Universidade de Aveiro
2025

Bruno
Duarte

Manipulação Cooperativa de Objetos com Robôs Humanóides Guiados por Visão

Vision-Guided Cooperative Object Manipulation with Humanoid Robots

Universidade de Aveiro
2025

Bruno
Duarte

Manipulação Cooperativa de Objetos com Robôs Humanóides Guiados por Visão

Vision-Guided Cooperative Object Manipulation with Humanoid Robots

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Robótica e Sistemas Inteligentes, realizada sob a orientação científica do Doutor Filipe Silva, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, do Doutor Vítor Santos, Professor associado do Departamento de Mecânica da Universidade de Aveiro.

o júri / the jury

presidente / president

vogais / examiners committee

agradecimentos / acknowledgements

Agradeço ao meu orientador, Professor Filipe Silva, pela sua orientação, apoio, e paciência ao longo do desenvolvimento desta dissertação.
Agradeço ao meu co-orientador, Professor Vítor Santos, pelos conselhos construtivos e práticos, e pelo seu entusiasmo ao longo deste projeto.
Agradeço aos meus colegas de laboratório e ao seu responsável, pela ajuda prática dada e por todo o conhecimento passado sobre temas em que eu não era tão proeficiente.
Agradeço aos meus amigos e colegas de curso pelo apoio e motivação dados ao longo deste período.
Agradeço aos meus primos, Ivete e Daniel, por toda a ajuda e apoio dados no início da minha vida académica.
Agradeço ao meu pai, por todo o apoio dado, mesmo estando fora.
Agradeço à minha mãe e à minha irmã, por toda a ajuda, apoio e motivação dados.
Agradeço a todos os meus familiares e amigos que se preocuparam comigo durante todo este período.

palavras-chave

Robôs humanóides, Coordenação multi-robô, Controlo baseado em Visão, Deteção e Localização de Objetos, Manipulação Coperativa

resumo

Os robôs humanóides oferecem uma plataforma versátil para a exploração da manipulação colaborativa, mas a sua complexidade inerente, resultante do elevado número de graus de liberdade, das restrições de equilíbrio e das limitações dos sensores, torna a coordenação particularmente desafiante. Esta dissertação apresenta o design, a implementação e a avaliação de uma estrutura modular que integra visão e controlo, permitindo que pequenos robôs humanóides percebam e se aproximem de objetos em preparação para tarefas colaborativas.

O sistema proposto combina a perceção global, fornecida por uma câmara externa posicionada acima do ambiente, com um refinamento local através da câmara embutida em cada robô. A visão externa permite a consciência espacial multi-robô, a deteção de objetos e a prevenção de colisões, enquanto a visão a bordo do robô oferece precisão detalhada para o alinhamento a curta distância. Um conjunto de módulos baseados em ROS processa estas entradas visuais, gerando comandos de alto nível que são transmitidos aos robôs, onde módulos de controlo de baixo nível os executam através de controladores personalizados de marcha, movimento e articulações.

Para lidar com inconsistências na locomoção devido à tração limitada das solas metálicas dos robôs, foram realizadas experiências com materiais alternativos nos pés. Os resultados demonstraram que a adição de camadas de borracha de neoprene e de alcatifa melhorou significativamente a estabilidade e a repetibilidade da marcha e da rotação, reduzindo o deslizamento e a assimetria entre as curvas para a esquerda e para a direita. Além disso, as experiências de visão confirmaram que a utilização combinada de câmaras externas e embarcadas mitiga eficazmente as limitações de cada fonte isoladamente, conduzindo a um comportamento de aproximação mais fiável.

Os resultados deste trabalho estabelecem uma base sólida para futuras investigações na área do transporte cooperativo de objetos com robôs humanóides. Embora persistam desafios, particularmente no que diz respeito à dependência de filtros de cor, à sensibilidade à iluminação e à escalabilidade, a estrutura proposta demonstra que a combinação de visão em múltiplas camadas com controlo modular constitui um caminho eficaz para alcançar uma colaboração multi-robô fiável.

keywords

Humanoid robots, Multi-robot coordination, Vision-based control, Object detection and localization, Cooperative manipulation.

abstract

Humanoid robots offer a versatile platform for exploring collaborative manipulation, but their inherent complexity, stemming from high degrees of freedom, balance constraints, and sensor limitations, makes coordination particularly challenging. This thesis presents the design, implementation, and evaluation of a modular framework that integrates vision and control to enable small humanoid robots to perceive and approach objects in preparation for collaborative tasks.

The proposed system combines global perception, provided by an external overhead camera, with local refinement through each robot's onboard camera. External vision enables multi-robot spatial awareness, object detection, and collision avoidance, while onboard vision provides fine-grained precision for close-range alignment. A set of ROS-based modules processes these visual inputs, generating high-level commands that are transmitted to the robots, where low-level control modules execute them through customized walking, motion, and joint controllers.

To address inconsistencies in locomotion due to limited traction from the robots' metal soles, experiments were conducted with alternative foot materials. Results demonstrated that the addition of neoprene rubber and carpet layers significantly improved stability and repeatability of walking and rotation, reducing slippage and asymmetry between left and right turns. Furthermore, vision experiments confirmed that the combined use of external and onboard cameras effectively mitigates the limitations of either source alone, leading to more reliable approach behavior.

The outcomes of this work establish a robust foundation for future research in cooperative object transportation with humanoid robots. While challenges remain, particularly regarding reliance on color filters, sensitivity to lighting, and scalability, the framework demonstrates that combining multi-layered vision with modular control is an effective pathway toward reliable multi-robot collaboration.

acknowledgement of use of AI tools

Recognition of the use of generative Artificial Intelligence technologies and tools, software and other support tools.

I acknowledge the use of ChatGPT 4 and 5 (Open AI, https://chat.openai.com) to improve the initial structure of the dissertation and to proofread, sumarize and embellish the final draft.

Contents

List of Figures

List of Tables

Glossary

CHAPTER 1

Introduction

This introductory chapter will talk about the motivation behind this work and it's main objectives, as well as talk a bit about the structure of the work.

1.1 Motivation

As robotic systems become more prevalent in everyday environments, like elder care, industry or even education, Humanoid robots in particular have gained increased attention both in research and in pratical applications due to their potential to operate in environments designed for humans, as their anthropomorphic form allows them to interact with our tools, objects and spaces without the need to adapt said tools, objects or spaces to their use, in contrast to the use of other types of robots.

However, the complexity of such environments more easily exceeds the capabilities of a single robot acting alone. To overcome this limitations, multi-robot systems, that enable robots to collaborate to extend the functional range of each individual unit, have emerged. With them, it is possible to tackle tasks that would otherwise be infeasible. Coordination becomes particularly important in scenarios where tasks must be divided, synchronized, or shared—such as manipulating large objects, covering wider areas, or compensating for limited sensing. As such, the ability for multiple humanoid robots to work together in a coordinated and intelligent manner is more essential than ever.

1.2 Problem description and Objectives

One of the central challenges in multi-robot coordination is perception. Robots must be able to perceive not only the environment but also each other, with sufficient accuracy and timeliness to support real-time decision-making. Vision-based control provides a rich source of information for this purpose but comes with its own set of challenges, including occlusion, limited field of view, and the need for accurate calibration.

While many existing systems rely on either onboard cameras or external tracking systems, in this work we will try to leverage the combined strengths of both. Onboard vision offers robot-localized perception, while external vision can provide a broader, more stable view of the environment. Integrating both systems presents an opportunity to improve reliability, precision, and task coordination.

So, in this thesis we will develop a system that enables two humanoid robots to perform coordinated tasks using both internal (onboard) and external vision systems. The goal is to design a perception and control framework that uses both visual inputs to guide the actions of both robots in a shared environment.

This involves solving several subproblems:

- How to acquire and process visual information from both internal and external sources in real time;
- How to establish communication and coordination strategies between the robots and a central controller;
- How to plan and execute collaborative behaviors based on the combined visual input.

The primary cooperative task explored in this work involves two humanoid robots working together to transport an aluminum bar from one location to another within the shared workspace. This task was chosen as it embodies the challenges of physical coordination, mutual awareness, and synchronized motion, all of which are fundamental to effective multi-robot collaboration. Successful execution of this task demands that both robots maintain a shared understanding of the bar's position and orientation, adapt their movements in real time, and respond to subtle environmental changes, making it an ideal scenario to evaluate the effectiveness of the proposed vision-based coordination system.

