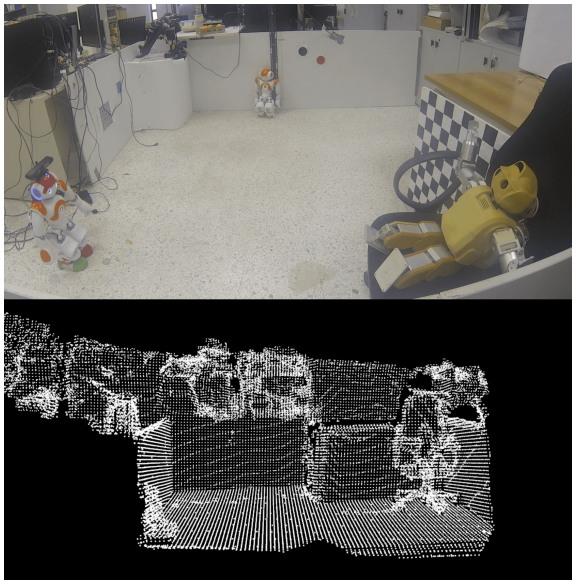

Figure 2.1: RGB-D SLAM reconstructed 3D map from a humanoid platform [15].

A broader survey by Roychoudhury et al. [18] reviews perception methods specifically designed for humanoid robots, covering visual and LiDAR-based localization under walking conditions. The survey notes that humanoid-mounted sensors must contend with viewpoint oscillation, kinematic calibration error, and mechanical compliance in the kinematic chain, factors that are negligible on wheeled platforms. It also highlights that Gaussian EKF-based localization remains a practical choice on humanoids because it is computationally lightweight and can naturally incorporate proprioceptive information from the robot's kinematic model.

Despite these advances, the computational cost of running full visual SLAM online on embedded humanoid hardware, combined with the sensitivity to walking-induced motion artifacts, has

motivated the exploration of localization approaches that decouple mapping from online estimation.

2.1.2 Landmark-Based Localization with the Extended Kalman Filter

Landmark-based approaches estimate the robot's pose relative to a sparse set of known environmental features, either artificial markers or natural landmarks detected at runtime. Rather than building and maintaining a dense geometric map, these methods operate on sparse observations and use probabilistic filters such as the Extended Kalman Filter (EKF) to fuse them with proprioceptive motion estimates [13].

The EKF maintains a state estimate and iteratively refines it in two stages. The *prediction* step projects the state forward using a motion model driven by odometry or velocity commands. The *update* step corrects the prediction using landmark observations, each consisting of a geometric measurement associated with a known landmark position in the map [12]. The EKF also propagates a covariance matrix that quantifies the uncertainty of the estimate, which is critical for both data association and downstream decision-making.

2.1.2.1 Fiducial Markers

Fiducial markers are planar binary patterns placed at known positions in the environment. They provide high-precision pose estimates, are detectable under varying lighting conditions, and require minimal computational resources [19, 20]. Figure 2.2 shows representative examples.

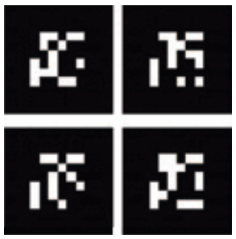

Figure 2.2: Examples of fiducial markers [19].

Lim et al. [19] proposed an EKF-based SLAM approach using fiducial markers with a monocular camera. Their method simultaneously estimates the robot and landmark positions in global coordinates, achieving sufficient accuracy for indoor navigation with only a camera and desktop-printed markers. Lei et al. [20] specifically deployed particle filter localization with artificial markers on the NAO humanoid robot. While particle filters offer flexibility for non-Gaussian distributions, they incur higher computational overhead than the EKF, making the latter more suitable for real-time operation on resource-constrained platforms. Raxit et al. [21] demonstrated that modern CNN-based detectors such as YOLOv8 can identify fiducial markers in real time on embedded GPUs, matching the accuracy of classical geometric detection while meeting the strict timing constraints of robotic perception pipelines.

The primary limitation of fiducial marker-based systems is the need for manual placement and maintenance of markers throughout the environment [19, 20]. At typical printed sizes, markers are not reliably detectable at long distances, which restricts their use to specific operational zones. This has motivated the parallel use of natural environmental features as landmarks.

2.1.2.2 Natural Landmarks and CNN-Based Detection

Natural landmark detection addresses the scalability limitations of fiducial markers by leveraging visual features already present in the environment without requiring manual marker installation [22].

Two families of approaches exist for extracting natural landmarks from camera images. The first relies on classical handcrafted feature descriptors such as Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), or ORB, which detect and describe local image patches that can be matched across viewpoints. These have been used extensively in SLAM front-ends [17] and, more recently, in learned feature-matching pipelines such as SuperGlue [23]. Sarlin et al. [23] introduced a graph neural network that jointly finds correspondences and rejects non-matchable points between image pairs, achieving state-of-the-art results on indoor and outdoor pose estimation, as shown in Figure 2.3. However, feature-matching methods require the map to store per-landmark visual descriptors and need the robot viewpoint to be sufficiently close to the mapping viewpoint for reliable matching [22].

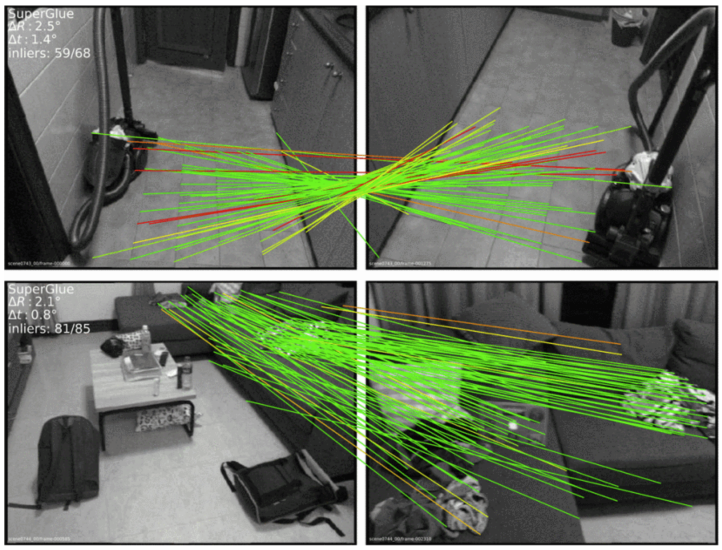

Figure 2.3: Pose estimation through image correspondences using SuperGlue [23].

The second family, which has gained prominence with the maturation of real-time object detectors, treats landmark detection as a semantic object detection problem. Rather than matching local image patches, a CNN is trained to detect specific categories of environmental structures, such as table corners, window frames, or ceiling features, and outputs a bounding box and semantic class for each detection [22]. This approach decouples the detection stage from the geometric

map: the map stores only the 3D positions and class labels of landmarks, not visual descriptors. At runtime, the detector provides semantic detections, and the association with map landmarks is resolved through geometric consistency and class matching. Figure 2.4 shows an example from the laboratory environment.

Figure 2.4: CNN-detected semantic landmarks in the target laboratory environment.

The direct inspiration for the localization system developed in this dissertation comes from recent work by Lopes et al. [24], who applied this principle in the context of robotic soccer. In their framework, a CNN detector identifies structured field landmarks, such as line intersections, goal posts, and penalty marks, and a neural network regresses the robot pose directly from the set of detected landmarks. A Bayesian weighting scheme fuses the CNN-based pose estimate with encoder dead reckoning, producing a continuous pose stream that is robust to intermittent detection failures. The present dissertation adopts the same principle of using a learned detector to extract semantic landmarks, but replaces the neural pose regression with a classical EKF. This substitution is motivated by the need for an interpretable estimator that explicitly models measurement uncertainty, operates at high frequency, and integrates naturally with the proprioceptive velocity commands and the kinematic chain provided by the robot platform.

A practical concern with CNN-based landmark detection is domain sensitivity. Chen [22] notes that models trained on one environment, such as an office, may not transfer well to environments with different visual characteristics, textures, or lighting. This domain gap requires collecting environment-specific training data. In the localization system described in Chapter 4, this is addressed by training the detector on images acquired directly from the robot's head-mounted camera in the target laboratory.

2.1.3 EKF Formulation and Implementation on Humanoid Platforms

The core of landmark-based localization is the EKF. This subsection reviews the components of the EKF formulation that are particularly relevant to humanoid deployment.

Motion model. The EKF prediction step propagates the robot state forward using a motion model driven by proprioceptive inputs. For humanoid robots with omnidirectional locomotion capabilities, the model typically receives forward velocity, lateral velocity, and yaw rate as control inputs. The local displacements over a time step are projected into the global frame using the current heading estimate. The motion model Jacobians with respect to state and control are required for covariance propagation [12]. Hartley et al. [25] proposed the Invariant Extended Kalman Filter (InEKF) for legged robot state estimation, demonstrating that a symmetry-preserving filter formulation yields improved consistency when the robot's motion model respects the Lie group structure of planar or 3D poses. Their contact-aided InEKF incorporates foot contact events as pseudo-measurements, leveraging the intermittent grounded phases of bipedal walking. While the InEKF offers theoretical advantages, its implementation complexity and reliance on accurate contact detection make the classical EKF a more pragmatic choice when the primary correction comes from visual landmarks rather than foot contacts.

Measurement model with dynamic camera pose. On a humanoid platform, the camera is not rigidly attached to the robot base; its pose varies with the head and torso configuration. The measurement model must therefore account for the time-varying transform between the robot trunk and the camera. The camera pose in the world frame is obtained by composing the estimated planar robot pose with the kinematic chain published by the robot's firmware. The predicted observation for a given map landmark is then computed by projecting the landmark position into the camera frame and expressing it in spherical coordinates (range, azimuth, elevation) [13]. Diniz et al. [26] compared EKF and Unscented Kalman Filter (UKF) formulations for trajectory estimation on the NAO humanoid, finding that the EKF achieves comparable accuracy to the UKF for planar localization at a fraction of the computational cost. Their work also confirms that the oscillations induced by walking do not fundamentally degrade EKF convergence when the measurement noise is appropriately tuned.

Observation generation from RGB-D. When landmarks are detected by a CNN, each detection must be converted into a geometric measurement. The 2D bounding box center is paired with a depth estimate extracted from the aligned depth image. Rather than sampling a single pixel, a robust depth estimate is obtained from a central crop of the bounding box, typically using a lower percentile to bias the estimate toward the nearest visible surface and reduce contamination from the background. The 3D point in the camera frame is obtained by back-projection using the calibrated camera intrinsics and then converted to the spherical observation format required by the EKF [27].

2.1.4 Data Association

Data association, determining which map landmark corresponds to each visual observation, is a critical and error-prone step in landmark-based localization. A single incorrect association can inject a large error into the state estimate and, if undetected, can cause filter divergence [28].

The simplest strategy is Individual Compatibility Nearest Neighbor (ICNN), which pairs each observation with the map landmark whose predicted observation minimizes the Mahalanobis distance, provided the distance falls below a chi-squared threshold [12]. ICNN is computationally efficient but can produce incorrect associations in environments with repetitive geometry or ambiguous landmark placement.

Joint Compatibility Branch and Bound (JCBB), introduced by Neira and Tardos [28], addresses this limitation by evaluating the joint compatibility of an entire set of observation-to-landmark pairings. JCBB searches the space of possible associations using a branch-and-bound algorithm, pruning branches whose joint Mahalanobis distance exceeds the chi-squared bound. This global consistency check makes JCBB significantly more robust than ICNN in cluttered or repetitive environments, at the cost of exponential worst-case complexity. In practice, the number of simultaneously observed landmarks is small enough for JCBB to run in real time.

Semantic class information provides an additional, computationally inexpensive gating mechanism. CNN-based detectors assign a class label to each detection (e.g., “table corner”, “window frame”). Before geometric compatibility is evaluated, observations are filtered to consider only map landmarks of the matching semantic category. This semantic gating reduces the number of geometric candidates, accelerates association, and eliminates a large class of potential ambiguities, which is a structural advantage of using a semantic map rather than a purely geometric one[22].

A complementary strategy, used when the robot operates in precision-critical zones, is to deploy fiducial markers with globally unique identifiers. ArUco markers, for example, encode an ID that directly resolves the association without ambiguity. By combining semantic gating for natural landmarks with direct ID-based matching for fiducial markers, the data association problem is effectively scoped to the set of natural detections, where the number of candidates is already constrained by the semantic filter.

2.1.5 Temporal Consistency and Delayed Observations

A practical challenge for EKF-based localization on embedded platforms is the delay between image acquisition and the moment a visual observation is available to the filter. CNN-based detectors such as YOLO require several tens of milliseconds for inference on embedded GPUs. During this interval, the robot continues to move under its own velocity commands, and by the time the measurement is ready the current EKF state no longer corresponds to the pose at which the image was captured.

If this temporal misalignment is ignored, the filter applies a correction derived from a past viewpoint to the current state estimate, introducing a systematic lag that grows with velocity and inference latency. Two mitigation strategies are commonly employed.

One possible mitigation is to down-weight stale observations, for example, by increasing the observation covariance in proportion to the estimated displacement that occurred during the delay. The positional uncertainty can be approximated from the current velocity multiplied by the delay duration, and the angular uncertainty from the yaw rate. In that case, stale observations are assigned lower weight in the update, reducing their influence on the state [29].

Delayed observations are a common issue in robot localization, especially when exteroceptive sensing and communication introduce latency. In mobile robotics, this problem is typically handled through delayed-state filtering or out-of-sequence measurement processing, rather than by applying the correction directly to the current estimate [30, 31].

The mechanism adopted in this dissertation follows that standard pattern. The filter maintains a circular buffer of past state and covariance snapshots at the prediction rate. When a delayed observation arrives, the estimate is rolled back to the corresponding measurement timestamp, the EKF update is applied at that past instant, and the corrected state is then propagated forward again using the stored control history. This rollback-and-replay strategy is closely related to delayed-state filtering formulations described in the robotics literature [32, 33].

For the present system, the main value of this approach is temporal consistency rather than algorithmic novelty. The correction is applied at the correct time instant, which avoids mixing a delayed visual observation with a newer predicted state. At the same time, the computational overhead remains small because the replay step reuses already stored prediction increments. The implementation of this mechanism in the present system is detailed in Chapter 4.

2.1.6 Comparative Analysis of Localization Methods

Table 2.1 summarizes the fundamental trade-offs. For a humanoid robot operating in a structured, known environment, a hybrid strategy emerges as the most practical approach. Semantic landmarks detected by a lightweight CNN provide scalable, infrastructure-free localization across the workspace. Fiducial markers deployed at precision-critical zones (picking stations) offer guaranteed high-accuracy corrections when the robot is in close proximity. Both observation types are fused within a single EKF, benefiting from the same kinematic camera-pose model and the same temporal consistency mechanisms. This design avoids the heavy computational footprint of full visual SLAM while retaining the flexibility to incorporate heterogeneous observation sources, a balance that is well suited to the constraints of embedded humanoid hardware.

Table 2.1: Comparison of vision-based localization methods for humanoid robots.

Method	Accuracy	Comp. Cost	Scalability	Key Limitation on Humanoids
Visual SLAM (ORB-SLAM3) [17]	High	Very High	High	Walking-induced motion blur; requires GPU
SLAM + humanoid sensors [15]	High	High	High	IMU/foot contact fusion adds sensor requirements
Fiducial markers + EKF [19, 20]	Very High	Low	Low	Manual marker placement; limited detection range
CNN-based feature matching [23]	Medium-High	Medium-High	Medium	Viewpoint sensitivity; descriptor storage
CNN-based semantic landmarks + EKF [24]	Medium-High	Medium	High	Training data required per environment
InEKF with foot contacts [25]	High	Medium	Medium	Requires reliable contact detection; complex implementation

2.2 Object Perception for Manipulation

Manipulating objects such as boxes requires the robot to detect them in the camera stream and estimate their spatial position relative to the robot. This section reviews three interrelated aspects of object perception: detection with convolutional neural networks, three-dimensional position estimation from RGB-D data, and, as a broader but related problem, full 6-DoF pose estimation.

2.2.1 CNN-Based Object Detection

Object detection in robotic perception is dominated by convolutional neural networks, with the YOLO family being the most widely adopted for real-time applications. Introduced by Redmon et al. [34], YOLO architectures perform detection and classification in a single forward pass, making them suitable for the strict latency requirements of onboard robotic perception [35]. Modern YOLO variants achieve detection rates of 30–60 frames per second on desktop GPUs and 15–30 frames per second on embedded platforms such as the NVIDIA Jetson when deployed through TensorRT [21].

For the box-handling application targeted by this work, the detector must recognize the target box from different viewpoints and under varying lighting, as illustrated in Figure 2.5. Training

a detector for a specific object class requires an annotated dataset of images captured under representative conditions. Data augmentation techniques, random scaling, rotation, color jitter, and mosaic, are routinely applied to improve robustness to viewpoint and illumination variation. When deployed on embedded hardware, the trained model is typically exported to an optimized inference engine such as TensorRT, which applies layer fusion, precision calibration, and kernel auto-tuning to maximize throughput on the target GPU.

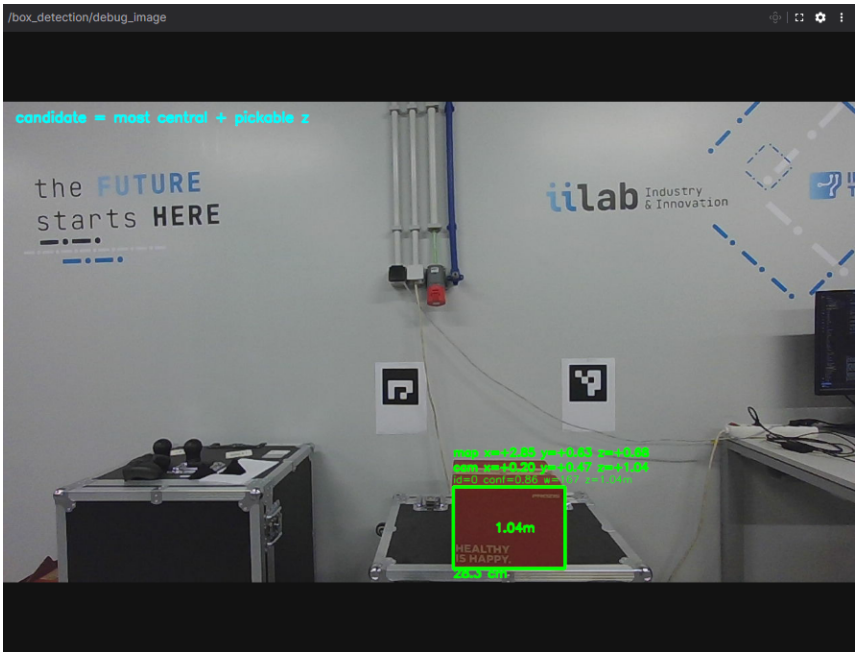

Figure 2.5: YOLO-based detection of the target box.

2.2.2 Three-Dimensional Position Estimation from RGB-D

Once an object is detected in the RGB image, its 3D position must be recovered. With an RGB-D camera that provides registered color and depth streams, the depth value at the pixel coordinates of the detection center can be used to back-project the two-dimensional (2D) detection into a 3D point in the camera optical frame using the calibrated camera intrinsics f_x, f_y, c_x, c_y:

$$X = \frac{(u - c_x)\, d}{f_x}, \quad Y = \frac{(v - c_y)\, d}{f_y}, \quad Z = d, \tag{2.1}$$

where (u, v) are the image coordinates and d is the depth estimate.

The depth estimate is sensitive to noise, missing values at depth discontinuities, and background contamination near the object boundaries. A single-pixel depth sample at the bounding-box center frequently falls on a reflective or textureless surface and returns an invalid reading. A more robust strategy samples depth over a central crop of the bounding box, excluding the edges where background mixing is most likely, and computes a robust statistic such as the 25th

percentile of the valid samples [27]. This biases the estimate toward the nearest surface, which typically corresponds to the front face of the object.

Additional sources of uncertainty include depth sensor noise, which increases quadratically with distance for time-of-flight and structured-light sensors, camera calibration errors in both intrinsics and the extrinsic transform to the robot base, and depth-RGB misalignment due to imperfect sensor registration. These factors propagate through the back-projection and must be accounted for in any manipulation strategy that depends on the estimated 3D position.

2.2.3 Six-Degree-of-Freedom Pose Estimation

While the present work simplifies the object perception problem to 2D detection plus depth-based 3D position estimation, assuming the box orientation is known *a priori* from the structure of the picking station, the broader literature addresses the more general problem of full 6-DoF pose estimation, which recovers both the position and the orientation of an object relative to the camera [36].

Methods for 6-DoF pose estimation can be broadly categorized along two axes: the level of object knowledge required (instance-level versus category-level) and the computational approach (learning-based versus geometry-based). Instance-level methods such as YOLOPose V2 [35] use deep neural networks to regress 2D keypoints and recover full 6D pose from a single RGB image, but require a detailed Computer-Aided Design (CAD) model for each object type. Category-level approaches such as Box6D [37] operate with a generic shape template and estimate both the pose and the object-specific dimensions, enabling generalization to unseen box variants at the cost of reduced accuracy. Geometry-based methods such as that of Cordeiro et al. [38] detect geometric primitives (edges and corners) and compute the pose through classical Perspective-n-Point (PnP) solvers, achieving real-time performance on CPU hardware without training data.

The choice between full 6-DoF pose estimation and the simplified 3D position approach depends on the deployment scenario. When the object is placed on a standardized support structure with known orientation, the three translational DoF are sufficient for successful manipulation, and the simplification avoids the complexity, training data requirements, and computational cost of full pose recovery [36].

2.3 Navigation and Path Tracking

Navigation in logistics environments requires the robot to move reliably between predefined workstations. This section reviews path planning methods, which compute a collision-free route given a map and a goal, and path tracking methods, which guide the robot along an already-defined path using feedback control.