This work targets small-scale humanoid robots, like the Darwin-OP platform, which are commonly used in academic research but face significant limitations in sensing and computation and the NAO platform, which is also very used in academic research and learning. The system must work within these constraints while demonstrating tangible improvements in task execution when using dual vision systems.

1.3 Structure

The remainder of this thesis is organized into several chapters, each addressing a specific aspect of the work and progressively building toward the development and validation of the proposed framework for collaborative humanoid robots.

- State of the Art: is chapter reviews the current literature on humanoid robot perception and control, with a particular focus on multi-agent coordination and object transportation. Both traditional computer vision approaches and recent deep learning methods are discussed, along with control strategies ranging from physics-based models to multi-agent reinforcement learning.
- Framework and Experimental Setup: This chapter introduces and talks about the process of understanding the robots available, the NAO and Darwin-OP. Their main characteristics, advantages, and limitations are described, as well as the software tools available and employed for simulation and development, such as Webots, ROS, and Choregraphe.

- Vision Framework: This chapter details the perception pipeline developed in the project. It describes the use of both external and onboard cameras, the image-processing strategies for object and robot detection, and the challenges of ensuring reliable localization under occlusion and lighting variability.
- Control Framework: This chapter presents the control architecture designed to govern robot behavior. Both the high-level commands managed by the central controller and the low-level modules implemented on the Darwin robots are discussed, along with the strategies for motion execution, balance preservation, and autonomous operation.
- Conclusions and Future Work: The thesis concludes with a summary of the main contributions, a discussion of the limitations of the proposed approach, and suggestions for directions in which the work can be extended, including cooperative transportation and improvements in communication and perception.

CHAPTER 2

State of the Art

In the field of humanoid robotics, collaborative object transportation poses a complex challenge that spans perception, control, and coordination. This chapter presents an overview of current approaches to visual perception and multi-robot coordination, particularly in the context of humanoid embodiments with many degrees of freedom. We examine both classical and modern techniques in object detection and robotic control, with a particular focus on machine learning-based coordination frameworks.

2.1 Vision in Humanoid Robots

2.1.1 Object Detection and Localization

A fundamental prerequisite for enabling robots to interact with their environment is the reliable detection and localization of objects. In humanoid robotics, this challenge is amplified by the complexity of unstructured environments, varying lighting conditions, and frequent occlusions. Traditional computer vision approaches remain highly relevant in this context due to their simplicity, low computational cost, and suitability for real-time applications on resource-constrained platforms such as small humanoid robots.

One of the most common strategies is color-based segmentation, where objects are identified based on predefined color ranges [1][2] [3] [4] [5] [6] [7]. By transforming the image from the RGB space to HSV, the hue, saturation, and brightness can be separately controlled, making it easier to account for lighting variations and shadows. Once the pixels within the target color range are extracted, morphological operations such as erosion and dilation are often applied to reduce noise and produce more coherent blobs, from which centroids can be computed to estimate object positions.

Shape-based methods provide an alternative or complementary approach. Techniques such as contour detection, Hough transforms, or bounding box fitting can be used to infer object orientation and approximate geometry. In scenarios where both the robots and the objects share similar visual properties (e.g., metallic reflectivity), these classical techniques can struggle due to ambiguous segmentation and poor contrast with the background.

To overcome such limitations, many works introduce fiducial markers (e. g. , AprilTags, ARToolkit markers, or even simple colored strips), which provide high contrast, geometric cues, and sometimes encoded IDs for object or robot identification [8] [9]. These solutions are particularly attractive in laboratory settings, where repeatability and robustness are prioritized over full generalization.

While traditional techniques do not match the flexibility and adaptability of modern learning-based methods, they can still be improved upon and remain a pragmatic choice for lightweight platforms and controlled environments, where their robustness and low latency are advantageous [10].

2.1.2 Learning-Based Approaches

When exploring deep learning–based approaches for vision, one of the most widely adopted methods is YOLO (You Only Look Once) [11] , a real-time object detection and image segmentation framework. YOLO has become a benchmark in robotic perception due to its balance of speed, accuracy, and ease of deployment. Unlike traditional region-proposal methods, YOLO treats detection as a single regression problem, directly predicting bounding boxes and class probabilities from full images in one evaluation. This architecture makes it particularly well-suited for real-time applications in robotics, where decisions must be taken within milliseconds.

YOLO has already demonstrated effectiveness in industrial robotic systems, especially from version YOLOv5 onward, where its performance in terms of accuracy and usability has made it attractive for integration in perception pipelines [12]. Moreover, its adaptability has been showcased in specialized domains, such as text-guided object detection [13], and, importantly, in applications on humanoid robots themselves. For instance, Biswas et al. [14] and Sun et al. successfully deployed YOLO on a NAO platform for reliable real-time detection of objects, highlighting its relevance to scenarios similar to ours, and there are also other similar works [15].

Architecturally, YOLO follows a deep convolutional design reminiscent of GoogLeNet, comprising 24 convolutional layers, four max-pooling layers, and two fully connected layers. Later versions, such as YOLOv4, YOLOv5, and more recently YOLOv7/YOLOv8, incorporate advanced elements such as residual connections, cross-stage partial networks (CSPNet), and anchor-free detection heads, improving both efficiency and robustness in cluttered or dynamic environments.

While other state-of-the-art detectors such as Faster R-CNN (Region-based Convolutional Neural Network) [16] [17] [18] [19] or SSD(Single Shot MultiBox Detector) [20] [21] [22] [23] [24] [25] have been extensively used in computer vision and robotics , they often involve trade-offs: Faster R-CNN provides higher accuracy but at the cost of computational speed, whereas SSD achieves faster inference but can struggle with small object detection. In contrast, YOLO offers a strong balance between accuracy, speed, and implementation simplicity, making it particularly suitable for real-time multi-robot coordination tasks.

These qualities make YOLO a strong candidate for integration in humanoid robot vision pipelines, particularly when seeking to move beyond handcrafted features such as color filtering, and toward more robust, scalable solutions.

2.1.3 Multi-Camera and Sensor Fusion Approaches

A recurring challenge in robotic vision is balancing local precision with global awareness. While onboard cameras provide a robot with a direct perspective for fine-grained tasks, their field of view is inherently limited, often subject to occlusions from the robot's own body or the manipulated object, and can also suffer from reduced stability and precision when the robot is in motion. Conversely, external cameras, whether mounted overhead or strategically placed in the environment, can capture the global scene but often lack the close-range accuracy needed for manipulation, and can similarly suffer from occlusions.

For this reason, many recent works in humanoid and multi-robot coordination have explored sensor fusion approaches, where local and global information are combined to obtain a more reliable and adaptable perception system [26]. Overhead cameras have been successfully integrated into soccer robotics platforms (e.g., RoboCup) to provide a better global positioning and team coordination [27], while onboard vision handles more precisely immediate interactions with the ball or obstacles [28]. However, in most cases they appear combined with other types of sensors. Similarly, in cooperative manipulation, fusing external cameras with robot-mounted sensors improves robustness to occlusion and lighting variability, and reduces the reliance on a single point of failure.

Typical strategies for fusion include:

- Hierarchical control, where global vision provides long-range navigation goals, and local vision ensures precise alignment during execution.
- Probabilistic fusion techniques, such as particle filters or Kalman filters, to merge uncertain detections from different perspectives.
- Learning-based fusion, where neural networks are trained to weight and combine different streams of visual input depending on the context.

In the context of this thesis, the combined use of the external overhead camera and the Darwin-OP's onboard camera follows a similar principle. The external camera ensures global coordination between multiple robots and avoids inter-robot collisions, while the onboard camera refines positioning and alignment when approaching the object for manipulation. This dual-layer vision approach leverages the strengths of each modality and mitigates their weaknesses, offering a practical solution for robust perception in cooperative humanoid robotics.

2.2 Control in Humanoid Robots

2.2.1 Control, Locomotion and Balance

Modern humanoid robot control frameworks are designed with modular architectures that separate locomotion, balance, and perception into interconnected yet independently manageable subsystems [29]. This modularity allows for adaptable gait generation and stability control, enabling researchers to fine-tune walking behaviors without reconfiguring the entire control stack. However, the inherent uncertainty introduced by multi-directional forces acting on the robot's body, such as ground reaction forces, inertia during motions, or other external disturbances, make it a challenge to create frameworks to sucessfully control the robots during tasks execution or locomotion, while still mantaining coordination and balance [30] [31] [32] [33] [34] [35].