2.3.1 Path Planning

Path planning for mobile robots is commonly divided into search-based methods on discrete grid maps and sampling-based methods in continuous configuration spaces [39]. Search-based planners such as A* and its variants expand nodes on an occupancy grid using a cost function that combines the distance traveled so far with a heuristic estimate of the remaining distance, guaranteeing optimal or near-optimal paths on fixed-resolution maps [39], as shown in Figure 2.6. Sampling-based planners such as Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRT), and their asymptotically optimal variants PRM* and RRT*, construct a graph or tree by randomly sampling configurations and connecting collision-free pairs [40], as illustrated in Figure 2.7. Sampling-based methods scale better to high-dimensional configuration spaces but introduce randomness and typically require post-processing to obtain smooth, executable trajectories.

Figure 2.6: Paths planned by traditional A* (left) and an improved variant (right) [41].

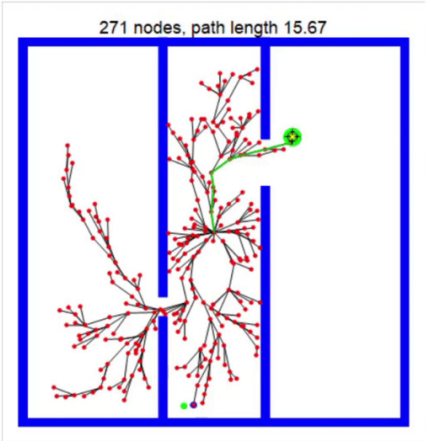

Figure 2.7: Exploration using RRT* in a continuous space [39].

For humanoid robots in logistics, where the global environment can be described accurately by a 2D occupancy grid and the objective is reliable movement between known work areas, grid-based methods such as A* are well aligned with the structured nature of warehouses [39]. In the present work, however, the need for online path planning is circumvented: the positions of picking stations are known in advance, and navigation consists of following predefined waypoint sequences rather than computing paths at runtime.

2.3.2 Path Tracking over Predefined Waypoints

When a path is already available as an ordered sequence of waypoints, the control problem reduces to tracking: computing velocity commands that drive the robot from its current pose toward the next waypoint while minimizing cross-track deviation.

Line-following control is a classical approach in which the path between consecutive waypoints defines a reference line. The along-track error (e_{along}) measures progress toward the goal, and the cross-track error (e_{perp}) measures lateral deviation from the line. A proportional controller on the cross-track error generates a lateral correction velocity, while a constant or gain-scheduled forward velocity drives progression along the path [39].

State-based tracking controllers partition the tracking problem into distinct regimes. In the approach phase (far from the path line), the controller prioritizes reducing the lateral error while aligning the heading toward the path direction. In the track phase (within a tolerance band), a constant nominal speed is maintained with small lateral corrections. In the final approach phase (near the goal), the controller switches to point-and-yaw regulation with reduced speed limits for high-precision positioning. Hysteresis thresholds govern transitions between states, preventing rapid switching when the robot operates near a boundary.

The velocity commands generated in the world frame must be transformed into the robot body frame using the current heading estimate, and the resulting commands must respect per-axis speed limits. On real humanoid platforms, static friction in the leg actuators introduces a minimum velocity requirement: the legs require a startup threshold to initiate motion from rest, and a lower hold threshold can be applied once the robot is already moving. The tracker must also handle pose-stream staleness: if the localization estimate is unavailable beyond a timeout, the controller must safely command zero velocity to prevent uncontrolled motion [7].

The path tracking module described in Chapter 5 implements these principles using the EKF pose estimate as feedback, with waypoints loaded from a configuration file.

2.4 Task-Level Capabilities in Logistics Environments

Beyond perception and navigation, a humanoid robot in logistics must reliably execute task-level capabilities: grasping objects, verifying task outcomes, and operating safely around human workers and other robots. This section reviews established approaches for each capability.

2.4.1 Grasping in Logistics Scenarios

Grasping and picking is the process of computing and executing end-effector configurations that achieve stable contact with an object for manipulation and transport. In warehouse applications, grasping typically targets textureless, planar-faced objects such as boxes.

Depth-based grasp planning predicts grasp poses directly from depth images without relying on rich texture, making it well-suited to this class of objects. Jiang et al. [42] propose a deep learning approach that outputs grasp candidates for planar-faced objects, achieving high success rates with suction-based end-effectors. The DoraPicker system [43] demonstrates how perception, pose estimation, and grasp planning can be integrated into a complete picking pipeline, emphasizing robustness through relatively simple grasp strategies such as top grasps on boxes.

When the target object has known geometry and the manipulation task is repetitive, an alternative to learned grasp planning is to use predefined grasp poses. The grasp sequence (reach, close, lift, and home) is programmed as a set of fixed end-effector targets expressed in the robot trunk frame. This approach eliminates the computational cost of online grasp synthesis and is particularly appropriate for compact humanoid platforms whose arm actuators have limited payload capacity. Dual-arm coordination can be achieved by sending simultaneous end-effector targets to both arms. Passive lifting strategies using custom end-effector extensions, such as L-shaped brackets that slide under box edges and lift from below, avoid the need for sustained gripping force and reduce the torque demand on small arm motors. The grasp sequence implemented in the present system follows this predefined approach and is described in Chapter 5.

2.4.2 Vision-Based Quality Inspection

In logistics workflows, verifying the content or condition of objects after picking can be integrated into the perception pipeline. A simplified version of vision-based quality inspection can be implemented by leveraging object detection models already trained for localization or box identification. Such models can verify product presence, count items, or detect packaging anomalies through multi-class detection without requiring additional specialized training [35]. Quality inspection is not implemented in the present work and remains a direction for future extension.

2.4.3 Collision Avoidance with Humans and Mobile Robots

In shared warehouse environments, humanoid robots must operate safely around human workers and mobile robots. At the global level, route planning and fleet coordination can avoid conflicts between agents. At the local level, a reactive collision avoidance layer responds to unexpected obstacles in the robot's immediate vicinity. Vision-based methods provide a practical approach with minimal additional sensing. Xie et al. [44] present a human–robot collision avoidance method based on RGB cameras and human pose estimation, defining safety zones around the robot and triggering protective behaviors when a person enters these zones. Collision avoidance is not addressed in the present system, which operates under the assumption of a static environment without

dynamic obstacles; it is identified as an important direction for real-world deployment beyond the controlled laboratory setting.

2.5 Robotic Simulation Platforms

Robotic simulation platforms provide safe, repeatable virtual environments for testing control modules before deployment on physical hardware. In the present work, the Isaac Sim platform was used to validate the locomotion and path tracking behavior of the Booster T1. Built on the NVIDIA Omniverse platform with GPU-accelerated PhysX physics and ROS 2 integration, Isaac Sim enables testing of velocity-based locomotion commands and finite-state controllers in a controlled environment, reducing the risk of damaging the physical robot during initial development [45]. Other ROS 2-compatible simulators such as Gazebo and Webots offer alternative trade-offs in terms of physics fidelity, ease of setup, and hardware requirements [45]. The simulation, seen in Figure 2.8 was used exclusively for locomotion and path tracking validation; the perception pipeline and arm manipulation modules were developed and tested directly on the physical robot.

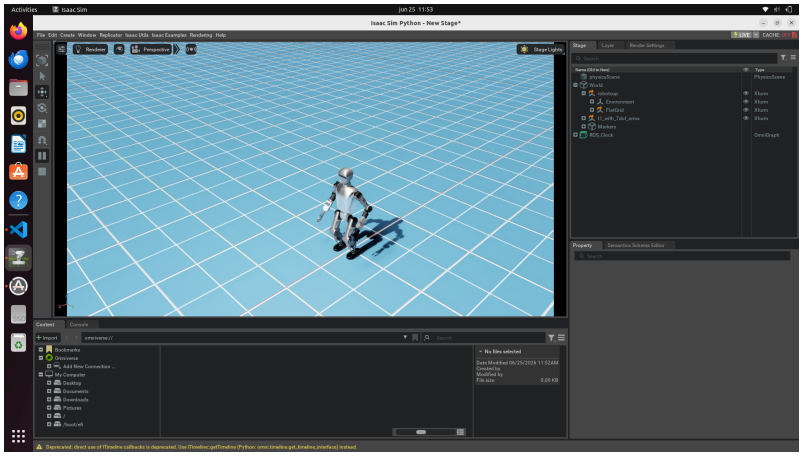

Figure 2.8: Isaac Sim simulation environment.

Chapter 3

System Architecture and Proposed Approach

This chapter presents the robotic platform, the overall software architecture, and the end-to-end mission pipeline that enables the Booster T1 humanoid robot to perform box-handling tasks in a structured indoor environment. The chapter begins with a detailed description of the hardware, the onboard computation, the control interfaces, and the reference frame conventions used throughout the system. It then describes the software tools and development environment, followed by the modular ROS 2 architecture, the communication layer between the perception and motion boards, and the sequence of phases that form a complete picking mission. The chapter also presents the high-level orchestrator state machine that sequences the mission phases, and the simulation environment used to validate the locomotion control before real-robot deployment. The goal is to provide a unified system-level view that contextualizes the technical details presented in the subsequent chapters. The system is developed under a set of assumptions that define its scope: the environment is static and structured, with a pre-built semantic map containing the positions of known landmarks and ArUco markers; the target box has fixed geometry and appearance; the picking stations are at known map locations; perception is treated as detection of known features rather than general object recognition; and the robot operates on a flat floor without dynamic obstacles. These assumptions allow the work to focus on localization, navigation, and manipulation integration rather than on scene reconstruction, general perception, or dynamic replanning. A more in-depth discussion of the system's operating assumptions and limitations is provided in Chapter 6.

3.1 Robotic Platform and Hardware

The robotic platform is the Booster T1, shown in Figure 3.1, a compact humanoid robot developed for research and application prototyping. The robot integrates bipedal locomotion, upper-body manipulation, and onboard sensing in a single platform, making it suitable for logistics-oriented tasks that require mobility, perception, and physical interaction with the environment.

Figure 3.1: Booster T1 humanoid robot.

The T1 follows a conventional humanoid morphology composed of a head, torso, two arms, and two legs. This configuration enables the robot to operate in human-oriented environments while combining locomotion and manipulation in a unified body structure. The version used in this work is the standard configuration equipped with the default upper-limb actuators and custom end-effectors adapted for the target box-handling tasks. The robot has a total of 23-DoF: 2 in the head (yaw and pitch), 4 in each arm, 1 in the waist, and 6 in each leg. This actuation layout provides sufficient mobility for the intended task sequence while maintaining a relatively compact morphology. For the present work, the most relevant components are the head-mounted sensing unit, the dual-arm manipulation system, and the lower-body locomotion structure that supports navigation in the workspace.

The robot is equipped with a head-mounted depth camera used to acquire visual and geometric information from the environment. In the experimental setup, the camera is an *Orbbec Gemini 2 335L*. This sensor integrates dual global-shutter infrared cameras, an RGB camera, and a 6-axis IMU, enabling the acquisition of both color and depth information. The camera is mounted on the robot head, as shown in Figure 3.2, which allows the sensing direction to be adjusted through the head yaw and pitch joints. This is particularly relevant for warehouse tasks, since the robot must observe nearby structures, transported boxes, and visual landmarks distributed throughout the environment. The depth stream is used to estimate the spatial configuration of nearby objects, while the RGB stream supports image-based detection. The integrated IMU also provides additional motion-related information that can support localization and state

estimation modules. From a software perspective, the camera is accessed through ROS 2 using the manufacturer-provided driver. The driver publishes color and depth image topics, as well as the corresponding camera calibration information, allowing these data streams to be integrated into the remaining modules of the robotic pipeline.

Figure 3.2: Orbbec Gemini 2 335L camera mounted on the robot head.

For manipulation, the T1 is equipped with dual grippers that are used to interact with the target box. The arm configuration provides the ability to position the end-effectors in front of the body and around the upper workspace associated with support structures such as tables and racks. In the present work, the stock end-effectors are replaced by custom L-shaped extensions, as shown in Figure 3.3. These extensions slide under the box edges, allowing the box to be lifted from below without requiring the arm motors to exert sustained gripping force. The arms move to a closed position and then lift upward; the box rests on the horizontal ledges of the L-shaped pieces, with the vertical flanges providing lateral constraint. This passive lifting strategy avoids loading the small arm actuators with the full box weight during transport.

Figure 3.3: Custom L-shaped end-effector extensions.

The T1 uses a split onboard computing architecture composed of two main processing boards:

a perception board (NVIDIA Jetson) and a motion board. The perception board is responsible for vision-related processing, neural inference, and localization. The motion board is responsible for low-level locomotion control, body-mode management, and arm actuation. This separation is a deliberate architectural choice. The perception board runs the computationally heavy modules (object detection with TensorRT, EKF localization, and visual processing) without interfering with the real-time control loops on the motion board. The motion board, in turn, runs the path tracker, the alignment controller, and the arm executor with low latency and direct access to the robot SDK. The two boards communicate via the onboard Ethernet network using ROS 2 topics on the same domain, which provides a clean interface between the high-level perception pipeline and the low-level motion control.

The robot supports omnidirectional walking, including forward, backward, lateral, and rotational motion. These locomotion capabilities are controlled by sending velocity commands to the robot SDK. The ‘move_executor_node‘ on the motion board subscribes to ‘/booster/move_cmd‘ (type ‘geometry_msgs/Vector3‘) and forwards the commands to ‘B1LocoClient‘ via the Booster SDK. The three components of the vector represent forward velocity (x), lateral velocity (y), and angular velocity (yaw rate). The SDK internally maps these to the joint-level trajectories that produce the corresponding omnidirectional gait. The motion board also handles the robot operating modes: *DAMP*, *PREP*, *WALK*, and *CUSTOM*. For the task pipeline, the relevant modes are PREP (stable standing posture) and WALK (locomotion enabled). The ‘move_executor_node‘ can optionally transition to WALK mode automatically when a non-zero motion command is received.

The arms are controlled via the ‘arm_grasp_executor_node‘, which accepts high-level commands (reach, close, up, home) and translates them into dual-arm end-effector poses sent to the SDK. Each pose is defined as a position (x, y, z) and an orientation $(roll, pitch, yaw)$, expressed in the robot trunk frame. The SDK command is ‘MoveDualHandEndEffector‘, which moves both arms simultaneously to the specified target poses over a configurable duration. The node also supports ‘StopHandEndEffector‘ (home), which returns the arms to a resting position. The head has 2 DoF (yaw and pitch) and carries the camera. In the present pipeline, the head is not actively controlled via ROS 2 nodes; its pose is obtained passively from the kinematic chain published by the robot firmware. The EKF localization node uses this kinematic chain to compute the camera pose in real time, as described in the following paragraph.

The system relies on a well-defined transform chain to relate sensor data, robot body parts, and the global map. The robot firmware publishes a set of body frames that describe the kinematic state of the platform at runtime. The most relevant frames are:

- **‘map‘**: the global fixed frame in which the semantic map and the robot’s planar pose are defined.
- **‘Trunk‘**: the robot base frame, corresponding to the torso. The EKF localization node estimates the planar pose of this frame in the ‘map‘.
- **‘Waist‘**: the waist joint frame, child of ‘Trunk‘, used as the height reference for the kinematic chain.

- **‘head_point‘**: the head reference frame, child of ‘Waist‘. The camera is mounted on the head and its optical frame is related to ‘head_point‘ by a static calibration transform.
- **‘left_hand_link‘ / ‘right_hand_link‘**: the elbow frames published by the robot firmware. These are not the physical end-effector tips; a static translation transform is applied to offset from the elbow to the actual grasping point at the end of the custom L-shaped extension. The offset was determined from the mechanical drawing and validated against the kinematic monitor and the real robot pose.
- **‘left_foot_link‘ / ‘right_foot_link‘**: the foot frames, used by the kinematic model to determine the trunk height above the ground.

The kinematic chain, shown in Figure 3.4, is used in two main ways. First, the EKF localization node uses a ‘CameraPoseProvider‘ to query the transform from ‘Trunk‘ to the camera optical frame via the ‘head_point‘ frame. This allows the filter to project the estimated planar trunk pose into a full 3D camera pose, which is needed to backproject visual detections into the map. Second, the box detector uses the same chain to transform detected 3D points from the camera frame into the ‘Trunk‘ frame, yielding the target position in robot-centric coordinates. The ‘Trunk‘ frame is also the reference in which the arm executor defines its grasp poses. The end-effector poses sent to the SDK are therefore expressed in ‘Trunk‘ coordinates, with the static elbow-to-tip offset already incorporated.

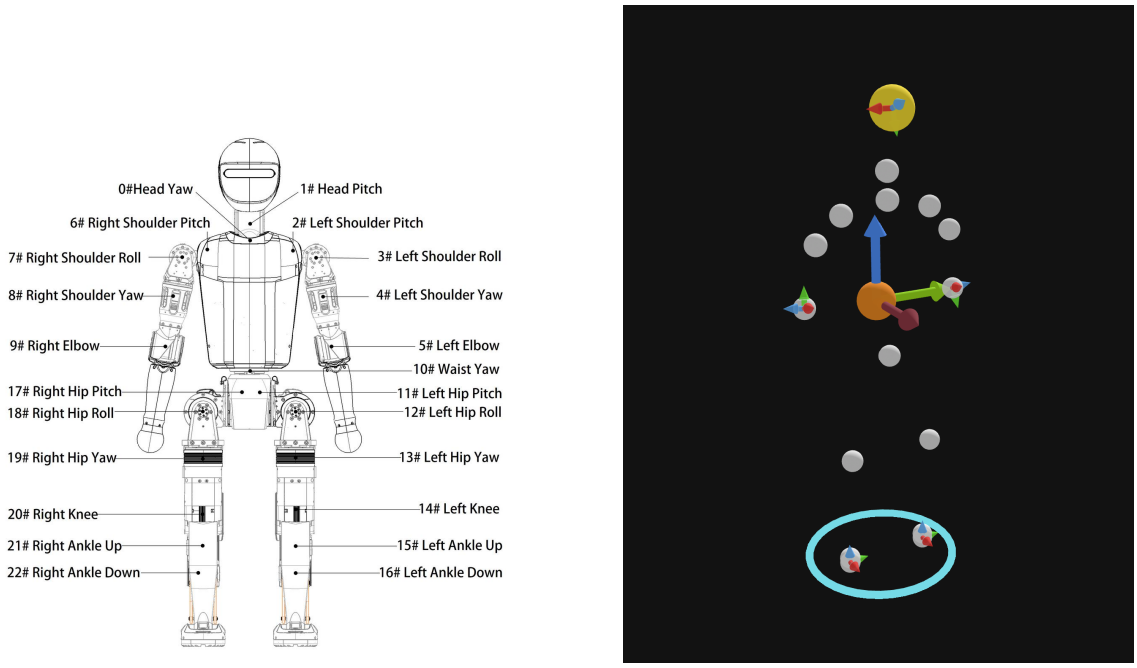

Figure 3.4: Kinematic structure of the Booster T1 humanoid robot. (Left) Reference kinematic model provided in the manufacturer’s documentation. (Right) Subset of frames and joints effectively used by the proposed system.

3.2 Software Tools and Environment

The system was developed using the software tools and frameworks listed in Table 3.1. ROS 2 provides the middleware for communication between the perception and motion boards. The path tracker and the alignment executor are implemented in Python, and the arm executor uses the Booster Robotics SDK. The visual detector is a YOLO v8 model trained with Roboflow and exported to TensorRT for deployment on the NVIDIA Jetson. The dataset annotation and offline analysis were performed with Python, OpenCV, and custom scripts. The simulation stage used Isaac Sim. For debugging and monitoring, Foxglove was used to visualize ROS topics and the state machine during real-robot experiments.

Real-robot data collection for both training datasets relied on rosbag2, which was used to record RGB and depth streams from the laboratory sessions. The resulting bags were processed offline with custom scripts that extracted the image frames and filtered similar images. This pipeline produced compact, diverse datasets for both the box-detection and the landmark-detection models without requiring manual on-site photography. Development and deployment on the embedded boards were done via Visual Studio Code with the Remote SSH extension, which provided direct access to the onboard file system, the ROS workspace, and the live terminal on the robot.

Table 3.1: Software tools and frameworks used in the system

Tool / Framework	**Role**
ROS 2	Middleware for inter-board communication and node orchestration.
Python	Programming language for mission orchestration, path tracking, visual alignment, data processing, and analysis scripts.
C++	Programming language used for the localization pipeline, ROS 2 nodes, and integration with the Booster SDK.
Booster Robotics SDK	Low-level interface for locomotion and arm control.
YOLOv8	Object detection backbone for box and landmark detection.
TensorRT	Optimized inference engine for the Jetson GPU.
Roboflow	Dataset annotation and model training platform.
Google Colab	Cloud training environment for YOLO model experiments.
MATLAB	Simulation, and EKF performance analysis.
OptiTrack	Motion-capture ground-truth system for real-world localization evaluation.
Isaac Sim	Physics-based simulation for locomotion validation.
Foxglove	ROS topic visualization and debugging interface.
rosbag2	Recording of real-robot sensor data for offline dataset generation.
Visual Studio Code (Remote SSH)	Remote development and debugging on the onboard boards.

3.3 Overall System Architecture

The software system is organized as a modular ROS 2 architecture in which each major function is implemented as an independent node. The nodes are distributed across the two onboard boards according to their computational and latency requirements. Table 3.2 summarizes the node distribution and the rationale.

Table 3.2: Module distribution across the two onboard boards

Node	Board	Description
‘ekf_localization_node‘	Perception Board	Runs the EKF localization pipeline, fusing visual and depth-based observations with the semantic map to publish the robot pose.
‘aruco_detector_node‘	Perception Board	Detects ArUco markers and publishes marker observations that can be fused by the localization stack.
‘box_detector_node‘	Perception Board	Runs box detection and computes the target candidate used by the alignment and grasp pipeline.
‘orchestrator_node‘	Perception Board	Implements the high-level mission logic and operator interface, coordinating the mission phases across the system.
‘path_tracker_node‘	Motion board	Tracks map-frame waypoints and generates locomotion commands for the navigation phase.
‘move_executor_node‘	Motion board	Bridges ROS 2 velocity commands to the Booster locomotion SDK and sends them to the robot.
‘align_approach_executor_node‘	Motion board	Executes visual alignment, forward approach, and retreat motions using motion-board control loops.
‘arm_grasp_executor_node‘	Motion board	Controls the dual-arm grasp sequence, including reach, close, up, and home presets.
‘measured_vel_adapter_node‘	Motion board	Converts robot feedback into velocity estimates used by the localization prediction step.
‘arm_grasp_monitor_node‘	Motion board	Monitors arm torque for safety and triggers a return to home on sustained abnormal load.