Locomotion is one of the defining challenges in humanoid robotics, as these systems must continuously maintain balance while walking, turning, or manipulating objects. Unlike wheeled robots, humanoids rely on a dynamically stable gait to remain upright, requiring the coordination of multiple joints and the constant adjustment of their center of mass (CoM). Even small perturbations, such as uneven flooring, external pushes, or the shifting of weight while carrying an object, can destabilize the robot if not properly accounted for.

A common foundation for humanoid balance is the Zero Moment Point (ZMP) criterion [36] [37], first introduced by Vukobratović [38] [39] and later refined into practical control methods. ZMP-based walking ensures that the projection of the robot's CoM remains within the support polygon formed by its feet, thereby preventing tipping [40]. Numerous gait generation frameworks for humanoid robots, including NAO and Darwin-OP [41] [42], employ ZMP controllers or derivatives such as the Linear Inverted Pendulum Model (LIPM) [43] [44] to generate stable trajectories.

In addition to ZMP control, capture point methods and model predictive control (MPC) have been proposed for improved robustness [45]. Capture point control enables the robot to "catch" its balance after disturbances by modifying step placement, while MPC predicts the future state of the robot over a time horizon, adjusting footstep locations and CoM trajectories accordingly. These approaches have been particularly effective in dealing with dynamic perturbations and have been applied in humanoids ranging from small-scale robots to human sized robots.

When locomotion is coupled with manipulation, the challenge increases. Adjustments of the arms shift the center of mass and introduce torques that affect balance. For this reason, some frameworks integrate whole-body control (WBC) approaches [46] [47], where locomotion and arm motions are solved in a unified optimization problem. This is particularly relevant in collaborative tasks, where robots must move while carrying or stabilizing objects together.

For small humanoids like Darwin-OP, locomotion modules are often simplified due to hardware constraints, relying on pattern generators or precomputed walking motions. However, modifications to allow independent arm movements, as required for carrying an object, highlight the importance of balancing upper- and lower-body coordination even in constrained platforms.

Overall, locomotion and balance research continues to evolve from rule-based approaches (like ZMP) toward predictive, optimization-based methods, many of which are now being enhanced through reinforcement learning [48]. In the context of this thesis, the emphasis lies on ensuring that locomotion remains consistent and stable enough to allow cooperative manipulation, despite hardware limitations and surface irregularities.

2.2.2 Trajectory Planning and Collision Avoidance

Beyond maintaining balance, humanoid robots must also plan and execute trajectories that guide them toward a goal while avoiding collisions with obstacles and, in multi-robot scenarios, with each other. Unlike wheeled robots, trajectory planning for humanoids is complicated by the need to consider step placement, body posture, and dynamic stability at every moment of motion.

Classical approaches to trajectory generation often rely on predefined walking patterns combined with simple obstacle avoidance strategies, such as potential fields or vector field histograms [49] [50] [51] or fuzzy control [52]. While computationally efficient, these methods may struggle with complex environments or when precise coordination between multiple agents is required.

More advanced approaches incorporate kinematic and dynamic models into the trajectory planning process. By predicting feasible step locations and the evolution of the robot's center of mass, planners can generate dynamically consistent walking paths. In particular, methods based on the Linear Inverted Pendulum Model (LIPM) have been extended to handle step timing and obstacle avoidance simultaneously.

In multi-robot contexts, coordination introduces further challenges. Robots must share workspace information and adjust trajectories in real time to prevent interference. Approaches such as reciprocal velocity obstacles (RVO) [53] [54] [55] [56] and optimal reciprocal collision avoidance (ORCA) [57] [58] have been successfully applied in multi-agent navigation, allowing each robot to locally adapt its velocity to avoid collisions while maintaining global coordination. In humanoid robotics, these concepts are often integrated with balance constraints to ensure collision-free yet stable motions.

More recent works explore learning-based methods, where policies for navigation and obstacle avoidance are learned through reinforcement learning or imitation learning. While promising in simulation, these approaches face challenges when transferring to physical humanoids due to noise, model inaccuracies, and limited onboard computation.

In this thesis, trajectory planning is addressed from a practical standpoint: the external camera provides global position estimates of both robots and the object, while high-level commands (e.g., "turn left," "walk forward") guide the robots along safe paths. Collision avoidance is achieved primarily through global awareness, with the external vision ensuring that robots do not interfere with each other, while fine adjustments near the object are handled locally by onboard vision.

2.2.3 Multi-Robot Coordination and Communication

When multiple humanoid robots operate in a shared environment, the challenge extends beyond individual locomotion and trajectory planning [59] [60]. Robots must also coordinate their actions to achieve shared objectives, avoid interfering with one another, and adapt to dynamic changes in their surroundings [61]. Effective coordination therefore requires both communication mechanisms and decision-making frameworks that balance global goals with local autonomy, allowing the robots to reach new heights of tasks available [62].

Early approaches to multi-robot coordination often relied on centralized architectures, where a single controller computed trajectories for all robots and distributed commands accordingly. While this ensures globally consistent decisions, centralized schemes suffer from scalability issues, single points of failure, and high communication demands.

By contrast, decentralized methods allow each robot to make decisions based on its own perception while exchanging information with its peers. This can be achieve by a Symmetry or Leader-Follower type of cooperation [63] [64] [65] Techniques such as behavior-based coordination and market-based task allocation have demonstrated flexibility in distributing roles and responsibilities among agents. In collaborative manipulation, however, decentralized approaches must ensure that local actions remain dynamically compatible, particularly when robots share physical contact with the same object.

Communication plays a crucial role in enabling these strategies. Standard protocols include TCP/IP and UDP, which trade off reliability and latency, as well as higher-level middleware such as ROS topics and services, which provide structured message passing between processes. More recent research has explored ad hoc wireless networks and low-latency communication frameworks for robot teams operating in dynamic environments.

In the context of humanoid robots, coordination is further constrained by balance and motion stability. For example, two robots approaching an object must not only avoid collisions but also synchronize their arrival times and postures to enable cooperative actions such as lifting or pushing [66]. Reinforcement learning has been applied to such problems, enabling robots to jointly optimize movement strategies through trial and error. However, physical humanoid platforms often require hybrid solutions that combine robust communication with simplified, rule-based coordination to guarantee safety.

In this thesis, a hybrid centralized approach is adopted. A central controller ensures global consistency by monitoring positions through an external camera and issuing commands to each robot. At the same time, local autonomy, activated when robots are close to the object, allows for fine-tuned adjustments without overloading the communication channel. This balance provides both robustness and flexibility, ensuring that coordination remains stable even in the presence of sensor noise or communication delays.

2.3 Multi-Agent Reinforcement Learning in Robotics

Reinforcement Learning (RL) has been increasingly applied to robotics due to its capacity to learn adaptive behaviors directly from interaction with the environment [67] [68] [69] [70] [71] [72].

When extended to multi-robot systems, it gives rise to Multi-Agent Reinforcement Learning (MARL), where each robot (agent) must not only optimize its own policy but also account for the actions of others. Compared to single-agent RL, MARL introduces additional complexity due to issues such as non-stationarity (as agents learn simultaneously, the environment keeps changing), credit assignment (determining which agent's action contributed to a reward), and scalability with larger teams [73].

Several paradigms have been proposed. A straightforward strategy is Centralized Training and Execution, in which a single controller learns and issues commands for all agents using full state information [74]. While this ensures tight coordination, it scales poorly as the number of agents grows and creates a single point of failure. To overcome these limitations, the widely adopted Centralized Training with Decentralized Execution (CTDE) paradigm was introduced, where agents are trained with access to global state information but operate independently at runtime [75] [76]. This approach underpins methods like MADDPG and QMIX [77] [78], which have been shown effective in tasks requiring high coordination. In contrast, fully decentralized learning relies only on local observations, enabling better scalability but often at the cost of reduced coordination [79]. A third trend, communication-aware MARL, allows agents to learn what and when to share information, which is particularly relevant in multi-robot systems with bandwidth or latency constraints.

Applications of MARL in robotics include multi-robot navigation and collision avoidance in crowded environments, cooperative manipulation where multiple agents jointly transport or manipulate objects, and task allocation where roles emerge dynamically from learned policies . These advances highlight MARL's potential for complex, dynamic, and distributed robotic systems.

Nevertheless, challenges remain for its deployment in humanoid robotics. Training requires large amounts of data, making real-world exploration infeasible without simulation, while sim-to-real transfer introduces robustness issues. Furthermore, the unstable dynamics and balance requirements of humanoids pose additional difficulties compared to wheeled or fixed-base robots. Despite these limitations, MARL presents a promising pathway for enabling autonomous, scalable cooperation in humanoid teams, complementing more traditional centralized control strategies.

2.4 Summary and Research Gap

The state of the art in vision-based control and multi-robot coordination highlights both the rapid progress and the remaining challenges in deploying humanoid robots for collaborative tasks. On the perception side, traditional computer vision techniques remain valuable for simple environments with well-defined color or shape features, while deep learning approaches, such as YOLO, have greatly expanded robustness and adaptability under complex and dynamic conditions. However, these methods often come at the cost of computational demand, dataset requirements, and reduced interpretability.

In locomotion and balance, extensive research has demonstrated that humanoid robots can achieve stable gaits and object manipulation even under uncertainty, but issues such as slippage, ground unevenness, and body coordination remain critical, particularly in small humanoid robots where hardware constraints are more severe. These challenges underscore the need for lightweight solutions that carefully integrate perception with control.

Finally, multi-agent reinforcement learning provides powerful frameworks for enabling robots to cooperate in transporting and manipulating objects. Centralized approaches deliver strong coordination but scale poorly; decentralized strategies scale better but often lack tight synchrony; and CTDE offers a promising compromise by combining centralized learning with decentralized autonomy. The choice of paradigm ultimately depends on the trade-off between coordination quality, scalability, and real-time feasibility.

In summary, while the field has produced numerous robust techniques for perception, control, and cooperation, gaps remain in integrating these components into cohesive systems capable of operating reliably in real-world scenarios. These insights directly motivate the approach of this thesis, which seeks to combine global and local vision with modular control strategies in order to achieve reliable collaborative object transportation using humanoid robots.

CHAPTER 3

Framework and Experimental Setup

This chapter will serve to present the conditions in which the work was done, more specifically the environment, the robots hardware and software available, and simulation. The importance of the work mentioned in this chapter cannont be overstated as it is where the conditions for all the posterior work are done.

3.1 Task Definition

The overall goal of this work is the design and validation of a modular framework that enables small humanoid robots to prepare for and execute collaborative object manipulation tasks. This complex process is broken down into a structured four-phase pipeline, where the responsibilities of global perception, local control, and coordinated motion are clearly separated. The objective is for two humanoid robots to cooperatively transport an aluminum bar (placed on a support) to a new designated support location. The general approach is structured as follows:

- Element Detection and Global Awareness: The system first detects all key elements: the two robots, the aluminum bar, and the two support locations, within the environment. This is achieved robustly using an external overhead camera and color markers combined with standard computer vision techniques. By leveraging a fixed, top-down view and geometric markers, the system simplifies the perceptual challenge, eliminating the need for complex, real- time extrinsic calibration between the camera and the robots.
- Coarse Approach and Assignment: Using the relative 2D information derived from the markers, the robots are guided to approach the target object. A simple proximity rule (based on distance in pixels) is used to assign each robot to a specific end of the object. This phase ensures the robots achieve a coarse alignment, preparing them for the object elevation.
- Coordinated Object Elevation: Once the robots are within close proximity and roughly aligned, they transition to a local control mode. Here, the robots use their onboard

cameras to refine their arm movements and achieve the precision needed to coordinately lift the object from its support, as illustrated in Figure 3.1.

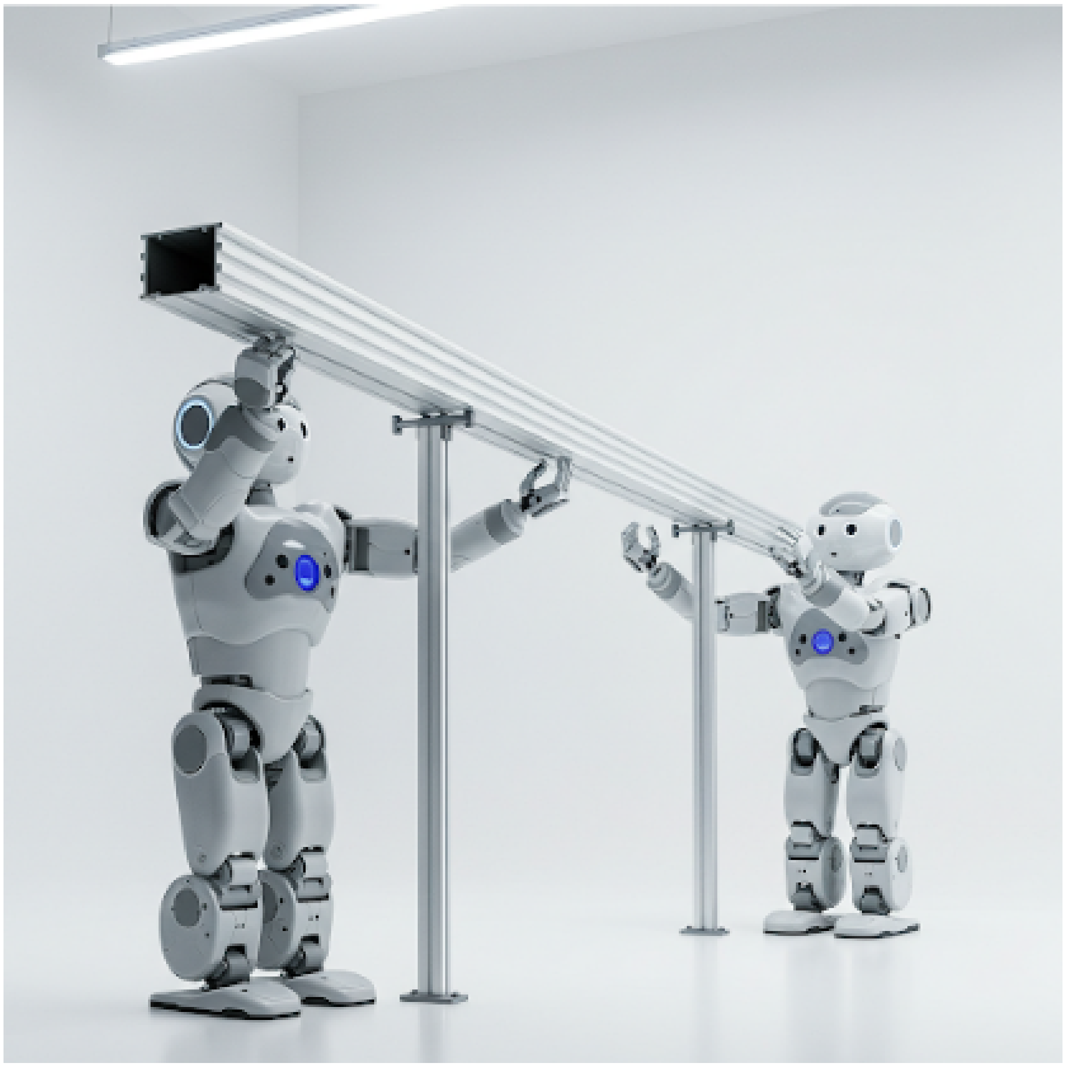


Figure 3.1: Example of 2 humanoid robots picking up an aluminum bar.

- Coordinated Transport and Releasing: The final phase involves planning a smooth trajectory for the cooperative transport of the object to the new support location. During this movement, the system must ensure stability and may integrate sensor data (such as onboard cameras or force sensors, if available) to monitor the object's orientation and correct for any trajectory deviations before precisely placing the bar onto the target support.

3.2 Familiarization and Setup

Before implementing the coordinated multi-robot system, a significant portion of the initial work was dedicated to understanding and configuring the tools and hardware required for the project. One of the things to evaluate were the conditions of the physical encountered in which the robots would operate. All of the work for this dissertation was developed in the

LAR of University of Aveiro, and that is where the robot's workstation is located. Factors such as floor texture, lighting conditions, and available space were semi-constant during the project and were analysed to identify any constraints that could affect the tasks of the robots via affecting locomotion or vision processing.