The perception board runs the nodes that require GPU acceleration and produce the global pose estimate (‘/localization/predicted_pose‘). The motion board runs the nodes that convert the

pose estimate and the visual target into concrete motion commands. The two boards share the same ‘ROS_DOMAIN_ID‘, so the topics of interest cross the network transparently.

The inter-board communication is based on standard ROS 2 topics over the onboard Ethernet network. The most critical data stream is the pose estimate published by the EKF node on the Jetson, which is consumed by the path tracker and the alignment executor on the motion board. The visual target stream published by the box detector is also consumed by the motion board. The ‘measured_vel_adapter_node‘ on the motion board converts the raw odometry stream from the robot firmware into velocity commands that feed the EKF prediction step on the perception board. In the reverse direction, the motion board publishes velocity commands and arm status that are used by the mission orchestration logic on the Jetson. The ‘arm_grasp_monitor_node‘ provides an additional safety layer by monitoring the torque on both arms and triggering a return to the home position if unexpected contact or grasp failure is detected.

3.4 ROS 2 Communication Layer

The main ROS 2 node distribution and the topics that connect them are shown in Figure 3.5. The pose topic (‘/localization/predicted_pose‘) is the central spatial reference used to track the robot’s location. The box detector publishes the target position in the ‘Trunk‘ frame via ‘/box_align/target‘. The ‘align_approach_executor_node‘ subscribes to this topic to drive the visual servoing loop, and the ‘orchestrator_node‘ subscribes to it to confirm the presence of the box before advancing the state machine. The orchestrator commands the path tracker and the alignment executor, and the motion board nodes publish their status back. The velocity command topic ‘/booster/move_cmd‘ is a multiplexed bus: the path tracker publishes on it during navigation, and the alignment executor publishes on it during align and approach. The ‘move_executor_node‘ consumes these commands and forwards them to the SDK. The key topics and their publishers and subscribers are listed in Table 3.3.

Because the two boards share the same ROS 2 domain and are connected by the onboard Ethernet switch, the topics cross the network transparently. The latency of the pose stream between the Jetson and the motion board is low enough for the path tracker and the alignment controller to operate reliably. The ‘orchestrator_node‘ on the Jetson does not directly command the robot motors; it only issues high-level commands (start path, start align, arm reach) to the motion board nodes, which then execute the low-level control loops.

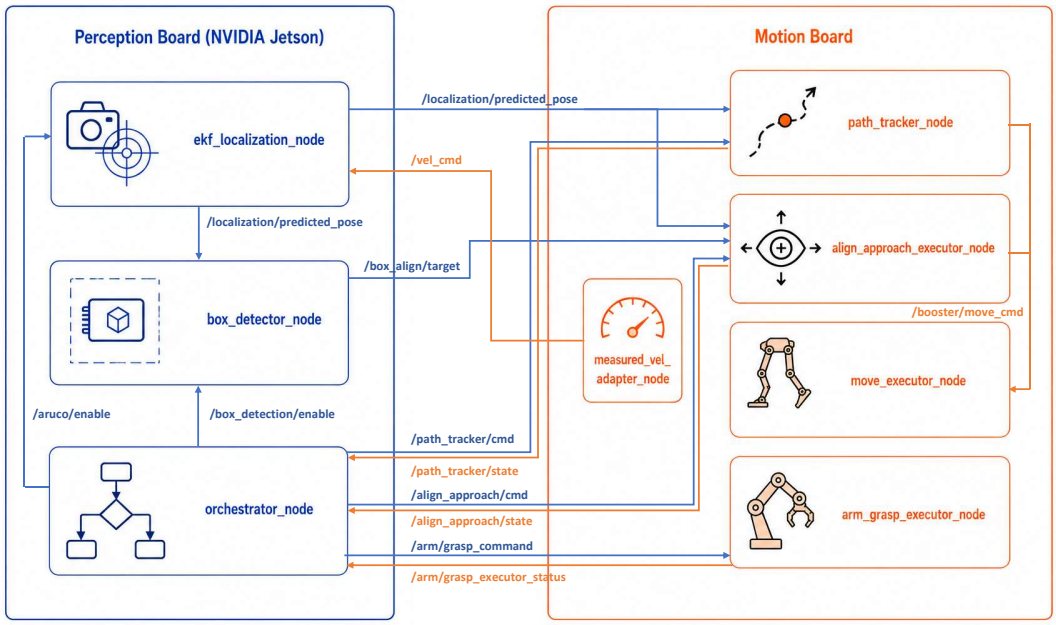

Figure 3.5: Onboard computing architecture and communication via ROS 2 topics over the onboard Ethernet network.

Table 3.3: Key ROS topics in the mission pipeline

Topic	**Publisher**	**Subscriber(s)**
‘/localization/predicted_pose‘	‘ekf_localization_node‘	‘path_tracker_node‘, ‘align_approach_executor_node‘
‘/box_align/target‘	‘box_detector_node‘	‘align_approach_executor_node‘, ‘orchestrator_node‘
‘/box_detection/enable‘	‘orchestrator_node‘	‘box_detector_node‘
‘/path_tracker/cmd‘	‘orchestrator_node‘	‘path_tracker_node‘
‘/path_tracker/state‘	‘path_tracker_node‘	‘orchestrator_node‘
‘/booster/move_cmd‘	‘path_tracker_node‘, ‘align_approach_executor_node‘	‘move_executor_node‘
‘/align_approach/cmd‘	‘orchestrator_node‘	‘align_approach_executor_node‘
‘/align_approach/state‘	‘align_approach_executor_node‘	‘orchestrator_node‘
‘/arm/grasp_command‘	‘orchestrator_node‘	‘arm_grasp_executor_node‘
‘/arm/grasp_executor_status‘	‘arm_grasp_executor_node‘	‘orchestrator_node‘
‘/orchestrator/command‘	Operator / CLI	‘orchestrator_node‘
‘/orchestrator/state‘	‘orchestrator_node‘	Debugging / Foxglove

3.5 End-to-End Mission Pipeline

From the operator's perspective, the system is organized as a set of independent high-level tasks launched from a common standby state. Instead of enforcing a fixed mission sequence, the operator or mission script can choose the next task according to the desired experiment: a predefined path can be executed with a command such as 'pt[0]', 'pt[1]', and so on, and a local picking routine can be launched with the 'pick' command once the robot is positioned near a station. After each task completes, or if it is interrupted with the abort command 'k', the system returns to standby and waits for the next command. This section describes this task-level control logic; the detailed local picking routine is described separately in Section 3.6.

The task-level command structure is shown in Figure 3.6. At this level, the system behaves as a command dispatcher: each high-level task is launched from 'STANDBY', runs independently, and returns to 'STANDBY' either after completion or after an abort command.

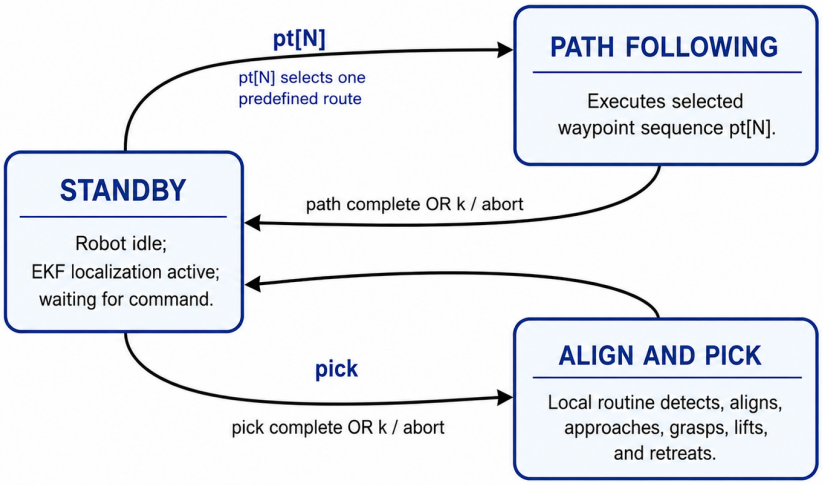

Figure 3.6: Task-level command state machine.

The path tracker receives the static waypoints from the Tom's Obvious, Minimal Language (TOML) configuration file and follows them according to the desired path. The command 'pt[i]' starts the waypoint sequence with index i. Each sequence can represent a different route through the workspace, such as moving from the initial pose to a picking station, moving between stations, or returning to a safe pose. These paths are not executed in a hard-coded order; each one is selected explicitly as a separate task.

The task-level transitions are therefore simple: from 'STANDBY', a 'pt[N]' command enters 'PATH FOLLOWING', and the system returns to 'STANDBY' when the path is completed or when 'k' is received. Similarly, a 'pick' command enters the picking task, and the system returns to 'STANDBY' when the pick routine completes or when 'k' is received. This command-dispatch structure keeps the station-level mission logic flexible and allows the operator to compose experiments by choosing the order of path and pick tasks.

3.6 Picking State Machine

The ‘orchestrator_node‘ implements the state machine used for the local picking routine. It runs on the perception board and communicates with the motion board nodes via the ROS 2 topics listed in Table 3.3. Unlike the task-level mission pipeline described above, this state machine starts once the robot is already positioned near a picking station.

The picking states are defined as follows:

- **IDLE** - The initial state. The robot is standing, the box detector is disabled, and the system is waiting for an operator command.
- **WAIT_BOX** - The box detector and ArUco detector are enabled. The orchestrator counts valid target frames and, in parallel, sends the ‘reach‘ command to prepare the arms for grasping. The state only advances when both the box has been confirmed and the reach motion has completed.
- **MOVE** - The orchestrator sends the ‘start‘ command to the alignment executor. This lower-level executor performs the lateral alignment and the forward approach automatically.
- **SETTLE** - A brief pause after the approach is complete, giving the legs enough time to stop the residual trotting motion that can persist briefly after walking.
- **ARM_CLOSE** - The orchestrator sends the ‘close‘ command to the arm executor.
- **ARM_UP** - The orchestrator sends the ‘up‘ command to lift the box.
- **RETREAT** - The orchestrator sends the ‘retreat‘ command to the alignment executor, which moves the robot backward by a fixed distance.
- **DONE** - The local pick is complete. The detectors are disabled and the system waits for the next command.

The transitions are triggered by status messages from the detector, the arm executor, and the alignment executor. The ‘MOVE‘ state hides a lower-level sequence inside the alignment executor: first the robot corrects its lateral offset with respect to the box, then it moves forward to the dynamically generated grasp waypoint using the EKF pose.

The operator interacts with this routine via the topic ‘/orchestrator/command‘. The command ‘pick‘ starts the local picking sequence from ‘IDLE‘, while ‘k‘ aborts the current routine immediately, sends stop commands to the path tracker and the alignment executor, disables the detectors, and returns the system to ‘IDLE‘. The orchestrator also publishes its current state on ‘/orchestrator/state‘ for debugging and visualization in tools such as Foxglove.

An important extension of this structure would be the addition of a complementary ‘place‘ task. This routine could reuse the same station-level navigation, perception activation, lateral alignment, forward approach, settling, and retreat logic as the pick routine. The main difference

would be in the arm command sequence: instead of preparing the arms with ‘reach‘, closing around the box, and lifting with ‘up‘, the place routine would start from the transported ‘up‘ posture, move the arms down to the ‘close‘ posture to place the box on the support surface, and then return to ‘reach‘ or ‘home‘ to release and clear the object. In this sense, placing can be treated as a mirrored manipulation routine built on top of the same locomotion and alignment infrastructure, making it a natural direction for future work.

3.7 Simulation Environment

The locomotion and control modules were first validated in simulation before being transferred to the physical robot. The simulation environment was built in Isaac Sim, a physics-based simulation platform that supports ROS 2 integration.

The simulation environment includes the T1 humanoid model and a flat test floor, as shown in Figure 3.7, it also includes the line representation for the path tracking framework. The primary purpose of the simulation was to validate the low-level locomotion control and the path tracking behavior before running on the physical platform. The ‘move_executor_node‘ and the ‘path_tracker_node‘ were tested in Isaac Sim to verify that the velocity commands produced the expected robot motion and that the path tracking states (APPROACH, TRACK, FINAL) transitioned correctly. The simulation did not include a full warehouse scene, the box detector, or the arm manipulation pipeline; those components were developed and tested directly on the physical robot in the laboratory environment.

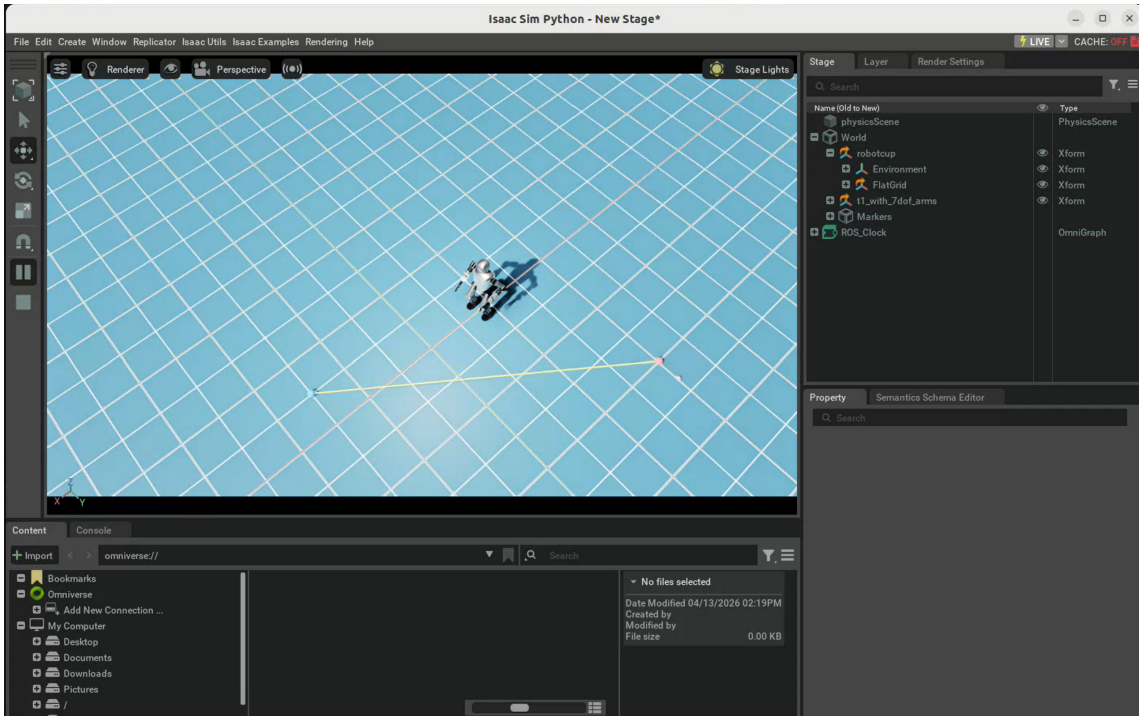

Figure 3.7: Path tracking in simulation.

The simulation served two main purposes. First, it allowed the path tracker and the move executor to be tuned and debugged in a safe environment, without the risk of damaging the physical robot. Second, it provided a repeatable testbed for verifying the integration between the ROS 2 nodes and the robot SDK interface, ensuring that the topic subscriptions, command mappings, and state transitions were correct before deployment. A video of the simulation validation is available in Appendix 7.1.

Chapter 4

Visual Localization

This chapter presents the visual localization system that estimates the planar pose of the Booster T1 humanoid robot in a known structured indoor environment. The chapter begins with the problem formulation and the design rationale behind the adopted EKF approach. It then describes the semantic map of the environment, the YOLO-based feature detector trained for the laboratory landmarks, and the observation generation pipeline that converts RGB-D detections into geometric measurements. The EKF formulation is presented in detail, including the motion model, the measurement model with dynamic camera pose, the update step, and the validation mechanisms. The chapter also covers the data association strategy that matches visual observations to map landmarks, the temporal robustness measures, including delayed updates, and the complementary ArUco marker system that provides high-precision corrections in picking zones. Finally, the experimental evaluation is presented, comprising a real-world OptiTrack comparison and controlled simulation studies. The output of this module is a continuous 30 Hz pose estimate that serves as the spatial reference for the navigation, alignment, and manipulation modules described in the next chapter.

4.1 Localization Problem Formulation

The robot is equipped with a head-mounted RGB-D camera and a proprietary motion controller that provides planar dead reckoning estimates together with a complete kinematic description of the body. The availability of these two sources of information is central to the design choices presented in this chapter. The dead reckoning provides a continuous proprioceptive motion signal, while the kinematic chain exposes the full 3D pose of both the robot's body and camera at any instant. These capabilities allow the localization problem to be formulated in a compact planar form without sacrificing the geometric richness of the 3D visual observations.

The state to be estimated is therefore defined as the planar pose of the robot,

$$\mathbf{x} = [x \;\; y \;\; \theta]^T, \tag{4.1}$$

where x and y represent the robot position in the global map reference frame and θ represents its yaw orientation. The vertical coordinate z, as well as the roll and pitch angles of the torso, are not included in the state vector. Because the robot platform maintains its own kinematic model and publishes the spatial transforms of the trunk, waist, and head frames, the height and attitude of the base can be recovered directly from the robot's internal state at runtime. In other words, the pose of the camera in the world frame is obtained by composing the estimated planar pose with the time-varying transform between the robot base and the camera, which is provided by the kinematic chain. This separation keeps the estimation state minimal while still allowing the measurement model to operate with full 3D geometry. It is therefore consistent with the navigation requirements of the considered environment, where motion is essentially planar, and the internal kinematics of the robot are sufficiently accurate to provide the missing vertical and angular components.

The localization system uses visual observations extracted from RGB-D data acquired by the head-mounted camera. These observations correspond to persistent semantic features in the environment, such as corners, edges, or other identifiable structural elements that can be detected reliably and associated with landmarks stored in the semantic map.

In addition to the natural semantic landmarks, the system is also able to exploit fiducial ArUco markers placed in the environment. These markers serve a complementary role: while the semantic landmarks support continuous localization during navigation, the ArUco markers provide high-precision corrections in specific operational zones such as picking stations, where the robot must align accurately with objects or workstations. The ArUco observations are extracted by a lightweight OpenCV-based pipeline using cv2.aruco and solvePnP, and are fused within the same EKF framework. The detector is callback-driven and processes the incoming RGB stream at the camera frame rate, without a dedicated internal timer, so its computational overhead remains modest compared with the CNN-based landmark detector. Unlike semantic landmarks, ArUco markers bypass the semantic gating stage because their identities are known a priori, allowing direct ID-based association. The ArUco observations also benefit from the same dynamic camera-pose and delayed-update mechanisms described in the following sections.

Given these inputs, the output of the localization system is a continuous estimate of the robot's planar pose and its associated uncertainty. This estimate is then used by the remaining modules of the robotic system as a spatial reference for decision-making and task execution.

The proposed approach is inspired by recent work in vision-based localization for robotic soccer, in which semantic natural landmarks are detected by a convolutional neural network and subsequently used to regress the robot pose [24]. In that framework, the detector identifies structured field markers and the pose is estimated directly by neural regression, combined with encoder dead reckoning through a Bayesian weighting scheme. The present work adopts the same principle of using a learned detector to identify semantic natural landmarks, but replaces the neural pose-regression stage with a classical EKF. This change is motivated by the need for an interpretable, lightweight estimator that can operate at high frequency and that explicitly models measurement uncertainty. The EKF formulation is also naturally compatible with the proprioceptive dead reckoning and the kinematic chain provided by the robot platform, since the camera pose can

be injected into the measurement model directly from the transforms published by the robot.

The choice of an EKF-based approach rather than a full visual SLAM system is not a claim of universal superiority, but a decision driven by the specific characteristics of the operational scenario. The environment is structured, largely known in advance, and intended for repeated task execution. Under these conditions, a pre-built semantic map is available, and the problem reduces to localizing the robot relative to that known map rather than building the map online. Although EKF-based SLAM can also be formulated with markers or landmarks, its practical cost grows as the number of mapped features increases, especially when the map must be maintained and updated online. Visual SLAM systems such as ORB-SLAM3 or Real-Time Appearance-Based Mapping (RTAB-Map) are designed to perform simultaneous mapping and localization, which involves dense or semi-dense map optimization, bundle adjustment, loop closure, and more complex map management. These capabilities are powerful when the environment is unknown, but they are also significantly more computationally demanding than a filter-based localization pipeline. By contrast, the proposed EKF updates a low-dimensional planar state at high rate using lightweight measurement fusion, making it much better suited to embedded execution. This reduction in computational load is particularly important here, because the available processing budget is already shared with the neural detector running on the same onboard hardware. The choice therefore reflects a targeted trade-off between accuracy, interpretability, and computational efficiency in a known environment with limited onboard resources.

In this framework, the use of semantic environmental features provides a natural way to constrain data association, making the update stage more robust than purely geometry-based matching in visually repetitive scenes. The semantic map is constructed offline and contains the known 3D positions of structural landmarks that the detector can recognize. Because the landmarks are semantically classified, each observation can be matched only against map features of the same class, which reduces ambiguity and simplifies the association problem.

4.2 Semantic Map of the Environment

The localization method assumes the existence of an offline map describing the relevant environmental features that can be observed by the robot during operation. Rather than representing the environment as an unstructured geometric space, the adopted map is semantic in nature, meaning that each mapped feature is described not only by its position but also by an associated category. This representation is particularly useful for visual localization, since the observation process begins with a semantic detection stage.

The map is stored as a plain text file in comma-separated format and loaded at runtime by the `MapDatabase` component of the EKF node. Each entry in the map contains four elements: a textual label, a 3D position (x, y, z) in the map frame, and a semantic category. The map used in the experimental environment, seen in Figure 4.1, contains approximately 30 to 40 landmarks and was constructed by manual measurement with a tape measure and laser, relying on structural

assumptions such as parallelism between walls and orthogonal table arrangements. This construction method is adequate for a structured laboratory environment, although it introduces a floor of map-related uncertainty that contributes to the overall localization error observed in real-world experiments.

The mapped features are distributed across a set of semantic categories that correspond to the classes learned by the visual detector. The principal categories include upper table corners, table wheels, window corners, ceiling features, and wall structures and text. These categories reflect distinct classes of relevant landmarks in the scene and are assigned class identifiers that the detector outputs at runtime. This distinction is important because the system does not attempt to match each visual observation against all landmarks in the map indiscriminately. Instead, the semantic class predicted in the visual detection stage is used to restrict the set of possible map correspondences before geometric validation is applied.

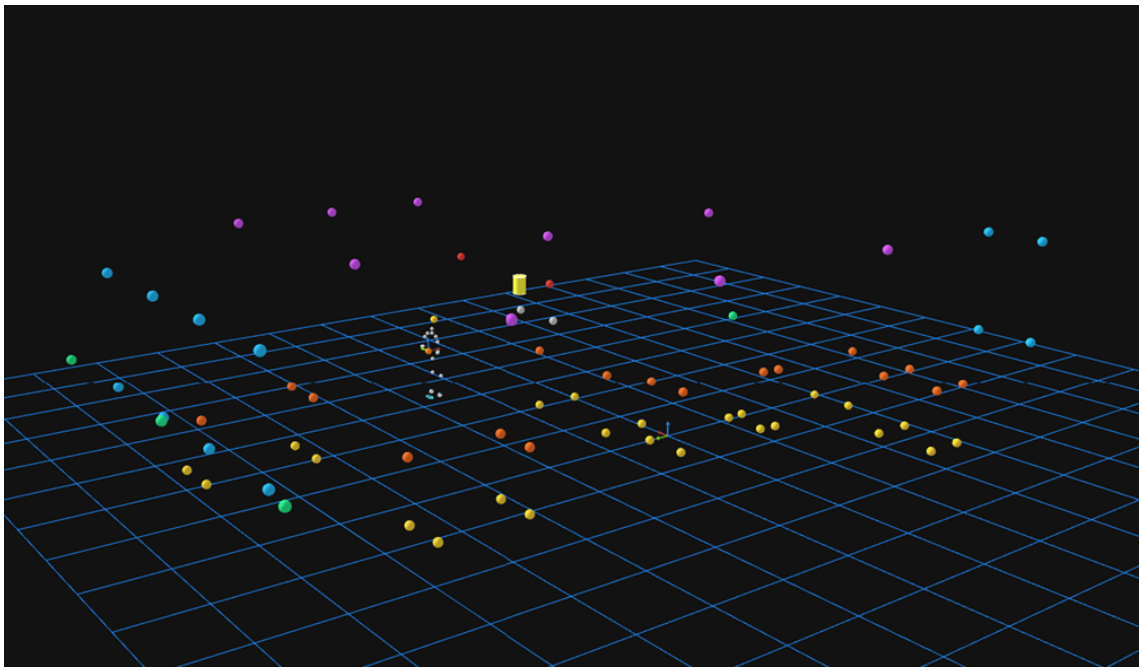

Figure 4.1: Mapped environment including both semantic landmarks ArUco markers.

In addition to the natural semantic landmarks, the map also contains the known positions of any ArUco markers deployed in the environment. These markers are treated as a special category of map features. Because their positions are known with high precision and their detection is robust to viewpoint variation, they provide strong geometric constraints during the update step.

4.3 Feature Detection for Localization

The visual perception module is responsible for detecting features that can be used as observations by the localization filter. The adopted solution uses a YOLO-based detector trained specifically for the environment and feature categories relevant to the localization task.

The detector was trained on a dataset of approximately 1000 annotated RGB images acquired directly from the robot-mounted camera. The dataset was collected under the same sensing conditions expected during runtime, including the same camera viewpoint, lighting conditions, and motion-induced blur that the robot experiences during navigation. Standard data augmentation techniques were applied during training to improve the detector's robustness to variations in lighting, scale, and partial occlusion.

The detector is trained to recognize six semantic classes that correspond to the categories defined in the semantic map. These classes are: upper table corners, table wheels, window corners, ceiling features, wall structures, and wall letters (specifically, the letter *A*). Each class represents a distinct type of structural landmark that can be reliably detected from the robot's camera viewpoint, and that has a corresponding entry in the pre-built map. Most classes being detected by the YOLO detector are shown in Figure 4.2.

Figure 4.2: Example of the detected semantic landmarks used for robot localization.

After dataset preparation and annotation, multiple detector configurations were evaluated to assess the trade-off between detection accuracy and computational cost. The YOLOv8 architecture was chosen as the detector backbone because it is designed specifically for real-time operation, offering a favorable balance between inference speed and detection accuracy compared with other contemporary object detection frameworks. This characteristic is particularly important in the present application, since the neural detector must run on the same embedded hardware as the localization filter, and the inference latency directly affects the end-to-end measurement delay.

Two model sizes were evaluated: YOLOv8s (small) and YOLOv8n (nano). The former provides higher capacity and slightly better accuracy, while the latter is lighter and faster. Testing both

sizes makes it possible to identify the point at which reduced model capacity begins to degrade detection performance in the target environment. In addition to model size, two input resolutions were tested: 960×960 and 768×768 pixels. Resolutions below 768 were not considered because the structural landmarks in the scene, such as table corners and wall features, are relatively small in the image. At lower resolutions, the spatial detail required to discriminate these compact features is lost, and empirical tests showed that the detection precision dropped below acceptable levels for the localization task. The 768×768 resolution was therefore the smallest that retained sufficient feature detail, while 960×960 provided a higher-fidelity alternative for comparison. The results of the four evaluated configurations are summarized in Table 4.1. All configurations were evaluated on a separate video sequence that was not used during training, preventing overlap between training and test samples.

Table 4.1: YOLOv8 detector comparison

Model	mAP_{50}	mAP_{50-95}	Precision	Recall	Latency (ms)
y8s_960	0.914	0.531	0.903	0.868	21.874
y8s_768	0.892	0.495	0.877	0.853	17.632
y8n_960	0.875	0.489	0.908	0.808	16.611
y8n_768	0.871	0.463	0.851	0.858	14.656

The results show that higher-capacity models provide slightly better accuracy, but all evaluated configurations achieve reliable detection performance in the target environment. The y8s_768 configuration was selected for the final system because it provides strong detection performance while maintaining inference rates compatible with real-time onboard operation. The inference latency of 17.6 ms per frame is well within the timing budget of the overall localization pipeline, and the detector outputs approximately 7 to 9 detections per frame under normal operating conditions.

For the final localization system, the detector is deployed through TensorRT to reduce inference latency and improve runtime efficiency on the target GPU hardware. TensorRT converts the model into a highly optimized execution engine for the specific device. During this conversion, TensorRT can fuse layers, select efficient kernels, and use reduced-precision execution when appropriate, which lowers both latency and memory overhead while preserving the learned structure of the network. As a result, the perception stage operates under more predictable timing conditions and with a smaller computational footprint. This is important in the present system because visual detections are later converted into EKF observations, where timing consistency directly affects the quality of the delayed-update mechanism.

Although the detector is optimized with TensorRT, the effective measurement rate still depends on the active perception modules and the scene content. When only the semantic landmark detector is enabled, the pipeline runs at about 28.4 Hz, whereas enabling additional modules such as the box detector and the ArUco detector reduces the effective rate to around 20 Hz due to the increased computational load. For this reason, these perception modules are enabled only when needed, and the EKF is designed to tolerate brief prediction-only intervals between successive visual updates.

At runtime, each RGB frame is processed by the detector to produce a set of candidate detections. Each detection includes the location of a bounding box in the image, an associated semantic class, and a confidence value. These detections are not used directly as state observations. Instead, they provide the initial perceptual evidence from which geometric observations are later constructed using depth information and camera calibration. In this sense, the detector output constitutes the first stage of the observation generation pipeline.

4.4 Observation Generation from RGB-D Data

The visual detections produced by the perception module must be converted into geometric measurements compatible with the localization filter. This is achieved by combining the semantic detections obtained from the RGB images with the corresponding depth information provided by the RGB-D sensor.

For each detected feature, the depth is not estimated from a single pixel at the center of the bounding box, since this is highly sensitive to noise, missing values, and background leakage near the bounding-box boundaries. Instead, depth is extracted from a central crop corresponding to 35% of the bounding box width and height. This value was selected empirically as a practical compromise: smaller crops were more sensitive to depth noise, while larger crops were more likely to include background pixels near the object edges. Invalid measurements and pixels outside the admissible depth range are discarded, and at least 4 valid depth pixels are required for the detection to be accepted. Let $\mathscr{D}$valid denote the set of valid depth samples inside the cropped region. The final depth estimate is computed as the 25th percentile of the valid samples in this cropped region,

$$\hat{d} = Q0.25(\mathscr{D}_{\text{valid}}), \tag{4.2}$$

which biases the estimate toward the nearest visible surface while reducing the influence of noisy measurements and background leakage near the box boundaries.

Given the image coordinates (u,v) of the detection center and the estimated depth $\hat{d}$, the corresponding 3D point in the camera optical frame is obtained by back projection using the calibrated camera intrinsics (f_x, f_y, c_x, c_y):

$$X = \frac{(u-c_x)\hat{d}}{f_x}, \tag{4.3}$$

$$Y = \frac{(v-c_y)\hat{d}}{f_y}, \tag{4.4}$$

$$Z = \hat{d}. \tag{4.5}$$

Using the ROS optical frame convention, where X points to the right, Y points downward, and

Z points forward, the EKF observation is then represented in spherical form

$$\mathbf{z}_i = \begin{bmatrix} r_i \\ \alpha_i \\ \beta_i \end{bmatrix}, \tag{4.6}$$

with

$$r_i = \sqrt{X^2 + Y^2 + Z^2}, \tag{4.7}$$

$$\alpha_i = atan2(X, Z), \tag{4.8}$$

$$\beta_i = atan2(-Y, Z). \tag{4.9}$$

The quantities r_i, α_i, and β_i are respectively the range, azimuth, and elevation of the detected feature relative to the camera.

An uncertainty model is associated with each observation. The observation covariance is modeled as a diagonal matrix in spherical coordinates,

$$\mathbf{R}_i = (\sigma_{r,i}^2, \sigma_{\alpha,i}^2, \sigma_{\beta,i}^2). \tag{4.10}$$

The range uncertainty is modeled as a distance-dependent term, with $\sigma_{r0} = 0.02$ m and $k_r = 0.05$ m, so that farther observations are assigned larger uncertainty.

$$\sigma_{r,i} = \sigma_{r0} + k_r r_i, \tag{4.11}$$

where r_i is the spherical range of the observation. These values were chosen from the sensor characteristics and empirical inspection of the RGB-D measurements, with the constant term accounting for the residual depth noise at short range and the slope term capturing the gradual loss of precision with distance. The angular uncertainties were set to fixed values of 0.06 rad for both azimuth and elevation, based on the camera geometry, and were kept constant because the dominant effect in practice was depth variation rather than angular drift.

4.5 Extended Kalman Filter Formulation

The localization system estimates the robot's pose using an EKF. The state vector is defined as

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}, \tag{4.12}$$

where x_k and y_k are the robot coordinates in the map frame and θ_k is its yaw angle at time step k. The control input is

$$\mathbf{u}_k = \begin{bmatrix} v_k \\ v_{n,k} \\ \omega_k \end{bmatrix}, \tag{4.13}$$

where v_k and $v_{n,k}$ are the forward and lateral velocities, respectively, and ω_k is the angular velocity.

4.5.1 Motion Model and Prediction

Over a sampling interval Δt, the local displacements are

$$d = v_k \Delta t, \tag{4.14}$$

$$d_n = v_{n,k} \Delta t, \tag{4.15}$$

$$\Delta\theta = \omega_k \Delta t. \tag{4.16}$$

For a case without rotation ($\omega_k = 0$), the state prediction is

$$x_{k+1} = x_k + d\cos(\theta_k) - d_n \sin(\theta_k), \tag{4.17}$$

$$y_{k+1} = y_k + d\sin(\theta_k) + d_n \cos(\theta_k), \tag{4.18}$$

$$\theta_{k+1} = \theta_k. \tag{4.19}$$

For a case with rotation ($\omega_k \neq 0$), the local displacements are

$$\Delta x_{\text{local}} = \frac{d\sin(\Delta\theta) - d_n(1-\cos(\Delta\theta))}{\Delta\theta}, \tag{4.20}$$

$$\Delta y_{\text{local}} = \frac{d(1-\cos(\Delta\theta)) + d_n \sin(\Delta\theta)}{\Delta\theta}, \tag{4.21}$$

and rotated into the global frame using the midpoint angle

$$\theta_{\text{mid}} = \theta_k + \frac{\Delta\theta}{2}. \tag{4.22}$$

The predicted state is then

$$x_{k+1} = x_k + \Delta x_{\text{local}} \cos(\theta_{\text{mid}}) - \Delta y_{\text{local}} \sin(\theta_{\text{mid}}), \tag{4.23}$$

$$y_{k+1} = y_k + \Delta x_{\text{local}} \sin(\theta_{\text{mid}}) + \Delta y_{\text{local}} \cos(\theta_{\text{mid}}), \tag{4.24}$$

$$\theta_{k+1} = \theta_k + \Delta\theta. \tag{4.25}$$

The state covariance is propagated through the linearized model as

$$\mathbf{P}_{k+1}^{-} = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^{\top} + \mathbf{W}_k \mathbf{Q} \mathbf{W}_k^{\top}, \tag{4.26}$$

where $\mathbf{F}_k = \partial f / \partial \mathbf{x}$ and $\mathbf{W}_k = \partial f / \partial \mathbf{u}$ are the Jacobians of the motion model with respect to the state and control, respectively, and $\mathbf{Q}$ is the process noise covariance matrix.

4.5.2 Dynamic Camera Pose and Measurement Model

The visual update must account for the fact that the head-mounted camera is not rigidly fixed with respect to the body frame. Its pose varies continuously with the robot configuration, and is therefore computed at runtime from the robot kinematic chain. The camera pose is obtained relative to the robot trunk and composed with the estimated global robot pose, yielding the camera transform at the time of the observation.

Let $\mathbf{p}_c^w$ and $\mathbf{R}_c^w$ denote the camera position and orientation in the world frame, respectively, at the time of the observation. For a mapped landmark j with a known global position $\mathbf{p}_j^w$, the predicted landmark vector in the camera frame is

$$\mathbf{v}_j^c = \left(\mathbf{R}_c^w\right)^{\top} \left(\mathbf{p}_j^w - \mathbf{p}_c^w\right). \tag{4.27}$$

The expected observation associated with landmark j is then written in spherical form as

$$h_j(\mathbf{x}_k^-) = \begin{bmatrix} \|\mathbf{v}_j^c\| \\ atan2(v_{j,x}^c, v_{j,z}^c) \\ atan2(-v_{j,y}^c, v_{j,z}^c) \end{bmatrix}, \tag{4.28}$$

where $\mathbf{x}_k^-$ is the predicted EKF state and the camera pose $(\mathbf{R}_c^w, \mathbf{p}_c^w)$ is evaluated from the robot configuration and state estimation at the same instant. This formulation allows the predicted measurement to reflect both the global robot pose and the time-varying camera pose induced by humanoid motion.

For each associated landmark observation, the innovation is computed as

$$\nu_j = \mathbf{z}_j - h_j(\mathbf{x}_k^-), \tag{4.29}$$

where $\mathbf{z}_j$ is the measured RGB-D observation. The corresponding measurement Jacobian $\mathbf{H}_j$ is obtained analytically from the linearization of the observation model with respect to the planar EKF state. The resulting predicted observation and Jacobian are then used in the EKF update step.

4.5.3 EKF Update and Validation

For each accepted observation–landmark pair, the innovation covariance is

$$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_k^- \mathbf{H}_i^\top + \mathbf{R}_i, \tag{4.30}$$

where $\mathbf{H}_i$ is the measurement Jacobian and $\mathbf{R}_i$ is the observation covariance associated with that detection. The Kalman gain is

$$\mathbf{K}_i = \mathbf{P}_k^- \mathbf{H}_i^\top \mathbf{S}_i^{-1}. \tag{4.31}$$

The state and covariance are updated as

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}_i \nu_i, \tag{4.32}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_i \mathbf{H}_i)\mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_i \mathbf{H}_i)^\top + \mathbf{K}_i \mathbf{R}_i \mathbf{K}_i^\top, \tag{4.33}$$

where the covariance update is computed using the Joseph form to improve numerical stability and preserve the consistency of the covariance matrix during recursive estimation.

The normalized innovation squared (NIS) is used as a final consistency check:

$$\mathrm{NIS}_i = \nu_i^\top \mathbf{S}_i^{-1} \nu_i. \tag{4.34}$$

The update step, therefore, depends on three acceptance conditions. Semantic compatibility restricts each observation to landmarks of the same class, geometric consistency checks whether the range and angle residuals are plausible for the candidate match, and statistical consistency validates the innovation through the NIS gate.

4.6 Data Association and Update Validation

Before an observation can be used by the EKF, it must be associated with the correct landmark in the semantic map. This step is critical, since a wrong correspondence can produce an incorrect state correction.

The association process begins with semantic filtering. Since each detection is assigned a class by the perception module, that class is used to restrict the set of candidate landmarks considered for matching. The system compares the predicted class of the observation against the category of each map feature, and only features with a matching category are retained as candidates. This avoids comparing each observation against the entire map and reduces ambiguity when several landmarks have similar geometric placement.

The remaining candidates are then evaluated geometrically. For each candidate, the expected observation is computed from the current state estimate and the corresponding camera pose, and

compared with the measured observation through the residual

$$\mathbf{e}_{ij} = \mathbf{z}_i - \hat{\mathbf{z}}_{ij}, \tag{4.35}$$

where $\mathbf{z}_i$ is the measured RGB-D observation and $\hat{\mathbf{z}}_{ij}$ is the observation predicted for map landmark j. Candidates are accepted only if they satisfy geometric consistency conditions on the range, azimuth, and elevation residuals, together with an overall score threshold.

The final correspondence is selected using a score-based strategy that favors the candidates with the smallest residual while preventing the same landmark from being assigned to multiple observations in the same update cycle. This strategy is sufficient for the adopted map, where semantic filtering keeps the number of plausible candidates limited. After a correspondence is selected, the observation is subjected to the EKF update and NIS validation described in Section 4.5. Only observations that pass both the association gate and the statistical validation are incorporated into the filter. The overall data association pipeline is illustrated in Figure 4.3.

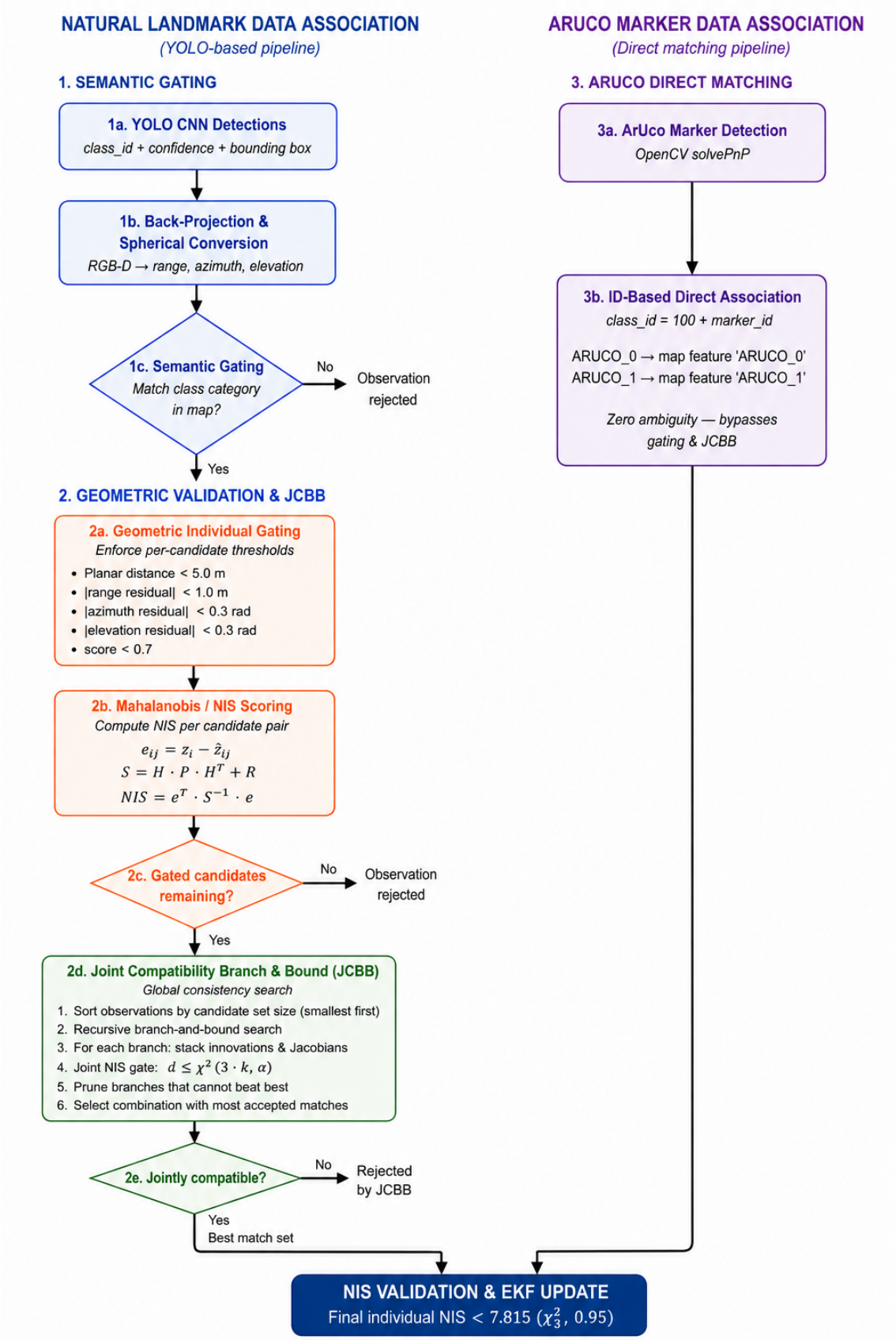

Figure 4.3: Data association pipeline, for both natural detected landmarks and ArUco markers.

4.7 Temporal Robustness and Real-Time Architecture

Having described the individual components of the localization pipeline, this section addresses the system-level concerns that ensure these components operate reliably together in real time on the embedded hardware. It presents the single-node architecture chosen to minimize inter-process latency, characterizes the decoupled prediction–update rates and the end-to-end measurement timing, and details the delayed-update mechanism that maintains temporal consistency despite neural inference delays.