In parallel, we also assessed the robot's conditions. Initially, we had available a NAO v5 robot and a Darwin-OP, but with the possibility to use 2 Darwin-OPs instead. Establishing a reliable communication pipeline was also an important part of this stage. This included establish a reliable communication pipeline to a central controller, and becoming familiar with the robot's API and the available motion and vision frameworks. With them, basic tests were conducted to validate robot connectivity and hardware conditions, such as the state of the battery or the motors, using motion commands.

The system's modular architecture is designed to manage the transition between global (external camera) and local (onboard camera) perception, ensuring reliable performance across all four phases. This four-phase approach forms the structural backbone of the remaining dissertation. The core perceptual components, encompassing Phase 1 (Element Detection) and the initial guidance for Phase 2 (Coarse Approach), are detailed and evaluated within Chapter 4: Dual-Vision Approach. This includes the development and validation of the external camera system and the integration with the onboard vision modules. Conversely, the implementation of the physical actions, namely the final approach refinement, Phase 3 (Coordinated Object Elevation), and the execution of Phase 4 (Coordinated Transport), is the focus of Chapter 5: Motion Control. This chapter addresses the locomotion algorithms, the stability improvements made to the robot platforms, and the low-level control necessary for stable, reliable cooperative manipulation.

This setup phase, although not directly part of the system's final functionality, was crucial in ensuring that the robots could be safely and reliably used for the experiments described in later chapters. It also provided valuable insights into the limitations and practical considerations of working with humanoid robots in a real-world setting.

3.3 Environment

The laboratory environment in which the experiments were conducted provided relatively stable conditions overall, but it was not without certain constraints that influenced robot performance. The illumination in the lab was generally consistent, with a mix of artificial lighting and natural light from nearby windows. However, during particularly sunny days, increased ambient light caused the floor to reflect more light than usual, which occasionally interfered with the vision system's performance, as both the robots and the obeject to transport were made of metal.

The floor surface, while overall flat and level, had a mineral-like texture that contributed to these light reflections. This subtle sparkling effect, caused by fine reflective particles embedded in the surface, introduced small fluctuations in brightness and contrast across captured images. While these effects were not severe, they required some tuning of the vision algorithms, especially in tasks involving thresholding or color segmentation.

In addition to its reflective properties, the grey-colored floor included minor surface fissures and cracks distributed unevenly across the workspace. Although not large enough to obstruct movement, these irregularities occasionally affected the robots' foot placement during walking or turning maneuvers. To minimize disruptions, key task areas were selected to avoid the most uneven sections.

Overall, the environment presented a mostly controlled but realistically imperfect setting, providing useful insights into how vision-based robotic systems cope with subtle environmental variations. These observations informed several of the design and implementation decisions detailed in subsequent chapters.

The aluminum bar, due to its rigid structure, moderate weight, and reflective surface, presents both mechanical and perceptual challenges: it requires precise positioning and force sharing between the robots during manipulation, while also potentially introducing visual noise due to light reflections.

3.4 Framework

3.4.1 External Camera

The external camera was rigidly mounted at a height of 180 cm,. This setup allows for the assumption of an approximated orthographic projection of the floor, which simplifies the overall complexity, avoiding the need for a full 3D extrinsic calibration.

To optimize computational performance and minimize latency, the global perception system primarily operates in the pixel domain, intentionally avoiding real-time conversion to real-world units (mm/cm). To facilitate external visual tracking and monitoring of the robot workspace, a static overhead camera was mounted at an approximate height of 180cm, positioned orthogonally to the robots' plane of movement, providing a complete top-down view of the area in which the robots would operate. To ensure measurement consistency within the pixel domain, the workspace was limited to the central area of the camera's field of view, where lens distortion is minimal, but the total area corresponded to a rectangle of 140cm * 105cm. Consequently, critical metrics such as proximity and alignment thresholds are defined directly in image coordinates. Although this approach streamlines the framework's architecture, it necessitates empirical validation to ensure that the defined pixel thresholds correspond reliably to the physical distances required for successful robot manipulation. This Logitech camera used was capable of capturing video at a frame rate of 30 frames per second, with an max image resolution of 720x480m. The structure on which the camera was fixed placed the camera at the center of the rectangle lenght-wise and 17cm width-wise, and directly facing downwards. The device was connected via USB-A to the central control unit, which in this case was a personal laptop computer serving as the main processing and coordination node of the system.

3.4.2 Communication

Information/Files Transfer

Communication with the robots involved two main aspects: first, the transfer of program files to the robots, including the executables that would run onboard; and second, the exchange of messages during program execution.

For file transfer and remote access, several options exist—such as Telnet, rsh, or more modern alternatives like Mosh and VPN-based access—but SSH quickly stood out due to its simplicity and robustness. Connecting only requires the robot's IP address and user account, using a command like: ssh user@ip

For added convenience, SSH keys can be configured, allowing secure, passwordless access. This ease of use, combined with strong encryption, made SSH the natural choice for program deployment.

For message exchange between robots and the central controller, the choice was between TCP/IP and UDP, the two most common transport-layer protocols. TCP was selected as the default because of its reliability: it guarantees ordered delivery and prevents packet loss, both critical to avoid desynchronization or collisions between robots. UDP, while faster and lower-latency, was less suitable since occasional lost packets could severely compromise coordinated behavior.

Network Configuration

Initially, communication was carried out over a wired LAN/Ethernet connection, which ensured low latency and reliable data transfer. However, this tethered setup restricted robot mobility and was impractical for natural collaborative movement. To overcome this limitation, the system was later adapted to a wireless configuration, using a smartphone hotspot to create a local Wi-Fi network for both the laptop and the robots.

This shift to wireless networking provided significant benefits: it freed the robots from physical constraints, simplified multi-robot communication (given the central controller's single Ethernet port), and offered straightforward setup and reliable performance in local conditions. The use of a personal smartphone hotspot was particularly advantageous compared to institutional networks such as Eduroam, which often involve complex authentication procedures and restricted access policies.

To connect to the robot via Wireless Internet, one may use the command line just like if one was connecting any computer with Ubuntu. However, simply connecting a monitor and a keyboard to the robot and using the desktop interface is also available, and a lot easier, so this was the chosen approach.

3.5 Humanoid Robots

When studying collaboration in robotics, small humanoid robots present both advantages and limitations. On the positive side, their compact size, relatively low cost, and built-in sensing and actuation capabilities make them accessible research platforms that allow for rapid

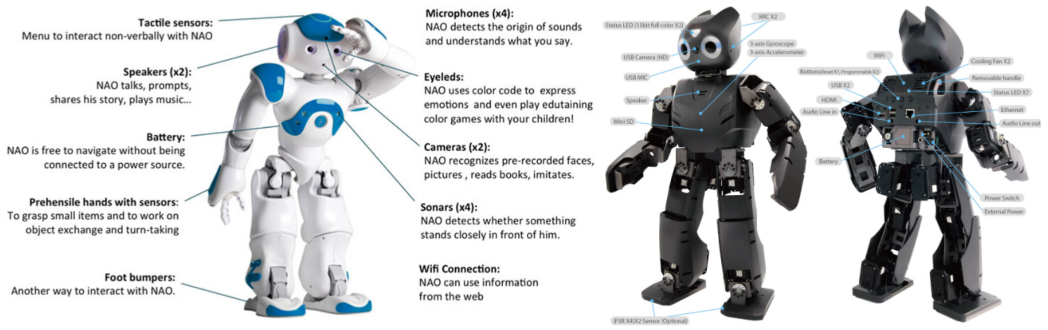

Tactile sensors:
Menu to interact non-verbally with NAO
Speakers (x2):
NAO talks, prompts, shares his story, plays music...
Battery:
NAO is free to navigate without being connected to a power source.
Prehensile hands with sensors:
To grasp small items and to work on object exchange and turn-taking
Foot bumpers:
Another way to interact with NAO.
Microphones (x4):
NAO detects the origin of sounds and understands what you say.
Eyeleds:
NAO uses color code to express emotions and even play edutaining color games with your children!
Cameras (x2):
NAO recognizes pre-recorded faces, pictures , reads books, imitates.
Sonars (x4):
NAO detects whether something stands closely in front of him.
Wifi Connection:
NAO can use information from the web
Status LED (15bit full color X2)
MIC X2
3-axis Gyroscope
3-axis Accelerometer
USB Camera (HD)
USB MIC
Speaker
Mini SD
(FSR X4)X2 Sensor (Optional)
WiFi
Buttons(Reset X1, Programmable X2)
USB X2
HDMI
Audio Line in
Battery
Cooling Fan X2
Removable handle
Status LED X7
Ethernet
Audio Line out
Power Switch
External Power

Figure 3.2: Image depicting the sensors available on both the NAO and Darwin-OP

prototyping and experimentation in safe environments. Their anthropomorphic design also enables realistic studies of human-like motion, perception, and interaction within multi-robot teams. However, their reduced power, payload capacity, and sensor resolution compared to larger platforms impose constraints on task complexity and scalability. Additionally, their limited stability and simplified mechanics can make precise coordination or manipulation more challenging, meaning results must often be extrapolated with care when transferring to full-scale systems.

Initially, the robotic platforms employed in this thesis were a Darwin-OP humanoid robot, selected for its compact form factor, onboard processing capabilities, and open-source software support; and a NAO, chosen for its wide adoption in research, robust motion libraries, integrated sensors, and user-friendly programming environment, which made it another attractive platform for initial development and testing. 3.2

3.5.1 NAO Robot

NAO is a widely used humanoid robot developed by SoftBank Robotics (formerly Aldebaran Robotics), designed for education, research, and human-robot interaction. Standing approximately 58 cm tall and weighing around 5.4 kg, NAO is equipped with 25 degrees of freedom, enabling a wide range of human-like motions including walking, gesturing, and sitting.

NAO features a rich suite of sensors, including stereo cameras, sonar and infrared sensors, inertial measurement units, and force-sensitive resistors in its feet. Its onboard processing is handled by an embedded Intel Atom processor, and it runs NAOqi, a proprietary operating system that supports Python, C++, and Choregraphe (a graphical programming interface). These features make NAO especially well-suited for tasks involving speech recognition, computer vision, object tracking, and coordinated motion. In addition, NAO integrates microphones, speakers, and expressive LEDs, which expand its capabilities for multimodal interaction and communication. Networking options such as Wi-Fi and Ethernet provide reliable connectivity, making it practical for multi-robot experiments and interaction with external controllers.

Thanks to its modular software architecture, wide adoption in the research community, and robust hardware platform, NAO has become a standard in academic and research environments for studies in robotics, artificial intelligence, and human-robot interaction.

Despite its versatility, certain limitations—such as limited payload capacity, relatively slow locomotion, and a more closed hardware design compared to open-source platforms—must be considered when selecting it for physically demanding or precision-critical tasks.

In the initial stages of this thesis, NAO was explored as a candidate platform for implementing cooperative multi-robot tasks. Its user-friendly programming interfaces (including support for Python and the Choregraphe visual environment), together with its integrated vision and communication capabilities, made it a convenient choice for early development and testing.

3.5.2 Darwin-OP

The Darwin-OP (Dynamic Anthropomorphic Robot with Intelligence – Open Platform) is a compact, fully programmable humanoid robot developed by ROBOTIS in collaboration with Virginia Tech and Purdue University, with support from the National Science Foundation (NSF). Designed to serve as an open research platform in the fields of humanoid locomotion, computer vision, and multi-agent robotics, Darwin-OP has been widely adopted in academia due to its accessibility, modifiability, and cost-effectiveness. And in this project, we used the first version of it.

This robot stands approximately 45 cm tall, weighs around 2.9 kg, and is equipped with 20 degrees of freedom actuated by Dynamixel MX-28 servos, offering precise position control and torque feedback. Internally, the first-generation Darwin-OP features an Intel Atom Z530 processor, 1 GB of RAM, and a modest onboard SSD, running a Linux-based operating system (commonly Ubuntu 9.10 or later).

One of the key advantages of Darwin-OP lies in its open-source hardware , in the form of circuit diagrams and CAD files; and software architecture, which provides researchers with low-level access to motor control, sensor data, and vision processing. Development is typically done in C++, with Python support available, and the system is highly customizable through its open SDK.

However, the hardware limitations of the original Darwin-OP model,particularly the constrained processor and memory,make it unsuitable for running the full Robot Operating System (ROS) in most use cases. While minimal ROS configurations have been attempted in research contexts, performance issues such as latency, dropped frames, and instability during locomotion often arise. As such, all the ROS work was done in the central controller, which in turn used communication protocols and Darwin-OP SDK to control the robot.

Throughout this project, Darwin-OP proved to be a reliable and responsive platform, especially for tasks requiring direct control and vision-based coordination. Its mechanical robustness, lightweight design, and ease of hardware access made it well-suited for the cooperative robotic tasks explored in this research.

The robot’s center of mass is strategically located near the pelvic region, which is considered optimal for achieving stable bipedal locomotion. This central positioning enhances the robot’s ability to maintain balance and supports a more efficient distribution of inertia during dynamic gait cycles, contributing to smoother and more human-like walking behavior.

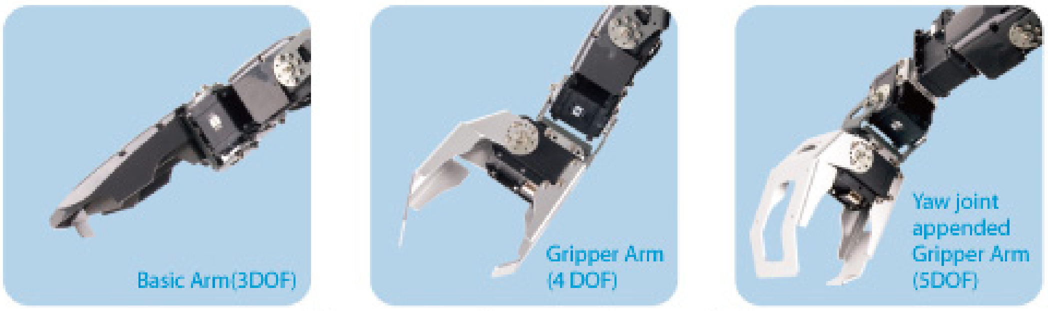


Figure 3.3: Possible grippers for the Darwin-OP hand

Furthermore, Darwin-OP features a modular, network-based architecture, which allows for flexible hardware customization. Each actuator operates as an independent module on a shared communication bus, making it straightforward to modify or replace specific components, such as the end-effectors or hand grips 3.3. This modularity is especially beneficial in experimental settings, where different tasks may require tailored mechanical configurations or sensor attachments.

In addition, the robot's hollow structural frames contribute to both reduced overall weight and increased internal accessibility. This design choice not only makes the robot lighter and more agile, but also provides room for the integration of additional sensors or wiring, should the user require extended functionality. Moreover, the open internal design significantly facilitates routine maintenance and hardware inspection, which is particularly valuable in long-term experimental or developmental environments.

3.6 Simulation

To minimize mechanical wear on the physical robots and to ensure continuous development even outside the LAR (Laboratório de Automação e Robótica), simulation played a crucial role throughout this project.

In the initial stages, development began with the NAO robot, for which Aldebaran provides a dedicated simulation and programming tool called Choregraphe. This application was specifically designed for NAO and Pepper robots, offering an intuitive visual interface that includes a wide array of built-in action blocks known as "macros". These macros range from basic tasks—such as speech synthesis, listening, sensing movement, and manipulating joints—to more complex, compound behaviors like standing up, walking, or reacting to visual cues.

One of Choregraphe's key advantages lies in its seamless integration with the physical NAO robot. By simply connecting the robot to a computer using an Ethernet cable, developers can transfer and execute behavior sequences in real-time on the actual robot. This feature made it easy to evaluate NAO's functionality and identify hardware or software limitations early in the project.

However, Choregraphe is exclusively tailored for Aldebaran's platforms and does not offer direct support for the Darwin-OP. As the focus of the project shifted toward the Darwin-OP robot, we began evaluating simulation tools that could support this hardware platform more effectively.

One such candidate was Gazebo, a widely used physics-based simulator known for its high-fidelity environments and tight integration with the Robot Operating System (ROS). Gazebo initially seemed promising due to prior experience and its extensive feature set. However, it proved to be highly resource-intensive and difficult to configure, especially on machines with limited processing power. Additionally, while Gazebo supports the NAO robot, accessing the necessary simulation libraries was not straightforward. Another major drawback was that ROS-based control code used within Gazebo could not be directly deployed to the Darwin-OP. This is because Darwin-OP's onboard memory is insufficient to run a full ROS environment, limiting the portability of code between simulation and reality.