4.7.1 System Architecture and Real-Time Performance

The localization pipeline is implemented as a single C++ ROS 2 node that integrates all stages of the process: image acquisition, YOLO detection, depth estimation, back projection, data association, and EKF prediction and update. This single-node architecture was chosen to eliminate the serialization and transport overhead that would occur if the detector, the observation generator, and the filter were running as separate processes. On the target embedded hardware, the neural detector already consumes the largest share of the computational budget, and any additional inter-process communication latency would further reduce the effective measurement rate.

The system operates with a decoupled prediction–update architecture. The EKF propagates the state at 30.0 Hz using the proprioceptive dead reckoning, producing a continuous pose estimate regardless of whether visual observations are available. This rate was chosen because increasing it further does not provide a proportional accuracy benefit for the robot's motion profile, while lowering it would increase the amount of uncorrected drift accumulated between predictions.

The visual measurement pipeline runs at approximately 28.4 Hz, with 98.8% of processed frames yielding at least one accepted observation. The small gap between the prediction rate and the update rate (1.6 Hz) represents prediction-only steps that ensure smooth pose output even during brief observation gaps. This is the typical rate when only the semantic detector is active; when the box and ArUco detectors are also enabled, the effective measurement rate drops to roughly 20 Hz, which is why those modules are activated only during the phases of the mission that actually require them.

The end-to-end latency from image acquisition to EKF update completion is approximately 91 ms, dominated by the neural detector inference. The EKF itself contributes virtually no wait time. This low computational cost is a defining advantage of the EKF formulation compared to other modern visual localization solutions. The overall architecture ensures that the localization system can operate in real time on the robot's onboard hardware, providing a pose estimate at 30 Hz with minimal latency and predictable timing.

4.7.2 Implementation Details

The node is implemented in C++ and compiled with optimization flags. The EKF prediction step runs on a dedicated timer callback at 30 Hz, while the image acquisition and neural inference are

handled in a separate callback triggered by the RGB camera topic. The depth image is processed in the same thread as the detection, but the backprojection and data association are executed on the detection callback. This separation of the prediction timer from the measurement callback guarantees that the pose estimate is published at a constant rate regardless of the variable latency of the neural detector.

The prediction callback maintains the current state estimate and the associated covariance matrix. It receives the velocity commands from the ‘measured_vel_adapter_node‘ via the topic ‘/booster/move_cmd‘, which carries the forward and lateral velocities v and vn and the heading rate ω. The EKF uses these values as the control inputs for the planar motion model. The timer callback is configured to be non-blocking, so a long-running detection callback does not delay the prediction step.

The visual measurement callback performs the following operations in sequence: it converts the camera image to the format required by the YOLO engine, runs the neural inference, extracts the bounding boxes, and reads the depth values from the corresponding pixels. The depth image is aligned with the RGB image in the camera driver, so no additional registration is needed inside the node. The depth values are averaged over a small window around the center of each bounding box, and the backprojection is performed using the camera intrinsics and the extrinsic transform from the camera frame to the robot base. The resulting 3D points are converted to spherical coordinates and packaged as observation messages.

The observation messages are processed by the data association thread. The data association logic follows the pipeline described in Section 4.6: semantic gating filters candidates by class, geometric consistency checks validate range and angle residuals, Mahalanobis scoring ranks the remaining candidates, and a joint compatibility search ensures globally consistent matching. ArUco markers bypass this pipeline through direct ID-based association. The accepted observations are then passed to the EKF update step.

The EKF update step is executed on the measurement callback, but the prediction timer is not interrupted. The update modifies the current state and covariance, and the updated pose is published immediately on ‘/localization/predicted_pose‘. The update callback is short enough to complete within the 33 ms interval between predictions, so there is no contention for the state buffer.

To support the delayed update mechanism described in the next subsection, the node maintains a circular buffer of past state and covariance snapshots. Each snapshot is stored at the same 30 Hz rate as the prediction step, and the buffer is sized to hold the maximum expected delay. The delay is obtained from the timestamp of the original image, which is propagated through the detection message. When a delayed observation arrives, the filter restores the closest snapshot, applies the update at that past time, and then replays the prediction history forward to the current time. The replay mechanism is a loop over the stored snapshots, applying the same motion model and control inputs that were used during the original prediction. The replay averages 6.6 prediction steps in approximately 0.4 ms, which is negligible compared to the overall pipeline latency.

4.7.3 Temporal Misalignment and Delayed Updates

The visual observations used by the localization system are not available immediately after image acquisition. The delay introduced by detection, inference, depth estimation, and backprojection amounts on average to approximately 91 ms, with a minimum of 74 ms and a maximum of 271 ms (measured during the real-world OptiTrack evaluation from Subsection 4.9.1). This means that, when a measurement reaches the filter, it may already correspond to a past robot pose. If this effect is ignored, the filter may use outdated information to correct the current state, leading to temporal inconsistency.

A first mitigation measure consists of increasing the observation covariance for delayed measurements. The end-to-end delay of each observation is obtained from the original image acquisition timestamp, propagated through the detection message header. The corresponding positional and angular uncertainty is estimated from the current velocity and heading rate, and added to the nominal observation covariance. This reduces the influence of stale observations with large delays without requiring additional computational steps.

A more consistent solution is introduced through delayed updates with rollback and replay. Instead of applying a delayed observation directly to the current state, the filter maintains a history of past states. When a delayed observation arrives, the filter restores the state at the corresponding measurement time, applies the update at that instant, and then replays the prediction history forward again to the present.

This mechanism ensures that a delayed observation is applied at the time at which it was generated, instead of correcting a state that already corresponds to a later robot pose. In this way, the estimator maintains temporal consistency between the predicted state and the observation used to update it.

4.8 ArUco Marker Localization

While the semantic landmark pipeline provides continuous localization across the workspace, there are operational zones, such as picking stations, where higher positional accuracy is required. In these situations, the robot must align precisely with objects or workstations, and the localization error provided by the general semantic features may be insufficient for fine manipulation. To address this, the system incorporates fiducial ArUco markers as a complementary source of observations.

ArUco markers are square binary patterns that can be detected robustly from a wide range of viewpoints and distances. Their positions are known with high precision in the map frame, and their detection yields a direct estimate of the camera-to-marker relative pose. The marker detector runs as a dedicated ROS 2 node that processes the same RGB stream used by the YOLO detector. When a marker is detected, as seen in Figure 4.4, the node publishes an observation message containing the marker identifier and the estimated relative pose, which is then converted into the same spherical range–azimuth–elevation representation used by the semantic observations.

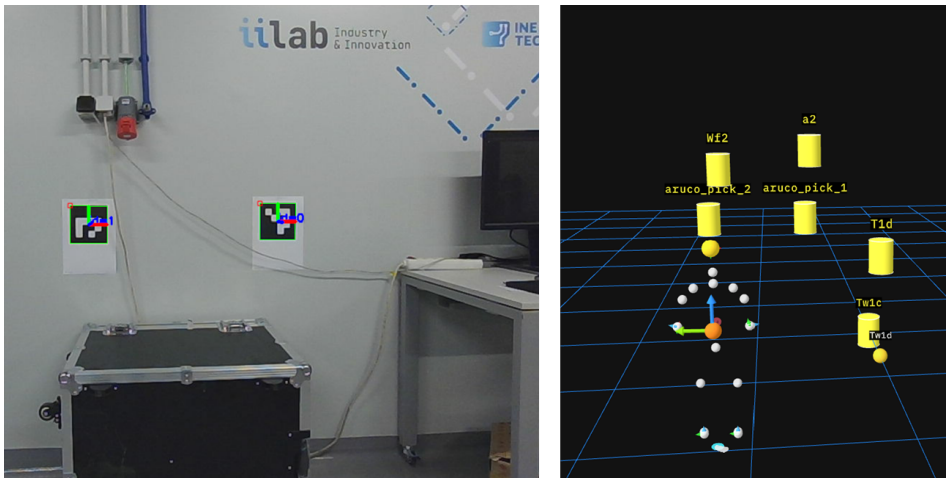

Figure 4.4: ArUco marker deployment. (Left) Physical marker in the laboratory environment. (Right) Corresponding marker position in the semantic map frame.

The ArUco observations are integrated into the EKF through the same update mechanism described in Section 4.5. Because the markers are treated as a special category in the semantic map, their observations bypass the general semantic detector and are matched directly to their known map positions. The ArUco observations provide stronger geometric constraints than natural semantic features, owing to the precise geometry of the marker pattern and the robustness of the detection algorithm. When the robot is in a picking zone and one or more markers are visible, the ArUco observations dominate the update and reduce the localization error to the centimeter level. When no markers are visible, the system falls back to the general semantic landmark pipeline, maintaining continuous localization across the workspace.

The combination of the two observation sources allows the system to balance generality and precision. The semantic landmarks provide coverage and continuity during navigation, while the ArUco markers provide the accuracy required for fine alignment tasks. This dual-mode behavior is transparent to the higher-level control modules, which receive a single pose estimate at 30 Hz regardless of which observation source is active.

4.9 Experimental Results

The proposed localization pipeline was validated through two complementary experimental campaigns with distinct objectives. A real-world evaluation through comparison with ground truth from motion capture quantifies the accuracy and repeatability of the complete system under real operating conditions. The MATLAB simulation study, by contrast, does not aim to replicate real-world performance; its purpose is to isolate and characterize the intrinsic behavior of the EKF under controlled, idealized conditions, by removing perception and mapping errors, to understand how landmark density, motion-to-measurement noise balance, and fiducial marker precision independently affect filter performance. The performance of the YOLO landmark detector, which supplies the visual observations used in both campaigns, is characterized separately in Section 4.3.

4.9.1 Real-World Evaluation with OptiTrack

To evaluate the EKF localization system under real operating conditions, the humanoid robot was deployed in the target indoor environment and tracked using an OptiTrack motion capture system as ground truth. Ten independent runs were conducted, with the robot navigating a trajectory of approximately 8 m at an average speed of 0.15 m/s. The EKF was initialized with the same initial pose in all runs, and the robot followed the same approximate path.

The aggregated results over the ten runs are summarized in Table 4.2. The EKF achieved a mean XY position error of 14.4 ± 1.5 cm and a mean absolute yaw error of $4.9 \pm 1.0^\circ$ across all runs. In comparison, dead reckoning based on the same odometry inputs produced a mean XY error of 63.1 ± 10.2 cm, illustrating the substantial benefit of fusing visual landmark observations for planar drift reduction.

Table 4.2: Real-world evaluation results (mean $\pm$ std over ten runs)

Metric	EKF	Dead Reckoning
Mean XY absolute error [cm]	14.4 ± 1.5	63.1 ± 10.2

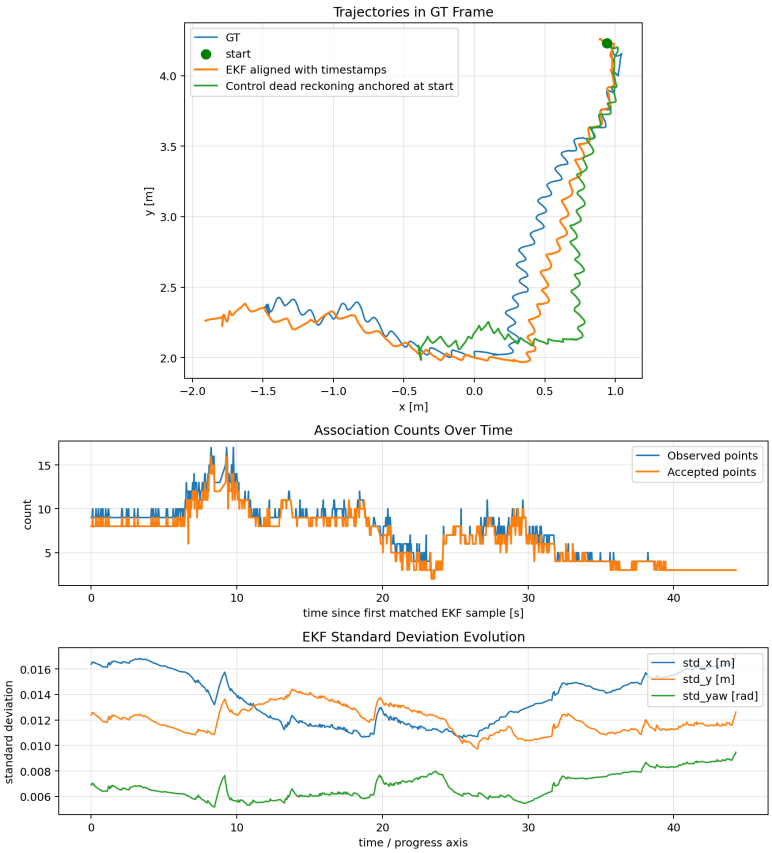

Figure 4.5: Top row shows the ground-truth trajectory (blue), the EKF estimate (orange), and the dead-reckoning trajectory (green) in the global frame. Bottom row shows the association counts and the evolution of the EKF standard deviation for position and heading.

The system operates with a decoupled prediction–update architecture: the EKF propagates state at 30.0Hz using odometry, while visual landmark observations are fused at 28.4Hz. The measured end-to-end latency from image acquisition to EKF update completion is 91 ± 6 ms, with the EKF itself contributing negligible overhead. This latency refers to the full observation pipeline, including image processing, neural inference, depth extraction, backprojection, and data association; it is therefore not directly equivalent to the number of replayed prediction steps. Because visual observations arrive with a delay, they correspond to a past robot pose rather than the current state. To preserve temporal consistency, the filter rolls back to the closest stored snapshot and replays an integer number of stored prediction steps forward to the present. Across the recorded experiments, the replay length averaged 6.6 steps and required only 0.4 ms, which is negligible compared with the total observation latency.

The dominant error sources in this evaluation relate to the environment map and the ground-truth assumption. The semantic map was constructed by manual measurement with a tape measure, relying on assumptions of structural parallelism and orthogonality that may not hold exactly in the physical environment. While the simulation studies discussed below achieve sub-5-cm accuracy under idealized conditions, the real-world error of 14.4 cm remains within practical navigation requirements for the intended application.

4.9.2 MATLAB Simulation Study

To complement the real-world observations, controlled simulation experiments were conducted in MATLAB to isolate and characterize the behavior of the EKF from perception-related uncertainty. The purpose of these experiments is not to validate the real-world performance of the system—which is addressed by the OptiTrack evaluation—but rather to understand how the EKF behaves under idealized conditions, to identify the conditions under which it converges and remains accurate, and to characterize its sensitivity to key design parameters.

The analysis assumes: (i) perfect feature detection and data association whenever visible, abstracting away detection errors; (ii) a priori known feature positions from offline mapping; (iii) controlled visibility conditions to maintain a constant number of observed features; and (iv) additive Gaussian noise with known covariance to emulate RGB-D measurement uncertainty. Unless otherwise specified, the EKF noise matrices follow the International System of Units, with linear dimensions in meters and angular measures in radians. The baseline motion noise is $\mathbf{Q} = (0.03^2, 0.03^2, 0.02^2)$ and the measurement noise parameters are $\sigma_{r0} = 0.02$ m, $k_r = 0.01$, $\sigma_\alpha = \sigma_\beta = 0.1$ rad. Each configuration is evaluated over $N = 200$ Monte Carlo trials with a sampling interval of $\Delta t = 40$ ms, which provides a stable statistical estimate while keeping the simulation workload manageable.

4.9.2.1 Effect of Feature Density

The goal of this study is to determine how many simultaneously observed landmarks are sufficient for the EKF to converge and maintain accurate localization, and to quantify the marginal benefit

of each additional observation. The robot follows a fixed circular trajectory while $n \in \{1, \ldots, 10\}$ features are uniformly distributed in the environment. All features remain visible at an average distance of approximately 3.2 m, so that performance variations depend only on feature count. The robot is initialized with a pose and orientation error, and the EKF is evaluated over 200 Monte Carlo trials per configuration, with the results shown in Figure 4.6.

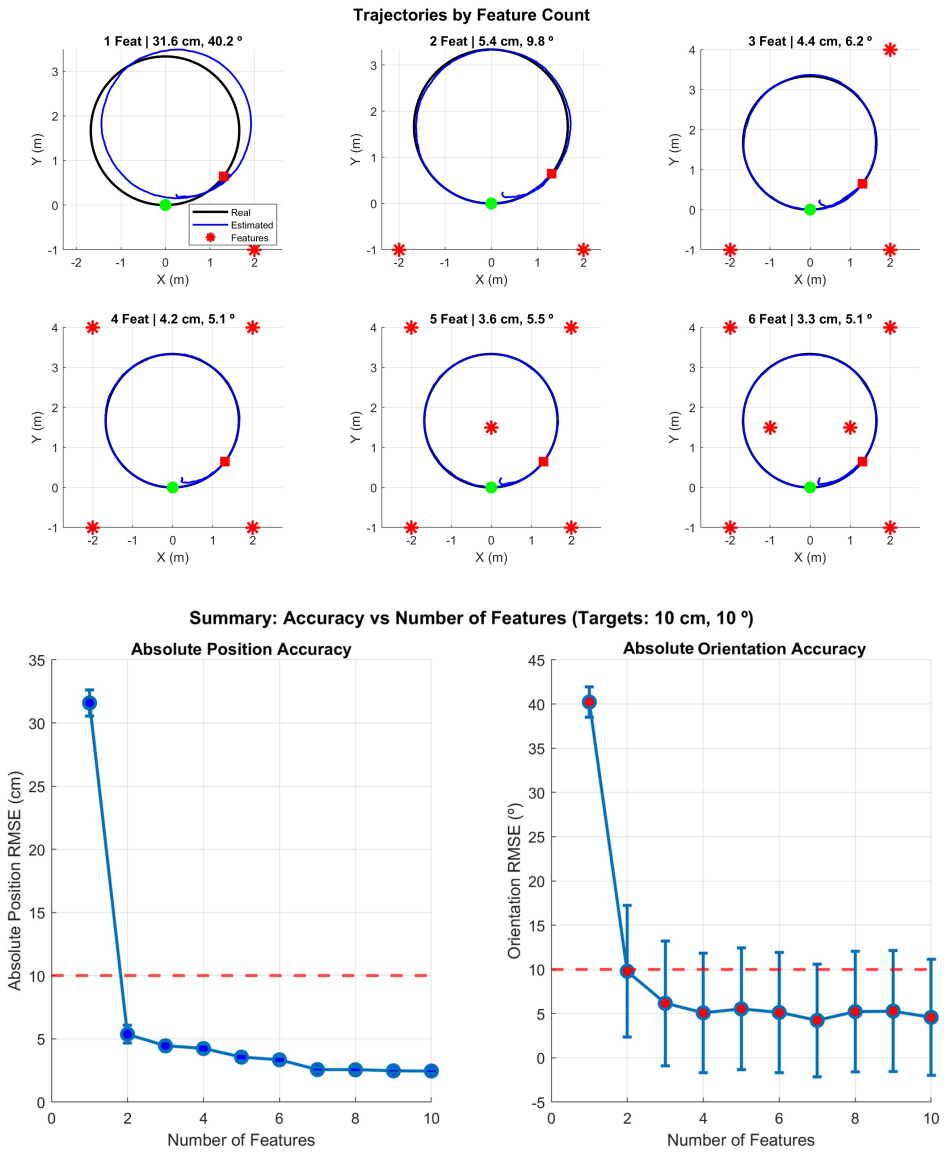

Figure 4.6: Top row shows real and estimated circular trajectories of the first 6 feature counts, with features (red stars) uniformly distributed. Bottom row quantifies absolute position RMSE (left) and orientation RMSE (right) as a function of the number of observed features.

Position Root Mean Square Error (RMSE) decreases from 31.6 cm (single feature) to 5.4 cm (two features), saturating near 2.5 cm for $n \geq 7$. The largest gain occurs with the second feature, since a single landmark provides insufficient geometric constraints to fully constrain the planar pose. Beyond $n = 2$, improvements diminish: the third feature reduces RMSE to 4.5 cm, with

only marginal gains thereafter. Orientation error shows similar behavior, dropping from 40.2° to 9.8° and stabilizing near 5°.

These results indicate that the EKF achieves near-optimal performance with two to three well-distributed features, and that additional observations beyond this range provide diminishing returns. In practice, this means that the localization pipeline is robust to intermittent detection failures, as it does not depend on observing a large number of landmarks simultaneously to maintain accurate pose estimates.

4.9.2.2 Effect of Motion-to-Measurement Noise Ratio

This study characterizes how the balance between motion model trust and measurement trust governs the filter's convergence rate, steady-state accuracy, and noise rejection. The trade-off is analyzed by varying the motion noise covariance $\mathbf{Q}$ while keeping the measurement noise $\mathbf{R}$ fixed. Two configurations are tested: $\mathbf{Q}_{\text{low}} = 0.01\,\mathbf{Q}_{\text{baseline}}$ (high trust in the motion model relative to measurements) and $\mathbf{Q}_{\text{high}} = 100\,\mathbf{Q}_{\text{baseline}}$ (low trust in the motion model, i.e., high trust in measurements). The robot follows a circular trajectory while observing 4 fixed features, starting from an initial pose error of $(0.25, 0.25)$ m and 90°.

Convergence behavior differs markedly between the two configurations. With low $\mathbf{Q}/\mathbf{R}$, the filter converges smoothly but slowly, requiring approximately 4.0 s (100 iterations at $v = 0.5$ m/s) to reach steady state, with position error decaying from roughly 25 cm to below 5 cm. This configuration acts as a strong temporal smoother dominated by the motion model. With high $\mathbf{Q}/\mathbf{R}$, convergence is faster (approximately 2.0 s to reach the same error band) but at the cost of noticeably noisier estimates, as the filter more aggressively follows each individual measurement update.

Steady-state statistics confirm this trade-off. Mean absolute position and orientation errors are 2.3 cm and 0.63° for $\mathbf{Q}_{\text{low}}$, versus 3.8 cm and 0.76° for $\mathbf{Q}_{\text{high}}$. The appropriate $\mathbf{Q}/\mathbf{R}$ ratio therefore depends on the desired balance between convergence speed and steady-state accuracy: a lower ratio favors smooth, accurate estimates after convergence, while a higher ratio provides faster recovery from large initial pose errors at the cost of increased steady-state noise.