Ultimately, the simulator selected for the majority of the project was Webots, an open-source, cross-platform robotics simulator developed by Cyberbotics. Webots offers native support for the Darwin-OP, complete with a ready-to-use 3D model and example controllers. Aldebaran has also provided an official NAO model for Webots, making it a versatile tool for multi-platform projects.

Webots was significantly lighter than Gazebo, allowing for smoother performance and easier iteration. Most importantly, code written and tested in Webots for the Darwin-OP could be cross-compiled and deployed directly to the robot with minimal adjustments. This feature offered a much faster development cycle and reduced discrepancies between simulated and real-world behavior.

3.6.1 Simulators Use - Choregraphe

Using Choregraphe to Interact with NAO

Once the NAO robot is physically connected to a computer via an Ethernet cable, launching Choregraphe provides an intuitive and interactive environment for robot control and development.To initiate the connection, the user simply clicks the "Connect" icon in the toolbar and selects the desired robot from the list of available devices. From there, the connection can be seamlessly transitioned to a wireless network, enabling untethered mobility for the robot during operation.

Alternatively, developers may choose to connect to a virtual robot,ideal for early,stage development and testing without physical hardware.

Upon successful connection, Choregraphe provides a comprehensive overview of the robot's current state. A 3D representation of the robot mirrors its posture in real-time, and the camera feed from the robot's onboard sensors is displayed within the interface. Individual joints can be manipulated directly through the GUI, allowing users to explore the robot's range of motion and physical condition,all without writing a single line of code. These features are accessible through intuitive control panels on the right-hand side of the application (insert image).

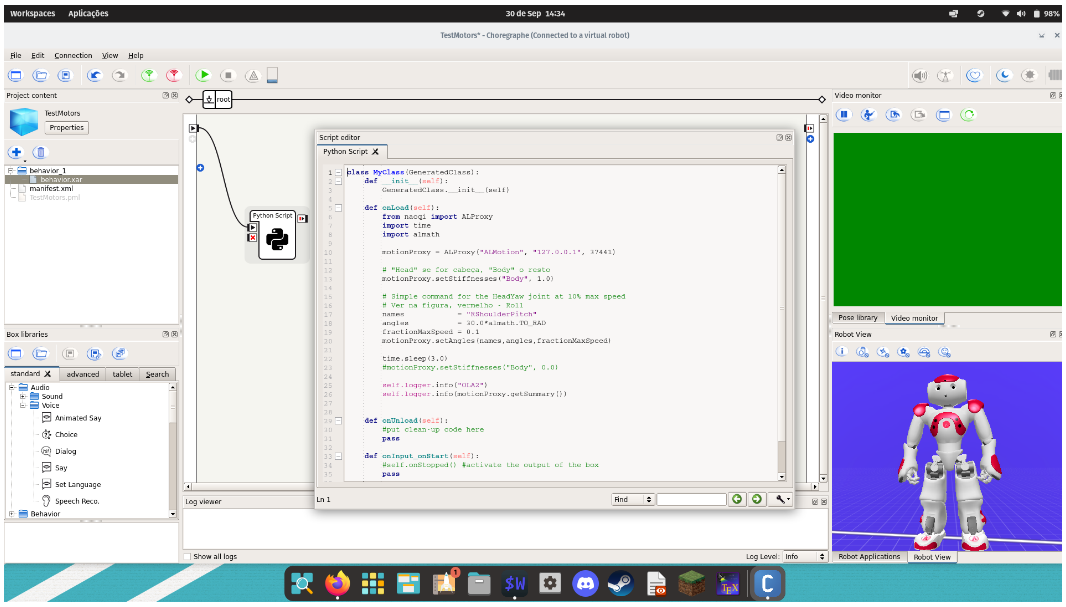
For more advanced behaviors, Choregraphe offers a visual, block-based programming environment. This drag-and-drop system enables the creation of intricate sequences using predefined blocks (called “boxes”) that cover a broad range of functionality:

- Basic Logic Blocks: Include fundamental programming elements such as conditionals, loops, and timers.
- Communication Blocks: Allow NAO to send and receive emails, transmit infrared signals, or manage its connectivity with Choregraphe and the internet.
- Audio Blocks: Enable speech synthesis, voice recognition, and the playback of pre-recorded sounds or phrases.
- Vision Blocks: Provide tools for facial and object recognition, as well as image capture and processing.
- Sensing Blocks: Offer access to NAO’s array of sensors, including temperature sensors, foot bumpers, fall detection, hand touch sensors, and posture monitoring.
- System Blocks: Allow users to retrieve system information such as the robot’s name, date/time, and log files.
- LED Control Blocks: Facilitate control of the robot’s LED indicators, including effects like blinking or color changes.
- Motion Blocks: Arguably the most powerful set, these allow for complex motion sequences like walking, dancing, or full-body coordination. Fine motor control is also possible through these blocks, making it easy to choreograph sophisticated behaviors.

For users who require deeper customization or wish to go beyond the predefined capabilities, Choregraphe allows inspection and modification of the Python code behind each block. Developers can copy this code into a template Python box, where they are free to extend the behavior or integrate external logic using standard Python programming practices.

This hybrid visual-code interface makes Choregraphe a remarkably powerful tool,not only for beginners exploring robot control for the first time but also for experienced developers building complex applications for the NAO platform.

In the following figure we can see what a normal use of Choregraphe may look like. In this specific image, we have loaded a program that was used to test the joints of the robot.3.4



```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        from naoqi import ALProxy
        import time
        import almath

        motionProxy = ALProxy("ALMotion", "127.0.0.1", 37441)

        # "Head" se for cabeça, "Body" o resto
        motionProxy.setStiffnesses("Body", 1.0)

        # Simple command for the HeadYaw joint at 10% max speed
        # Ver na figura, vermelho - Roll
        names            = "RShoulderPitch"
        angles           = 30.0*almath.TO_RAD
        fractionMaxSpeed = 0.1
        motionProxy.setAngles(names,angles,fractionMaxSpeed)

        time.sleep(3.0)
        #motionProxy.setStiffnesses("Body", 0.0)

        self.logger.info("OLA2")
        self.logger.info(motionProxy.getSummary())


    def onUnload(self):
        #put clean-up code here
        pass

    def onInput_onStart(self):
        #self.onStopped() #activate the output of the box
        pass
```

Figure 3.4: Normal Interface when using Choregraphe

3.6.2 Simulators Use - Webots

Webots provides a powerful and user-friendly environment for simulating robotic behavior, including full support for the Darwin-OP robot. To begin working with Darwin-OP in Webots, the most straightforward approach is to open one of the pre-existing example worlds that come bundled with the simulator. These example scenarios often correspond directly to the built-in demonstration programs provided with the real robot itself—such as basic walking, soccer, or object tracking,making the learning curve smoother if you're already familiar with the real hardware.

This direct equivalence allows users to prototype and validate behaviors in simulation that can later be deployed onto the actual robot with minimal adjustments. Moreover, Webots supports cross-compilation, which means code written and tested in simulation can be compiled and transferred directly to the robot to be run natively.

Exploring the Simulation Environment

Once an example world is loaded,such as the soccer demo world,you will see several nodes within the scene tree, each representing a component of the simulation. Among these, two nodes are particularly essential:

- WorldInfo: This node defines the fundamental parameters of the simulation environment. It includes values such as the physics engine settings, random seed (used for stochastic behaviors), gravity, time step granularity, and the coordinate system. Adjusting these allows for fine-tuning simulation realism or replicating specific conditions.
- Viewpoint: As the name suggests, this node controls the perspective from which the simulation is viewed. You can configure camera angles, zoom level, or even script dynamic camera behavior for better analysis during complex behaviors.

Other nodes are grouped and color-coded for ease of navigation and generally represent PROTO objects. These can include robots, static or dynamic obstacles, visual backgrounds, and other elements of the simulated environment. Webots' modular design allows users to modify these nodes or create new ones entirely, making it easy to tailor the simulation to specific experimental setups.