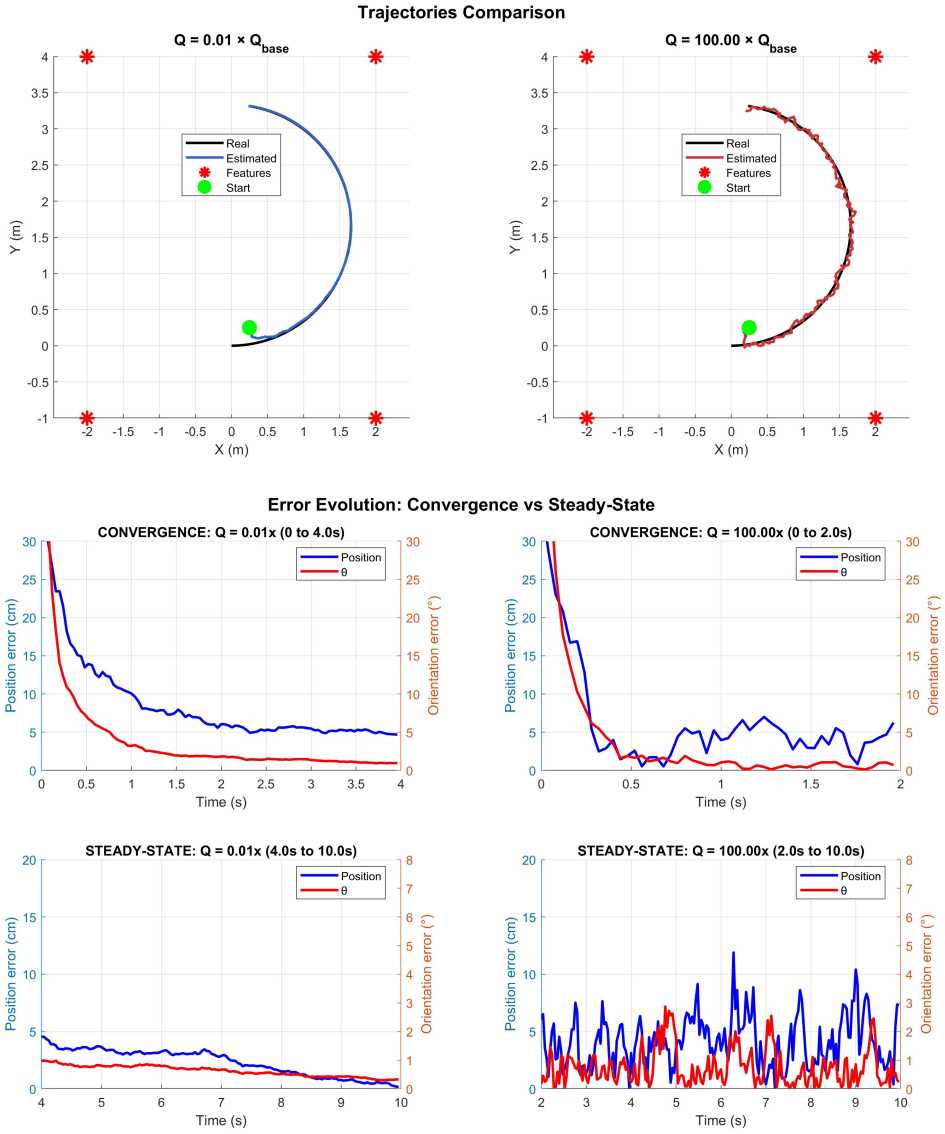

Figure 4.7: Top row compares estimated and real trajectories for low and high motion model noise settings, with natural features shown as red stars. Bottom row shows the corresponding evolution of absolute position error (blue) and orientation error (red) during convergence and steady-state intervals.

4.9.2.3 Effect of Measurement Noise on Fiducial Marker Localization

This study characterizes the accuracy limits and noise sensitivity of the EKF when using fiducial markers (ArUco) as the sole observation source. The scenario is designed to mimic a docking or picking station approach, where the robot must navigate to a waypoint and achieve centimeter-level alignment. The goal is not to validate the ArUco system as deployed on the real robot, but rather to characterize the intrinsic accuracy limits of fiducial-based EKF localization and to identify the noise tolerance envelope within which the filter can deliver reliable high-precision corrections.

Two fiducial markers are placed symmetrically at a height of 2.0 m, separated by approximately 1.0 m, and the target docking waypoint is located between them at a distance of approximately 0.5 m. This geometry is chosen because it provides an approximately 90° viewing angle between the markers from the waypoint, which is near-optimal for triangulation.

The robot is initialized with a random pose error and follows a converging trajectory toward the docking waypoint. The experiment is repeated for 200 trajectories at each of six measurement noise levels, parameterized by a noise scale α that multiplies the baseline measurement covariance. This sweep is used to characterize the system's sensitivity to progressively noisier fiducial observations. The success criterion is defined as a final position error below 1.0 cm and an orientation error below 1.0°.

Figure 4.8 shows the convergence behavior of the Monte Carlo trajectories for selected noise levels and quantifies the performance metrics across the full α sweep. At the lowest noise level ($\alpha = 0.01$), the success rate is 99.5%, with a median maximum absolute position error of 0.6 cm and a median maximum absolute orientation error of 0.2°. As the noise scale increases, the success rate degrades gradually: 98.0% at $\alpha = 0.05$, 90.5% at $\alpha = 0.15$, and 86.0% at $\alpha = 0.25$. Beyond $\alpha = 0.50$, performance drops sharply, with success rates falling to 49.5% and 11.0% at $\alpha = 1.00$. The median maximum absolute position error increases from 0.6 cm to 11.6 cm across the range, while the orientation error increases from 0.2° to 4.1°.

These results reveal that fiducial-based EKF localization can achieve sub-centimeter accuracy under favorable noise conditions, but that performance is sensitive to large measurement errors, degrading sharply beyond a noise scale of approximately $\alpha = 0.25$. This degradation identifies a practical design threshold: to reliably achieve sub-centimeter accuracy at docking stations, the fiducial marker detection must operate with an effective measurement noise satisfying $\alpha \leq 0.15$, where the measurement covariance is $\mathbf{R} = ((\alpha\sigma_{r,i})^2, (\alpha\sigma_{\alpha,i})^2, (\alpha\sigma_{\beta,i})^2)$ and $\sigma_{r,i}$ follows the distance-dependent model defined in Section 4.4. In this setup, $\alpha \leq 0.15$ emerges as a practical threshold for maintaining sub-centimeter performance, rather than as a theoretical limit. In practice, this sensitivity is mitigated by two factors: ArUco markers provide geometrically constrained, low-noise observations by design, and the system combines fiducial observations with the general semantic landmark pipeline, maintaining continuous localization even when fiducial observations are temporarily compromised or unavailable. The characterization of this noise envelope informs the operational conditions under which ArUco-based corrections can be trusted for precision alignment tasks.

The real-world evaluation confirms that the proposed localization pipeline meets the accuracy, repeatability, and real-time performance requirements for humanoid robot navigation and manipulation in structured indoor environments. The controlled simulation studies complement these results by characterizing the EKF's behavior under idealized conditions, clarifying the roles of feature density, noise tuning, and fiducial marker geometry in determining localization performance, and providing practical guidance for system configuration and filter parameter selection. A Video of the robot navigating and localizing in the laboratory is included in Appendix 7.2.

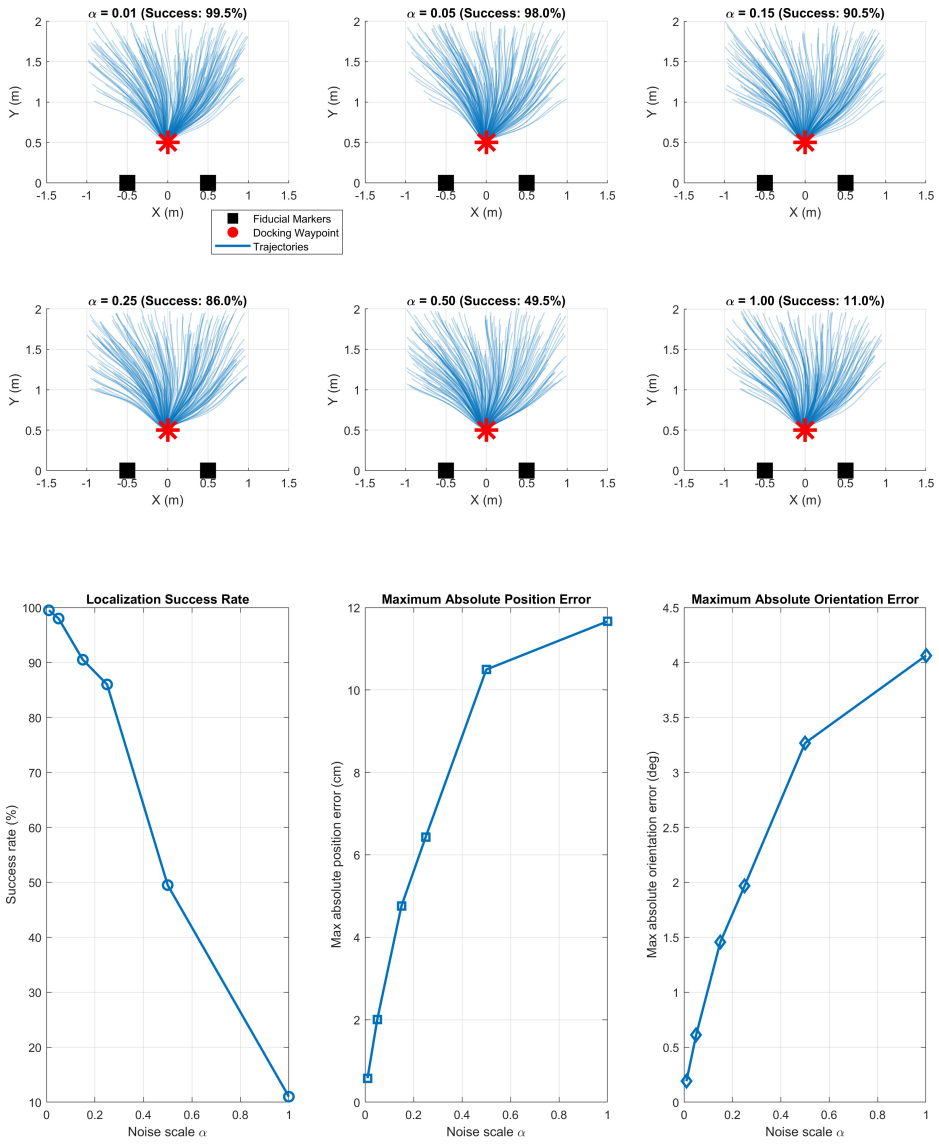

Figure 4.8: Top row shows the spatial convergence of the Monte Carlo trajectories for selected measurement noise scales α, with the two fiducial markers placed symmetrically at a height of 2.0 m. Bottom row quantifies the success rate of achieving sub-centimeter and sub-degree error (left), the median maximum absolute position error (center), and the median maximum absolute orientation error (right) as a function of the noise scale α.

4.10 Adaptation to a New Environment: Robotics National Festival 2026

During the development of this work, an opportunity arose to validate the localization framework outside the laboratory setting. At the Robotics National Festival 2026, held in Barcelos, Portugal, the system was deployed on a standard RoboCup soccer field. This deployment provided a practical test of the framework's adaptability to a new environment with distinct visual characteristics, lighting conditions, and spatial geometry.

The target environment was a regulation RoboCup soccer field located indoors under mixed lighting conditions. The playing surface consists of a grass carpet with white boundaries and field lines, goal structures at both ends, and standardized ground markings including penalty areas and center points. Because the field geometry is fully specified by competition regulations, the semantic map could be constructed directly from the documented dimensions.

The adaptation of the system to the soccer field required two principal environment-specific components: a new semantic map and a dedicated YOLO detector for the field landmarks.

The semantic map was constructed from the documented field geometry. The mapped landmarks, shown in Figure 4.9 corresponded to four semantic categories: goal corners (*gc*), field line corners (*c*), as well as "T" and "+" shaped intersections (*t* and *p*, respectively). The complete map contained approximately thirty landmarks distributed across the field.

Figure 4.9: YOLO landmark detection on the RoboCup soccer field.

The visual detector was trained from scratch using approximately thirty RGB images acquired directly from the robot-mounted camera at the venue. This small dataset reflects the practical time constraints of an event setting. Due to the limited training set, the resulting model was not as heavily optimized as the laboratory detector; however, the nature of the soccer field environment

compensated for this. The field presents a large number of simultaneously visible landmarks, and the EKF requires at least two well-distributed features to maintain accurate localization, as characterized in previous sections. In practice, the detector observed the majority of visible landmarks in each frame, with only occasional omissions at longer ranges, showing that even a smaller dataset can prove to be sufficient for localization, depending on how visible the features being detected are.

Once configured, the robot localized itself continuously while walking across the field. The EKF maintained a consistent pose estimate while the robot navigated the environment using the path tracking module described in Chapter 5.

This deployment demonstrated that the proposed localization framework is not restricted to the laboratory conditions under which it was originally developed. The successful operation on a standard RoboCup field, achieved with minimal reconfiguration and a lightweight detector, supports the claim that the separation between learned semantic perception and model-based geometric estimation enables rapid adaptation to new structured environments. Representative video and image material of the robot navigating the soccer field are provided in Appendix 7.3.

Chapter 5

Navigation, Perception and Manipulation

This chapter describes the motion, perception, and manipulation modules that enable the humanoid robot to perform the target box-handling tasks. The system architecture and end-to-end pipeline are described in Chapter 3. This chapter focuses on the individual components and their integration. The pipeline combines three core capabilities: map-frame waypoint navigation, visual object detection and alignment, and dual-arm grasping. A high-level mission executor orchestrates these components into a repeatable picking sequence. The chapter begins with the path tracking module, which guides the robot along predefined waypoints using the EKF pose estimate. It then describes the visual object detection pipeline that identifies the target box and estimates its 3D position. The visual alignment and picking approach strategy, which combines reactive visual servoing with pose-based waypoint navigation, is presented next. The grasping and manipulation module, including the dual-arm grasp sequence and the custom L-shaped end-effectors, is then detailed. The chapter also presents the mission execution and orchestration layer that sequences the navigation, perception, and manipulation modules into a complete picking cycle. Finally, the quantitative results of the end-to-end picking experiments are presented.

5.1 Navigation and Path Tracking

The path tracking module guides the robot along a predefined sequence of waypoints in the map frame. It is implemented as a ROS 2 node that consumes the localized pose estimate from the EKF described in Chapter 4 and publishes motion commands to the locomotion interface. The tracker treats the waypoint list as a sequence of line segments, each one defined by a start waypoint and a goal waypoint, and advances from one segment to the next as soon as the current goal is satisfied.

The controller is organized as a finite-state machine with four states: IDLE, APPROACH, TRACK, and FINAL, as shown in Figure 5.1. This structure separates the problem into three distinct behaviors: waiting for a mission, converging to the path line, following the line with steady

motion, and finally converging precisely to the segment goal and preferred yaw. The main idea is to keep the lateral correction explicit while keeping the forward motion simple and predictable.

When a segment is already satisfied, the tracker immediately advances to the next waypoint pair. The initial state is selected according to the current pose, the lateral deviation from the line, and the distance to the goal, so the controller can resume from the most appropriate phase instead of always restarting from APPROACH.

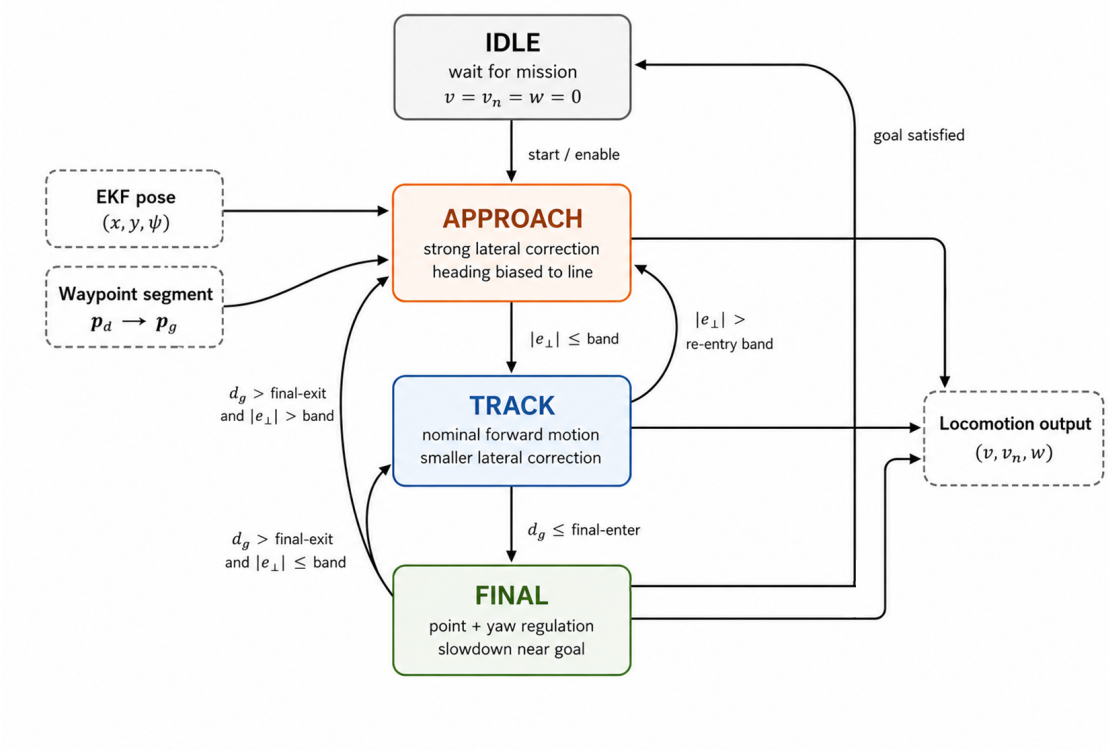

Figure 5.1: Finite-state machine for path tracking.

For a given segment, the start waypoint $\mathbf{p}_d$ defines the segment direction and the goal waypoint $\mathbf{p}_g$ defines the segment endpoint. The unit tangent and normal vectors are

$$\mathbf{t} = \frac{\mathbf{p}_g - \mathbf{p}_d}{\|\mathbf{p}_g - \mathbf{p}_d\|} \qquad \mathbf{n} = \begin{bmatrix} -t_y \\ t_x \end{bmatrix}. \tag{5.1}$$

To make the geometry explicit, the robot position $\mathbf{p}$ can be orthogonally projected onto the segment line as

$$\mathbf{p}_{\text{proj}} = \mathbf{p}_d + \big((\mathbf{p} - \mathbf{p}_d)^T \mathbf{t}\big)\mathbf{t}. \tag{5.2}$$

The signed lateral deviation from the line is then measured by

$$e_\perp = (\mathbf{p} - \mathbf{p}_{\text{proj}})^T \mathbf{n}. \tag{5.3}$$

In the APPROACH state, the robot is considered to be far from the line and the controller uses a stronger lateral correction to drive it back toward the segment. The world-frame command is built from a nominal forward speed and a lateral term:

$$u_t = v_{\text{nom}}, \qquad u_\perp = -k_\perp^{\text{app}} e_\perp \,, \tag{5.4}$$

$$\mathbf{v}_w = u_t \mathbf{t} + u_\perp \mathbf{n} \,. \tag{5.5}$$

The nominal speed is fixed by configuration, so the robot advances even when the longitudinal error changes. The heading reference in this state is also biased toward the line:

$$\theta_{\text{ref}} = \theta_{\text{line}} - \arctan(k_c e_\perp) \,, \tag{5.6}$$

where $\theta_{\text{line}} = \text{atan2}(t_y, t_x)$. The angular command is then

$$w = k_\theta^{\text{app}} \,\text{wrap}(\theta_{\text{ref}} - \psi) \,. \tag{5.7}$$

The APPROACH state is used when the robot is too far from the line for stable line following. The transition into TRACK occurs when $|e_\perp|$ enters the approach band, and the transition back to APPROACH occurs if the deviation grows again beyond the re-entry band.

In the TRACK state, the translational law is the same in structure, but the lateral gain is smaller:

$$u_t = v_{\text{nom}}, \qquad u_\perp = -k_\perp^{\text{trk}} e_\perp \,, \tag{5.8}$$

$$\mathbf{v}_w = u_t \mathbf{t} + u_\perp \mathbf{n} \,. \tag{5.9}$$

The difference is in the heading law. Here the robot simply aligns with the segment direction:

$$\theta_{\text{ref}} = \theta_{\text{line}} \,, \qquad w = k_\theta^{\text{app}} \,\text{wrap}(\theta_{\text{ref}} - \psi) \,. \tag{5.10}$$

This state is used when the robot is already close enough to the line that the controller can follow it steadily with small corrections. It is the steady line-tracking regime of the module.

The world-frame command is converted to the robot body frame using the current yaw estimate:

$$\begin{bmatrix} v \\ v_n \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \mathbf{v}_w \,. \tag{5.11}$$

This command is then scaled so that the direction is preserved while the per-axis limits are respected. In the real-robot configuration, the forward and backward limits are asymmetric, which avoids overcommanding the locomotion interface while keeping the motion smooth. A separate breakaway shaping stage enforces minimum non-zero commands when the robot is starting from rest or reversing direction.

The FINAL state is used when the robot reaches the vicinity of the segment goal. At this point the tracker abandons line following and switches to direct point-and-yaw regulation. The position error is expressed in the robot body frame as

$$e_{\text{forward}} = \cos\psi\,(x_g - x) + \sin\psi\,(y_g - y) \tag{5.12}$$

$$e_{\text{lateral}} = -\sin\psi\,(x_g - x) + \cos\psi\,(y_g - y)\,. \tag{5.13}$$

The translational command is then computed as

$$u_t = v_{\text{nom}}, \qquad u_\perp = -k_\perp^{\text{trk}} e_\perp\,, \tag{5.14}$$

$$\mathbf{v}_w = u_t\mathbf{t} + u_\perp\mathbf{n}\,. \tag{5.15}$$

where the caps are reduced further inside a slowdown (0.10 m) zone near the goal. If the final waypoint defines a desired yaw, that yaw is used as the heading reference; otherwise, the segment heading is reused. The angular command is again a proportional wrapped error term with a separate final-state gain. The segment is considered complete only when the position error and the yaw error are both within their tolerances. If the robot leaves the final zone again, the state machine falls back to TRACK or APPROACH depending on the lateral deviation, which keeps the controller hysteretic and prevents oscillatory switching near the goal.

The state machine is initialized from the current pose and waypoint segment. If the robot is already near the goal and the yaw is acceptable, the tracker remains idle; otherwise, it starts in APPROACH, TRACK, or FINAL depending on the lateral offset and the distance to the goal. During execution, the node also stops motion if the pose estimate becomes stale or if an alignment task activates concurrently, which avoids conflicting command sources. The controller parameters used throughout the experimental evaluation are summarized in Table 5.1.

Table 5.1: Path tracker limits and parameters

Parameter	Value
Maximum forward speed	0.16 m/s
Maximum backward speed	0.18 m/s
Maximum lateral speed	0.22 m/s
Maximum angular speed	0.35 rad/s
Nominal line speed	0.16 m/s
Goal position tolerance	0.05 m
Goal yaw tolerance	0.10 rad
Final slowdown distance	0.10 m

The motion limits were selected conservatively to keep the robot stable and safe during real-robot operation, given the humanoid gait and the limited workspace around the picking stations. The forward, backward, and lateral speed caps can be increased, but doing so would reduce the margin available for safe stopping and would also make the localization loop more demanding, since the robot would travel farther between visual updates. In other words, higher speeds would increase the temporal gap between predictions and corrections, which may degrade pose accuracy if the perception pipeline cannot keep up. The goal position and yaw tolerances were set to be sufficient for coarse waypoint navigation, rather than for final grasp alignment. Fine positioning is handled later in the picking pipeline, where visual alignment and ArUco-based corrections provide higher precision near the picking station. As a result, the navigation tolerances can remain relatively permissive without affecting the final pick, although they could be reduced in the future if a more precise waypoint-following behavior becomes necessary.

5.2 Visual Object Detection

The object detection module identifies the target box, seen in Figure 5.2 and estimates its 3D position relative to the robot. It is implemented as a standalone ROS 2 node that runs a YOLO v8 object detector compiled to a TensorRT engine.

Figure 5.2: Target boxes used in the pick-and-place experiments.

The detector is trained to recognize a single class representing examples of multiple box faces, which correspond to the visible faces of the target box from different viewpoints. The node synchronizes RGB and depth frames from the camera, runs inference on the RGB image, and outputs detections with bounding box coordinates. For each detection, the node computes a representative depth by sampling the central region of the bounding box in the aligned depth image. This depth estimate, together with the bounding box center and the camera intrinsics, is used to backproject the detection into a 3D point in the camera frame.

The 3D point is then transformed into the robot Trunk frame using the kinematic chain (camera - head_point - Trunk), yielding the target position in robot-centric coordinates. In addition, the node transforms the Trunk-frame position into the global map frame by compounding it with the current EKF robot pose. This map-frame position is used to validate that the detected box lies within the expected picking station, defined by the area between the two ArUco markers loaded from the semantic map. Detections that fall outside this region are rejected.

Figure 5.3: YOLO box detection when multiple candidates are available.

When multiple box faces are visible, the node selects the candidate face with the largest horizontal width in the image. This heuristic prioritizes the face that is most frontal to the camera, which generally provides the most reliable depth estimate and the best geometry for grasping. If only one face is visible, it is selected automatically. The candidate is published as a compact JavaScript Object Notation (JSON) message containing the 3D position in the Trunk frame, the bounding box width, and the estimated depth. The candidate stream is used by the alignment controller and the mission executor to drive the visual servoing phase.

5.3 Visual Alignment and Picking Approach

The visual alignment and picking approach module combines reactive visual servoing with pose-based waypoint navigation to position the robot at the grasping station. The strategy is deliberately divided into two phases: a lateral alignment driven purely by visual feedback, followed by a forward approach guided by the estimated robot pose in the map frame, as illustrated in Figure 5.4.

When the robot is positioned near the picking station (approximately 1 m from the target), the box is typically visible and the candidate stream is active. The alignment controller uses the lateral

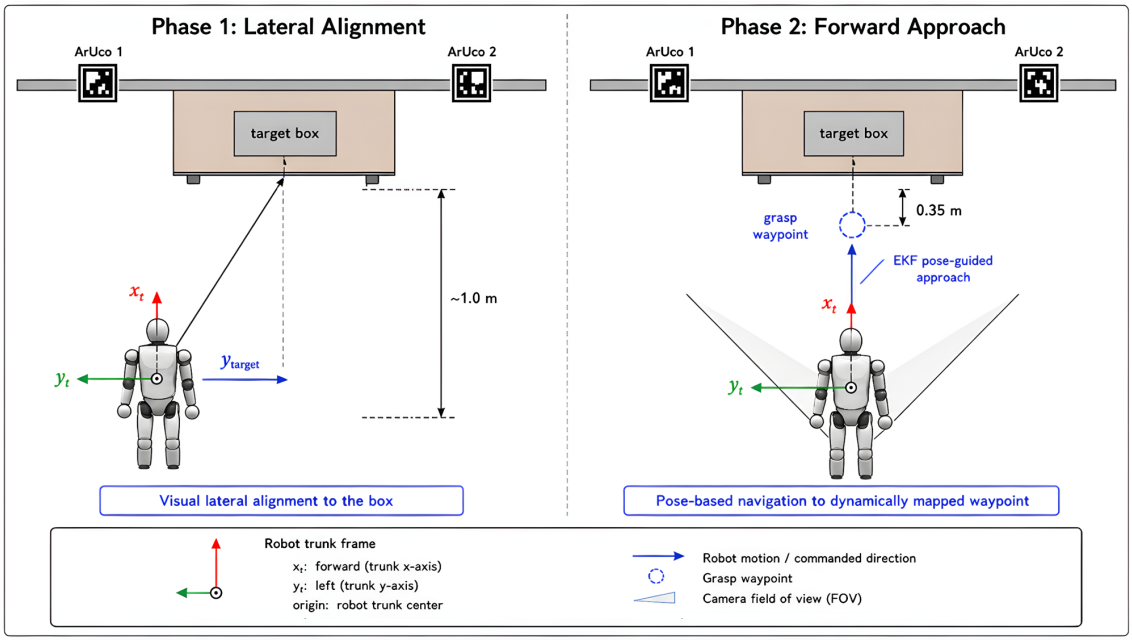

Figure 5.4: Two-phase picking strategy: lateral alignment, followed by forward approach.

component (y) of the target position in the Trunk frame to center the robot laterally with respect to the box. A proportional controller generates a lateral velocity command:

$$v_n = -k_p \, y_{\text{target}} \,, \tag{5.16}$$

The alignment is considered complete when $|y_{\text{target}}| < 0.02$ m. During alignment, only lateral correction is performed. This decouples the lateral alignment from the forward approach, simplifying the control and reducing the risk of collision. The alignment controller also monitors the target stream for staleness. If the box is not detected for more than 5 s, the alignment is aborted to prevent uncontrolled motion.

No rotational correction is applied during the visual alignment phase. The path tracker, described in Section 5.1, is configured with a target yaw on the final waypoint of the approach segment. By design, the robot arrives at the picking station already facing the correct direction, so the heading is aligned during the navigation phase rather than through the visual servoing loop. This decoupling keeps the lateral alignment controller simple and avoids the risk of oscillatory behavior that would arise from simultaneously correcting position and orientation using the same visual feedback.

Once the lateral alignment is complete, the system captures the current EKF pose and the last valid target position in the Trunk frame. The box position is transformed into the map frame by compounding the robot pose with the relative target position:

$$x_{\text{box,map}} = x_{\text{robot}} + x_{\text{target}}\cos(\theta) - y_{\text{target}}\sin(\theta)\,, \tag{5.17}$$

$$y_{\text{box,map}} = y_{\text{robot}} + x_{\text{target}}\sin(\theta) + y_{\text{target}}\cos(\theta)\,. \tag{5.18}$$

An approach waypoint is then computed at a fixed distance (0.35 m by default) from the box position in the direction of the robot's current heading. This waypoint is published to the path tracker, which guides the robot to the final grasping position using the EKF pose alone. This distance was chosen so that the robot stops at a practical grasping position, centered enough for the L-shaped end-effectors to reach the box reliably without overcommitting too close to the station. The camera may lose sight of the box during the forward approach, but because the waypoint is now defined in the absolute map frame, the robot can complete the approach without visual feedback.

This two-phase strategy combines the reactivity and precision of visual servoing for lateral correction with the robustness of pose-based navigation for the final forward motion. The separation is motivated by the limited field of view of the head-mounted camera, at close range, the box may move out of view or become partially occluded, making pure visual servoing unreliable for the forward axis. The EKF pose, continuously corrected by landmark observations and ArUco markers when available, provides a stable reference for the final approach.

The picking-stage parameters, summarized in Table 5.2, were chosen more tightly than the global navigation values because the robot is already close to the box and can exploit the higher-precision ArUco-based corrections available at the picking station. Both the lateral alignment and approach tolerances were set to 0.02 m so that the robot is positioned precisely in front of the box, which is a tighter requirement than in global waypoint navigation. The maximum lateral speed was kept conservative to reduce overshoot while moving. The retreat distance of 1.0 m ensures that the robot clears the local picking area before path tracking resumes, bringing it back to a safer region of the workspace where collisions are less likely. The alignment timeout is used as a safety bound rather than a performance target, and can be adjusted depending on the mission setup.

Table 5.2: Picking-stage parameters

Parameter	Value
Maximum lateral alignment speed	0.10 m/s
Lateral alignment tolerance	0.02 m
Approach waypoint tolerance	0.02 m
Alignment timeout	2.0 s
Approach waypoint distance	0.35 m
Retreat distance	1.0 m

5.4 Grasping and Manipulation

The grasping module executes the physical pick operation using the robot's dual arms. It is implemented as a ROS 2 package that provides a command interface for the dual-arm grasp executor, together with utility nodes for monitoring and safety.

The arm grasp executor accepts high-level commands (reach, close, up, home) and translates them into dual-arm end-effector hardcoded poses via the robot SDK, as shown in Figure 5.5. The poses are defined in the Trunk frame and are configured for the target box dimensions. The four commands form a complete grasp sequence: reach extends the arms forward to a pre-grasp position, clearing the robot's body; close moves the arms inward to grasp the box; up lifts the box to a transport height; and home returns the arms to the lowered resting position.

Figure 5.5: Dual-arm grasp sequence positions: reach (top left), close (top right), up (bottom left), and home (bottom right).

The use of hardcoded poses is a deliberate design choice motivated by a limitation observed during development: when commanding the arms to a given 6-DoF target position, the onboard kinematic monitor reports final end-effector coordinates that differ from the commanded values, particularly along the lateral (y) axis that controls the distance between the two arms. This discrepancy was verified both through the kinematic monitor and physically with a tape measure on the robot, confirming that the actual arm positions do not match the commanded poses. Because of this mismatch, the arm executor cannot reliably place the end-effectors at dynamically computed positions. The box detector does estimate the physical width of the detected box face (through depth-based backprojection), which in principle would allow the grasp aperture to be adjusted per-box. However, since the arm motion does not precisely reach the commanded lateral coordinates, this dynamic approach was not viable, and the grasp sequence was instead tuned to a fixed set of poses that work reliably for the target box geometry.

The robot does not perform a force-closing grasp in the traditional sense. The stock end-effectors are replaced by custom L-shaped extensions that slide under the box edges, seen in Figure 5.6, allowing the box to be lifted from below without requiring the arm motors to exert sustained gripping force. The arms move to the close position and then lift upward; the box rests on the horizontal ledges of the L-shaped pieces, with the vertical flanges providing lateral constraint. This passive lifting strategy avoids loading the small arm actuators with the full box weight during transport.

Figure 5.6: End-effector extensions positioned beneath the box edges.

The executor accepts commands via a ROS 2 topic and publishes status feedback. Commands

can be sent as simple strings (reach, close, up, home) or as JSON payloads that override the default poses and timing for a specific command. The executor supports both simultaneous dual-arm motion (default) and sequential single-arm motion. The robot must be in the appropriate operating mode before arm commands are accepted; the executor does not manage the robot mode, leaving this responsibility to the higher-level mission controller.

A monitoring node tracks arm effort and can trigger a safety return to home when both arms hit sustained torque peaks, indicating unexpected contact or grasp failure. The default positions, durations, and safety parameters are stored in a Yet Another Markup Language (YAML) configuration file and can be overridden at runtime via ROS 2 parameters.

5.5 Mission Execution and Orchestration

The mission execution and orchestration layer sequences the navigation, perception, and manipulation modules into a complete picking pipeline. This layer is implemented as two components: a mission executor that runs the state machine of an individual picking cycle, and an orchestrator that manages the queue of picking stations and provides the operator interface.

The mission executor implements a finite-state machine that drives the robot through a single picking cycle. The states are defined as follows:

- **FOLLOW_PATH:** The robot follows the static waypoints from the TOML file until the final waypoint near the picking station is reached.
- **WAIT_FOR_BOX:** The executor waits for the box detector to publish a valid candidate with a 3D target position.
- **MAP_BOX:** A computational step with no motion. The system captures the current EKF pose and the last valid target position, computes the box position in the map frame, and generates a dynamic grasp waypoint at a fixed distance from the box.
- **ARM_REACH:** The arms extend to the pre-grasp position.
- **ALIGN_LATERAL:** The robot performs visual lateral alignment using the y component of the target position in the Trunk frame, as described in Section 5.3.
- **APPROACH_FORWARD:** The robot moves to the grasp waypoint using the EKF pose only, without visual feedback.
- **ARM_CLOSE:** The arms close to grasp the box.
- **ARM_UP:** The arms lift the box to the transport height.
- **RETREAT:** The robot moves backward for a fixed duration to clear the picking station.

Transitions between states are triggered by completion conditions or timeouts. The executor publishes continuous status feedback and supports an abort command that stops the current mission and returns the robot to a safe state.

The executor monitors several failure conditions throughout the cycle. If the path tracker enters IDLE without having reached the final waypoint, the mission is aborted. If the box candidate is not received for more than a configurable timeout (default 2.0 s) during ALIGN_LATERAL, the mission is aborted. If the EKF pose is not received for more than a timeout during APPROACH_FORWARD, or if the robot does not converge to the grasp waypoint within a time limit, the mission is aborted. If the arm grasp executor reports a command failure, the mission is also aborted. These timeouts and safety mechanisms ensure that the robot does not continue an uncontrolled motion when a critical sensor or actuator fails.

The orchestrator maintains a queue of picking stations and initiates missions sequentially. A complete end-to-end mission execution is shown in Appendix 7.5. It exposes an operator interface through a ROS 2 service that accepts commands such as start and abort. The orchestrator transitions between three states: STANDBY, where the robot is idle and waiting for a command; NAVIGATING, where a mission is in progress; and ALL_DONE, when all stations in the queue have been processed. When a mission reports completion, the orchestrator automatically launches the next station if one remains, or transitions to ALL_DONE if the queue is empty.

5.6 Picking Experimental Evaluation

To validate the integrated navigation, perception, and manipulation pipeline, the robot was asked to execute repeated picking missions under different initial positions relative to the target box. A total of 20 trials were carried out, with the robot starting from slightly different map-frame positions while the box remained fixed at the picking station, in order for the alignment and approach requirements to be varied. A trial was counted as successful only when the robot completed the full sequence and grasped the box correctly. This number was considered sufficient to capture the main behavior of the system without spending excessive time on repetitive trials, given that the available robot start positions relative to the box were limited and would otherwise begin to repeat.

The overall success rate was 16 out of 20 trials. The 4 failures all followed the same pattern: when the box was located on the robot's right-hand side, the visual alignment module tended to overcorrect laterally, which occasionally prevented the robot from arriving at a stable grasping pose. In the remaining trials, the mission executor completed the sequence as expected.

These results indicate that the integrated system is able to complete the task reliably in most cases, while also exposing a clear failure mode in the lateral alignment stage. Since the issue appears consistently on one side of the robot, it suggests that the alignment controller may benefit from side-dependent tuning or from a small compensation term in future work.

Some representative successful and failed runs are included in Appendix 7.4, which provides a more intuitive view of the system behavior.

While 80% indicates that the integrated pipeline is functional, this success rate would be insufficient for industrial deployment, where near-perfect reliability is expected. Also, the sample size of 20 trials, although sufficient to identify the dominant failure mode, is still very small and limits the statistical confidence of the measured success rate. The lateral alignment corrections necessary to address the overcorrection behavior were identified during the experimental evaluation but could not be implemented within the scope of this work, due to time constraints. Nevertheless, the experiments were valuable in pinpointing the specific phase where the system requires further refinement.

Chapter 6

Conclusions

6.1 Summary and Retrospective

This dissertation addressed the problem of autonomous box manipulation with a humanoid robot in a structured indoor environment, motivated by the growing need for flexible automation solutions in logistics warehouses. The work focused on the Booster T1 humanoid platform and developed an integrated perception, localization, navigation, and manipulation pipeline that relies exclusively on the robot's onboard RGB-D camera as the exteroceptive sensor.

The starting point was the observation, documented in Chapter 1, that fixed automation systems are efficient for high-volume, low-variety operations but exhibit rigidity when layouts or product types change. Humanoid robots emerge as a promising alternative, combining bipedal locomotion with dual-arm manipulation on a single platform that can operate in spaces originally designed for humans. However, their effective deployment is not yet feasible; it requires solving fundamental challenges of perception, localization, and precise actuation, particularly when all sensing and processing must be performed onboard without relying on external infrastructure.

From this context, the dissertation proposed a modular software architecture organized into ROS 2 nodes, distributed across two onboard processing boards: a perception board responsible for neural inference and state estimation, and task orchestration, and a motion board responsible for path tracking, alignment, and arm execution (Chapter 3). This separation was not merely a software engineering choice but a response to the latency and computational constraints of the embedded hardware: YOLO inference and the EKF consume GPU-intensive resources, while gait control and arm motion demand low-latency response and direct access to the robot's control interface.

The technical heart of the system lies in the visual localization solution presented in Chapter 4. An EKF fuses proprioceptive dead reckoning with observations of natural semantic landmarks detected by a YOLO convolutional neural network trained specifically for the laboratory environment, complemented by fiducial ArUco markers in precision-critical zones. The system operates in real time at 30 Hz, incorporating a delayed-update mechanism with rollback and replay to maintain temporal consistency despite neural inference latency (91 ms). The choice of the EKF

over other localization algorithms reflected a deliberate decision to favor computational efficiency and real-time operation over higher precision, dependent on higher computational costs and dense environment reconstruction.

The navigation and manipulation modules (Chapter 5) were designed to operate with the EKF. The path tracker consumes the pose estimate to follow predefined waypoint sequences, while the picking module uses both the box detection and robot position to correctly approach the box for grasping. Grasping is performed by a predefined arm sequence (reach, close, up, home) using passive L-shaped end-effectors that lift the box from below, avoiding the need for sustained gripping force. All of these components are orchestrated by a finite-state machine that sequences the mission phases as desired by the operator or predefined mission script.

A distinctive characteristic of this work is that the components are not merely independent modules placed in series, but an integrated system in which each element compensates for the limitations of the others. The path tracker depends on the robustness of the EKF to avoid drift accumulation; the visual alignment compensates for the high tolerance of the path tracker with corrections based on ArUco markers that provide centimeter-level precision where natural semantic landmarks are insufficient for fine alignment. A video demonstrating the full integrated mission is available in Appendix 7.5.

6.2 Main Contributions

The main contributions of this work are:

1. **An integrated perception-driven pipeline** for box pick-and-place with a humanoid robot, combining visual localization, object detection, path tracking, and dual-arm manipulation operating exclusively with onboard processing.

2. **A real-time semantic landmark-based localization system** using an EKF, specifically designed to address the challenges of humanoid platforms: the measurement model accounts for body configuration and head motion, the filter operates within the computational constraints of embedded humanoid hardware by fusing lightweight CNN-based detections with proprioceptive dead reckoning, and the delayed-update mechanism with rollback and replay compensates for the neural inference latency that is particularly critical on resource-constrained platforms.

3. **A distributed two-board software architecture** that separates GPU-accelerated neural inference and state estimation on the perception board from low-latency motion control on the motion board, communicating via onboard ROS 2 topics, enabling real-time operation within the computational constraints of embedded humanoid hardware.

4. **Experimental validation** of the localization system against a motion capture ground truth system, demonstrating substantial reduction in planar pose error compared to dead reckoning alone, and of the complete picking pipeline through repeated end-to-end trials with quantitative phase-level success metrics.

5. **Controlled Monte Carlo simulation studies** characterizing the intrinsic sensitivity of the EKF to landmark density, motion-to-measurement noise ratio, and fiducial marker geometry, providing practical guidance for filter parameter selection and system configuration.

6. **A platform-agnostic design** that was built from the ground up on generic robot interfaces: omnidirectional velocity commands, standard kinematic transforms, and onboard RGB-D camera streams, and arm-manipulation through coordinates, rather than relying on vendor-specific middleware beyond the low-level locomotion SDK. This makes the pipeline inherently portable to other humanoid platforms with equivalent sensing and actuation capabilities.

6.3 Discussion of Results

This section interprets the experimental results presented in the previous chapters, placing them in the context of the system objectives and identifying their practical implications. The discussion begins with the localization performance, comparing the EKF solution against dead reckoning and ground truth on the real robot. It then examines the insights gained from the MATLAB simulation studies, which isolate the intrinsic behavior of the filter from perception uncertainties. Finally, the picking pipeline performance is analyzed, including the identification of failure modes and their root causes.

6.3.1 Localization Performance

The most rigorous evaluation of the localization system was performed through ten independent trials on the real robot, with ground truth provided by an OptiTrack motion capture system (Section 4.9.1). The robot traversed a trajectory of approximately 8 m at an average speed of 0.15 m/s. Over the course of each run, the dead reckoning solution, which integrates the same proprioceptive odometry used by the EKF prediction step without any exteroceptive correction, accumulated drift, reaching a mean position error of 63.1 ± 10.2 cm by the end of the trajectory. In contrast, the EKF, fusing the same odometry with visual landmark observations, bounded the error to 14.4 ± 1.5 cm. This comparison illustrates not merely a quantitative improvement, but a qualitative difference in behavior: without visual corrections, the odometry error grows unbounded with distance travelled, whereas the EKF continuously resets the accumulated drift through landmark updates, maintaining a stable pose estimate regardless of trajectory length.

Part of the residual error of 14 cm can be attributed to uncertainties in the semantic map, which was constructed by manual measurement with a tape measure and laser under structural parallelism and orthogonality assumptions that do not hold exactly in the physical environment.

In practice, the 14 cm error is adequate for navigation between picking stations, while the ArUco markers, when visible near the station, provide the precision required for fine alignment needed for grasping.

From a temporal perspective, the system maintains a decoupled prediction–update architecture: the EKF propagates state at 30.0 Hz using proprioceptive odometry, while visual observations are fused at approximately 28.4 Hz. The end-to-end latency from image acquisition to EKF update completion is 91 ± 6 ms, dominated by the neural detector inference. The EKF itself contributes negligible overhead. The delayed-update mechanism recovers this latency by rolling back to the image acquisition timestamp, followed by replay of an average of 6.6 prediction steps in approximately 0.4 ms, a marginal computational cost that ensures temporal consistency without degrading the pose publication rate. Video recordings of the robot localizing and navigating in the laboratory environment are provided in Appendix 7.2.

6.3.2 Insights from MATLAB Simulations

The Monte Carlo simulation studies (Section 4.9) isolated the EKF behavior from perception uncertainties, allowing characterization of its sensitivity to design parameters under idealized conditions.

In the landmark density study, position RMSE decreased from 31.6 cm (single landmark) to 5.4 cm (two landmarks), saturating near 2.5 cm for $n \geq 7$. The most significant gain occurs with the second landmark, since a single feature provides insufficient geometric constraints to fully constrain the planar pose. This result has important practical implications: the localization pipeline is robust to intermittent detection failures, as it does not depend on observing a large number of landmarks simultaneously to maintain accurate pose estimates.

The Q/R ratio study revealed the classical trade-off between convergence speed and steady-state stability. Configurations with low Q/R (high trust in the motion model) converge smoothly but slowly, compared to configurations with high Q/R (low trust in motion), which converge faster but with noisier steady-state estimates. These results directly inform the selection of filter noise parameters on the real robot, where a compromise was chosen to allow rapid recovery from large initial errors without excessively sacrificing trajectory smoothness as well as future tuning for better results.

Finally, the fiducial marker noise sensitivity study characterized the tolerance envelope within which the EKF can deliver precise enough corrections at stations. The success rate (defined as final error < 1 cm and orientation < 1) is 99.5% at $\alpha = 0.01$, degrading gradually. The practical threshold of $\alpha \leq 0.15$ defines the operational conditions under which ArUco-based corrections can be trusted for precision alignment. The simulation, therefore, bridges the gap between filter theory and physical deployment: it tells the designer how "clean" the observations must be, and the designer then selects marker geometry and station layout to meet that requirement. In practice, ArUco markers provide geometrically constrained, low-noise observations by design, and the system combines them with the general semantic landmark pipeline, maintaining continuous localization even when fiducial observations are temporarily compromised or unavailable.

6.3.3 Picking Performance

The alignment and picking pipeline was evaluated through 20 repeated picking trials, with the robot starting from different positions relative to the box at the picking station (Section 5.6). The overall success rate was 16 out of 20 trials (80%).

Analysis of the 4 failed trials reveals a systematic pattern: three failures occurred when the box was located on the robot's right-hand side, where the visual alignment module tended to over-correct laterally, occasionally preventing the robot from reaching a stable grasping pose. This unilateral behavior suggests that the alignment controller would benefit from side-dependent tuning or a compensation term, as discussed in Section 5.3. The fourth failure did not follow this specific pattern.

The 80% success rate is interpreted as evidence that the integrated system is capable of completing the task reliably in most cases, but also that the visual alignment phase currently represents the primary robustness bottleneck of the pipeline. The path tracker, box detection, forward approach, and grasp sequence operated consistently in the successful trials. The practical message from these results is that the sensor fusion and orchestration architecture works, but that the precision of the final visual servoing requires additional refinement to achieve consistent bilateral robustness. Representative picking trials, including both successful and failed runs, are shown in Appendix 7.4.

6.3.4 Adaptability to New Environments

A particularly meaningful complementary test of the localization framework occurred during the Robotics National Festival, in Barcelos (Section 4.10). The localization system was adapted and deployed on a standard RoboCup soccer field, an environment with distinct visual characteristics, lighting conditions, and spatial geometry from the laboratory.

The adaptation required two essential components: a new semantic map constructed from the documented field dimensions, containing approximately 30 landmarks in four semantic categories (goal corners, line corners, T-shaped intersections, and + intersections); and a YOLO detector trained from approximately 30 RGB images acquired directly from the robot's camera at the venue.

The robot localized itself continuously while walking across the field, demonstrating that the proposed framework is not restricted to the laboratory conditions under which it was originally developed. This experience validates the central hypothesis that the separation between learned semantic perception and model-based geometric estimation enables rapid adaptation to new structured environments, provided the map and detector are reconfigured. It constitutes a strong argument of generalizability: the system is not merely a laboratory demonstrator, but an architecture that can be reconfigured for new scenarios in a timeframe compatible with practical applications. A video of the robot operating on the RoboCup field is available in Appendix 7.3.

6.4 Limitations

The system presented in this dissertation was conceived as an end-to-end integration effort, deliberately scoped to address the core challenges of visual localization and task sequencing on a humanoid platform. The operating assumptions discussed below define the boundaries of this scope. Rather than representing shortcomings of the proposed architecture, they correspond to deliberate design decisions that made the problem tractable within the time constraints of this dissertation while preserving the modularity and extensibility of the developed framework.

One important limitation concerns the operating environment. The proposed system assumes a structured and static workspace, without considering the presence of dynamic obstacles, moving humans, or other robots. Consequently, aspects such as reactive collision avoidance, human–robot interaction, and navigation in shared environments fall outside the scope of the current implementation.

The manipulation pipeline also relies on assumptions regarding the target object. The box detection and alignment stages assume that the object is positioned with its front face parallel to the picking station and perpendicular to the robot's approach direction. Although this assumption proved sufficient for the experimental validation presented in this dissertation, boxes placed with significant rotations or tilted orientations may prevent the predefined grasp sequence from succeeding reliably.

Another limitation is associated with the localization framework. The proposed EKF-based solution requires an *a priori* semantic map containing the positions of both semantic landmarks and ArUco markers. The system therefore assumes that this information is available before operation and does not perform simultaneous mapping, automatic map generation, or online map updates during deployment.

The overall software architecture was designed for a single humanoid robot operating independently. While the modular ROS 2 implementation could serve as the basis for larger robotic systems, multi-robot operation would require additional mechanisms for task allocation, collision avoidance, communication management, and distributed state estimation, none of which were addressed within the scope of this work.

Finally, the proposed solution remains constrained by the computational resources available onboard the Booster T1 platform. When only the semantic landmark detector is active, the localization pipeline operates at approximately 28.4 Hz. However, enabling additional perception modules, such as box detection and ArUco detection, reduces the effective processing rate to approximately 20 Hz. This limitation is mitigated by activating each perception module only during the mission phases in which it is required. Nevertheless, future extensions involving computationally demanding capabilities, such as 6-DoF pose estimation, vision-based quality inspection, or dynamic obstacle avoidance, will likely require either further algorithmic optimization or additional computational resources.

Overall, these limitations do not compromise the validity of the proposed system within the objectives defined for this dissertation. Instead, they establish a well-defined operating scope

while identifying the main directions in which the architecture can naturally evolve towards more autonomous and robust humanoid manipulation systems.

6.5 Future Work

Several future research directions naturally emerge from both the limitations discussed above and the experience gained during the implementation and experimental validation of the proposed system. Overall, future work should focus on increasing the robustness, autonomy, and applicability of the developed framework while preserving the modular architecture presented throughout this dissertation.

One important direction concerns the systematic optimization of the numerous parameters used throughout the perception, localization, navigation, and manipulation pipeline. Many of these parameters were selected through empirical tuning and qualitative assessment rather than systematic sensitivity studies or formal optimization procedures. A more comprehensive parameter optimization process, guided by quantitative performance metrics, would likely improve the overall accuracy and robustness of the complete system.

A natural and simple extension of the current implementation is the addition of the placement stage, completing the full pick-and-place cycle. Since the locomotion, localization, and alignment infrastructure has already been developed, implementing the place operation would only require reversing the current arm motion sequence used for picking, in order to lower the transported box onto a support surface before retracting the manipulators.

The perception pipeline could also be extended beyond its current functionality by incorporating additional vision-based capabilities. The neural detection framework already used for box identification could support post-pick quality inspection tasks, such as product presence verification, item counting, and packaging anomaly detection. In parallel, additional object classes could be incorporated to detect humans, other robots, and dynamic obstacles, enabling perception-driven collision avoidance in shared environments. These extensions would broaden the applicability of the existing perception framework while fully exploiting the onboard RGB-D sensing infrastructure, and without increasing computational load.

Another important direction concerns the manipulation stage itself. The current implementation relies on predefined grasp coordinates during arm execution. Incorporating real-time visual servoing would allow the robot to continuously adjust its grasp pose according to the observed position of the box, increasing robustness against residual positioning errors and small variations in object placement. However, enabling this capability in the current system would first require addressing the arm coordinate mismatch described in the limitations above. Until the arm motion can be calibrated or augmented with closed-loop feedback on the actual end-effector position, adjusting grasp poses in real time would propagate the same positioning uncertainty into the grasp, without guaranteeing improved accuracy.

The perception system could benefit from active head control during task execution. Rather than maintaining a fixed head orientation, the robot could continuously track the target box throughout the approach and grasping phases, preserving visual contact and providing continuous feedback for more accurate alignment corrections.

The current manipulation pipeline estimates only the three-dimensional position of the target object. Replacing this simplified representation with a full 6-DoF object pose estimation framework would enable reliable manipulation of arbitrarily oriented boxes as well as objects with more complex geometries, significantly increasing the flexibility of the system.

Another important direction concerns the navigation framework. The current implementation performs path tracking over predefined waypoint sequences manually specified in the mission configuration. While this approach is well-suited for structured and repetitive warehouse layouts, it does not perform online path planning or trajectory generation. Future work should therefore investigate the integration of autonomous path planning algorithms capable of generating collision-free trajectories online according to the current environment and mission objectives. Combined with reactive obstacle avoidance, this would allow the robot to operate more autonomously in dynamic warehouse environments.

Finally, the proposed architecture could be extended towards multi-robot operation. Supporting coordinated fleets of humanoid robots and autonomous mobile robots would require mechanisms for distributed mapping, task allocation, inter-robot coordination, and collision-free motion planning, allowing the proposed framework to address more complex logistics scenarios.

In summary, future developments should focus on improving the robustness of the existing pipeline, extending its manipulation capabilities, and broadening its applicability to more realistic warehouse environments. These research directions build directly upon the architecture developed in this dissertation and represent a natural progression towards more capable humanoid robotic systems for logistics applications.

References

[1] Nils Boysen, René de Koster, and Felix Weidinger. "Warehousing in the e-commerce era: A survey". In: *European Journal of Operational Research* 277.2 (2019), pp. 396–411. DOI: 10.1016/j.ejor.2018.08.023.

[2] Rodrigo Bernardo, João M. C. Sousa, and Paulo J. S. Gonçalves. "Survey on robotic systems for internal logistics". In: *Journal of Manufacturing Systems* 65 (2022), pp. 339–350. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2022.09.014.

[3] Mikell P. Groover. *Automation, Production Systems, and Computer-Integrated Manufacturing*. 5th. Pearson, 2020.

[4] László Monostori. "Cyber-physical production systems: Roots, expectations and R&D challenges". In: *Procedia CIRP* 17 (2014), pp. 9–13.

[5] Yuchuang Tong, Huaping Liu, and Zhengtao Zhang. "Advancements in Humanoid Robots: A Comprehensive Review and Future Prospects". In: *IEEE/CAA Journal of Automatica Sinica* 11.2 (2024), pp. 301–328. DOI: 10.1109/JAS.2023.124140.

[6] Valerio Villani et al. "Survey on human-robot collaboration in industrial settings". In: *Mechatronics* 55 (2018), pp. 248–266.

[7] Shuuji Kajita et al. *Introduction to Humanoid Robotics*. Springer, 2014.

[8] Z. Gu et al. "Humanoid Locomotion and Manipulation: Current Progress and Challenges in Control, Planning, and Learning". In: *arXiv preprint arXiv:2501.02116* (Jan. 2025). DOI: 10.48550/arXiv.2501.02116.

[9] Interesting Engineering and The Robot Report. *Digit Humanoid: Commercial Deployment at Scale in Logistics*. Nov. 2025. URL: https://interestingengineering.com/ai-robotics/digit-humanoid-robot-moves-100000-totes.

[10] Enkikai and Elaraf. *Figure 02 Humanoid Robot: Commercial Deployment and Manufacturing Scale-Up*. Oct. 2025. URL: https://www.figure.ai/news/helix-logistics.

[11] Robot Magazine and AICerts. *Tesla Optimus: Humanoid Robot Internal Factory Deployment and Manufacturing Strategy*. Dec. 2025. URL: https://www.robot-magazine.fr/en/tesla-robotics-toward-a-new-humanoid-robot-industry/.

[12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005. ISBN: 9780262201629.

[13] Ricardo B. Sousa, Héber M. Sobreira, and António Paulo Moreira. *A systematic literature review on long-term localization and mapping for mobile robots*. 2023.

[14] Carolina Rutili de Lima et al. "Humanoid Robot Motion Planning Approaches: a Survey". In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 110 (2 June 2024), pp. 1–22. ISSN: 15730409. DOI: 10.1007/s10846-024-02117-z.

[15] Stylianos Piperakis et al. “Humanoid Robot Dense RGB-D SLAM for Embedded Devices”. In: *3rd Workshop on Legged Robots, IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[16] Tim Bailey et al. “Consistency of the EKF-SLAM Algorithm”. In: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 3562–3568. DOI: 10.1109/IROS.2006.281644.

[17] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890. DOI: 10.1109/TRO.2021.3075644.

[18] Abhishek Roychoudhury et al. “Perception for Humanoid Robots”. In: *Current Robotics Reports* 4 (2023), pp. 127–140. DOI: 10.1007/s43154-023-00110-y.

[19] H. Lim, S. Y. Hwang, and S. H. Lee. “Real-time single camera SLAM using fiducial markers”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2009, pp. 1554–1559. URL: https://ieeexplore.ieee.org/document/5333379/.

[20] Guohua Lei et al. “An indoor localization method for humanoid robot based on artificial landmark”. In: *Proceedings - 5th International Conference on Instrumentation and Measurement, Computer, Communication, and Control (IMCCC)*. Institute of Electrical and Electronics Engineers Inc., Feb. 2016, pp. 1854–1857. ISBN: 9781467377232. DOI: 10.1109/IMCCC.2015.394.

[21] Sourav Raxit, Simant Bahadur Singh, and Abdullah Al Redwan Newaz. “YoloTag: Vision-based Robust UAV Navigation with Fiducial Markers”. In: *IEEE International Workshop on Robot and Human Communication (RO-MAN)* (Oct. 2024), pp. 311–316. DOI: 10.1109/RO-MAN60168.2024.10731319.

[22] Changhao Chen et al. “A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence”. In: *arXiv preprint arXiv:2006.12567* (June 2020). DOI: 10.48550/arXiv.2006.12567.

[23] Paul-Edouard Sarlin et al. “SuperGlue: Learning Feature Matching with Graph Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020, pp. 4938–4947. DOI: 10.1109/CVPR42600.2020.00499.

[24] Carolina C. Lopes et al. “Multi-Neural Network Localisation System with Regression and Classification on Football Autonomous Robots”. In: *AI* 6.2 (2025). DOI: 10.3390/ai6020027.

[25] Ross Hartley et al. “Contact-Aided Invariant Extended Kalman Filtering for Robot State Estimation”. In: *The International Journal of Robotics Research* 39.4 (2020), pp. 402–430. DOI: 10.1177/0278364919894385.

[26] Filipe F. Diniz, Guilherme A. Borges, et al. “Trajectory of Humanoid Robots with the Unscented Kalman Filter”. In: *2024 Brazilian Symposium on Robotics (SBR) and 2024 Workshop on Robotics in Education (WRE)*. IEEE. 2024.

[27] Peter Corke. *Robotics, Vision and Control*. 2nd. Springer, 2021.

[28] José Neira and Juan D. Tardós. “Data Association in Stochastic Mapping Using the Joint Compatibility Test”. In: *IEEE Transactions on Robotics and Automation* 17.6 (2001), pp. 890–897. DOI: 10.1109/70.976019.

[29] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press, 1988. ISBN: 978-0120797608.

[30] Manh Duong Phung et al. "Localization of Networked Robot Systems Subject to Random Delay and Packet Loss". In: *arXiv preprint arXiv:1612.01034* (2016). DOI: 10.1109/AIM.2013.6584297.

[31] Cesar Cádena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[32] Ryan M. Eustice et al. "Exactly Sparse Delayed-State Filters for View-Based SLAM". In: *IEEE Transactions on Robotics* 22.6 (2006), pp. 1100–1114. DOI: 10.1109/TRO.2006.886264.

[33] Tara Mina, Lindsey Marinello, and John Christian. "Remarks on Stochastic Cloning and Delayed-State Filtering". In: *arXiv preprint arXiv:2508.21260* (2025). DOI: 10.48550/arXiv.2508.21260.

[34] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[35] Arul Selvam Periyasamy, Max Schwarz, and Sven Behnke. "YOLOPose V2: Understanding and improving transformer-based 6D pose estimation". In: *Robotics and Autonomous Systems* 169 (2023), p. 104490. ISSN: 0921-8890. DOI: 10.1016/j.robot.2023.104490. URL: https://www.sciencedirect.com/science/article/abs/pii/S092188902300129X.

[36] J. Guan et al. "A Survey of 6DoF Object Pose Estimation Methods for Different Application Scenarios". In: *Sensors* 24.4 (Feb. 2024), p. 1076. DOI: 10.3390/s24041076.

[37] Yintao Ma et al. "Box6D: Zero-shot Category-level 6D Pose Estimation of Warehouse Boxes". In: *arXiv preprint arXiv:2511.15884* (2025). DOI: 10.48550/arXiv.2511.15884.

[38] Artur Cordeiro et al. "6D Pose Estimation for Objects based on Polygons in Cluttered and Densely Occluded Environments". In: *Proceedings of the 2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE. Paredes de Coura, Portugal, May 2024, pp. 15–21. ISBN: 979-8-3503-5234-4. DOI: 10.1109/ICARSC61747.2024.10535921.

[39] Karthik Karur et al. "A Survey of Path Planning Algorithms for Mobile Robots". In: *Vehicles* 3.3 (2021), pp. 448–468. DOI: 10.3390/vehicles3030027.

[40] Alka Choudhary. "Sampling-based Path Planning Algorithms: A Survey". In: *arXiv preprint* (2023). arXiv: 2304.14839 [cs.RO]. URL: https://arxiv.org/abs/2304.14839.

[41] Jiayuan Yu et al. "Path planning method of a humanoid robot based on a home network". In: 2019, pp. 875–880. DOI: 10.1109/ICARM.2019.8833853.

[42] Rui Jiang et al. "Depth Image–Based Deep Learning of Grasp Planning for Textureless Planar-Faced Objects in Vision-Guided Robotic Bin-Picking". In: *Sensors* 20.3 (2020), p. 706. ISSN: 1424-8220. DOI: 10.3390/s20030706. URL: https://www.mdpi.com/1424-8220/20/3/706.

[43] Hao Zhang et al. “DoraPicker: An Autonomous Picking System for General Objects”. In: *arXiv preprint arXiv:1603.06317* (2016). DOI: 10.48550/arXiv.1603.06317.

[44] Yuxuan Xie, Hao Wang, et al. “A Human-Robot Collision Avoidance Method Using a Single Camera”. In: *IEEE Transactions on Automation Science and Engineering* 19.4 (2022), pp. 3430–3442. DOI: 10.1109/TASE.2022.3145678.

[45] Florent P. Audonnet, Andrew Hamilton, and Gerardo Aragon-Camarasa. “A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2”. In: *The 22nd International Conference on Control, Automation and Systems (ICCAS 2022)*. IEEE. 2022, pp. 755–762. DOI: 10.23919/ICCAS55662.2022.10003832.

Chapter 7

Supplementary Material and Visual Documentation

To complement the textual description of the system, video material was produced documenting the robot's behavior at various stages of development. Each video corresponds to a specific functional area of the system and is publicly available on YouTube.

7.1 Simulation Validation (Isaac Sim)

Video showing 2 clips of the Booster T1 executing locomotion commands and path tracking in the Isaac Sim physics simulator, validating the control modules before deployment on the physical platform.
Watch at: YouTube - Isaac Sim Simulation

7.2 Localization and Navigation

Video showing the robot navigating through the laboratory environment while continuously localizing itself using the EKF with semantic landmarks. The video includes two distinct waypoint paths, demonstrating the system's ability to follow different routes while maintaining accurate pose estimation.
Watch at: YouTube - Localization and Navigation

7.3 Robotics National Festival 2026

Video of the robot navigating and localizing on the RoboCup soccer field during the Robotics National Festival 2026 in Barcelos. The video includes two clips of the robot following different paths across the field, validating the portability of the localization system to a completely different environment from the laboratory.
Watch at: YouTube - Robotics Festival 2026

7.4 Picking Trials

Video compilation of four representative picking trials recorded from the 20 experiments conducted, showing the visual lateral alignment, forward approach, dual-arm grasp sequence with custom L-shaped end-effectors, box lifting, and retreat. The selected clips were chosen to represent the range of outcomes observed during the experimental campaign.
Watch at: YouTube - Picking Trials

7.5 Complete End-to-End Mission

Video of a full mission execution showing the robot following a predefined path to the picking station, performing visual alignment and forward approach, executing the dual-arm grasp sequence to pick the box, and then navigating away along a different path. This video demonstrates the complete integrated cycle orchestrated by the finite-state machine, from initial navigation through manipulation to departure.
Watch at: YouTube - Complete End-to-End Mission