Writing and Deploying Code

Development can begin in two ways:

- Creating a new project, where you define a custom world and your own C++ (or Python, Java, etc.) controller files; or
- Modifying an example project, in which case Webots will prompt you to either overwrite the default files or save your modifications as a new, independent project directory.

The core logic typically resides in a file such as main.cpp, which can interact directly with the simulated robot's sensors, motors, and actuators through the Webots API. The simulator also provides access to advanced utilities like kinematic solvers, camera feeds, and physics interactions, allowing detailed and accurate development.

Cross-Compiling for the Real Robot

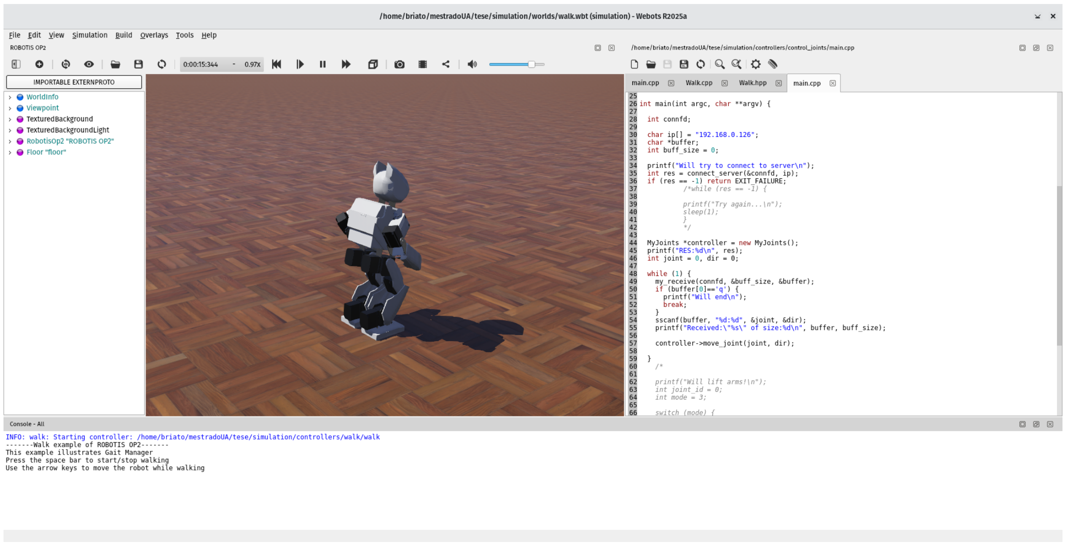
To deploy code developed in Webots to the actual Darwin-OP robot, cross-compilation is required. This process allows the source code written in a host system (typically x86_64) to be compiled into executable binaries compatible with the Darwin-OP's ARM-based architecture.

Here is a summarized procedure for this workflow:

- Obtain the Cross-Compilation Toolkit - Download the Darwin-OP cross-compilation package from Robotis GitHub repository or follow the link provided in the official documentation.
- Transfer to the Robot - Copy the toolkit to the robot under the /darwin directory via scp, USB, or any file transfer method.
- Build the Sources - After compiling the controller code on your local system, structure the directory to mirror that used in the simulator. Then transfer your source files to the robot, navigate to the proper directory, and compile directly on the robot using make.
- Reference Guide - A comprehensive and updated tutorial for this process is available at: `https://emanual.robotis.com/docs/en/platform/op/simulation/#transferring-the-sources`

This process allows seamless transition from simulation to real-world testing, accelerating development while preserving hardware longevity by minimizing unnecessary wear and tear during prototyping.

In the figure, we can see a world with a Darwin-OP robot in it. It encompasses a fase of the project where the simulator was used to test out motions for the robot.3.5



```
int main(int argc, char **argv) {

  int connfd;

  char ip[] = "192.168.0.126";
  char *buffer;
  int buff_size = 0;

  printf("Will try to connect to server\n");
  int res = connect_server(&connfd, ip);
  if (res == -1) return EXIT_FAILURE;
           /*while (res == -1) {

           printf("Try again...\n");
           sleep(1);
           }
           */

  MyJoints *controller = new MyJoints();
  printf("RES:%d\n", res);
  int joint = 0, dir = 0;

  while (1) {
    my_receive(connfd, &buff_size, &buffer);
    if (buffer[0]=='q') {
      printf("Will end\n");
      break;
    }
    sscanf(buffer, "%d:%d", &joint, &dir);
    printf("Received:\"%s\" of size:%d\n", buffer, buff_size);

    controller->move_joint(joint, dir);

  }
    /*

    printf("Will lift arms!\n");
    int joint_id = 0;
    int mode = 3;

    switch (mode) {
```

```
INFO: walk: Starting controller: /home/briato/mestradoUA/tese/simulation/controllers/walk/walk
-------Walk example of ROBOTIS OP2-------
This example illustrates Gait Manager
Press the space bar to start/stop walking
Use the arrow keys to move the robot while walking
```

Figure 3.5: Normal Interface when using Webots

3.7 Conclusions

However, during practical development, several critical limitations emerged when using the NAO robot.

One of the most impactful challenges was the difficulty in accessing and installing official software packages provided by Aldebaran. Due to the age of the specific NAO unit available for this work, an older-generation model, many of the required SDKs and development tools (such as naoqi-sdk, python-naoqi, and ALMotion/ALVideoDevice modules) were either deprecated, unavailable through official channels, or incompatible with current operating systems and Python versions. This significantly hindered the ability to maintain or extend the robot's functionality within a modern development environment.

Compounding this issue were hardware instabilities, more notably:

- it had eletro-mechanical problems with the right shoulder motor, which did not move at all;
- it could also not walk, as when we run the available macros to do so, he would turn off after barely flexing his knees, probably because of not getting enough eletrical corrent from the charger.

Due to all these motives, mainly the inability to walk, as well as some eventual dificulties with simulating both robots, the NAO robot was discarded from the project, and instead it was used 2 Darwin-OP. Nonetheless, the experience gained from working with NAO contributed to the broader understanding of humanoid robot interfaces, and helped inform the selection and configuration of the tools used in the final implementation.

As the NAO was eventually phased out and the project fully transitioned to using two Darwin-OP robots, the advantages of Webots became even more prominent, providing reliable and scalable simulation support for coordinated multi-robot tasks.

CHAPTER 4

Vision Framework

In collaborative robotic tasks, perception and motion must operate in close synergy to achieve reliable and coordinated action. This is especially critical for humanoid robots, where even small errors in perception can cascade into instability, failed manipulation, or loss of balance. In this chapter, we address the first of these two essential components: vision, leveraging both the onboard cameras of the robots and an external camera.

4.1 Initial Problem

The goal of this stage is to provide reliable perception of both the robots and the object using visual information. This involves not only detecting and identifying key elements in the scene but also estimating their orientation and relative positions. At this point, the focus is on perception rather than manipulation, ensuring that the robots have the spatial awareness needed for subsequent control and coordination tasks.

Key challenges addressed in this section include:

- Object Detection in Complex Environments: The object (an aluminum bar) presents difficulties due to reflections and similarities in color with the robots and floor. Robust detection requires the use of external and onboard vision, supported by artificial markers to improve reliability.
- Robot Orientation Estimation: Unlike simple detection, the robots' orientation must also be inferred. This is achieved through visual markers (e.g., strips or T-shaped patterns), which provide stable features to extract heading direction even under variable lighting.
- Fusion of External and Onboard Cameras: The external camera provides a global view of the scene, while the robots' onboard cameras deliver local, more detailed perspectives. Combining these sources ensures more robustness against occlusions and improves positional accuracy.

This section therefore focuses on the vision algorithms and filtering techniques used to extract accurate and stable positional data, forming the foundation for motion planning and control.

4.2 Framework

The vision framework is divided into two layers: code running on the central controller and complementary routines running directly on the Darwin-OP robots. This division ensures that both global and local perspectives are combined to provide reliable perception.

On the central controller, two main ROS packages form the backbone of the vision system:

- ExternalCamera: This package interfaces with an external overhead camera, continuously capturing images of the environment and publishing them to a ROS topic. It provides a global perspective, covering both robots and the object simultaneously.
- ProcessImage: For each robot in operation, a dedicated ProcessImage node is launched. This package processes the external camera feed, identifies the robot and the target object, and extracts key spatial information such as position and orientation. This information forms the basis for higher-level decisions, such as planning trajectories or aligning robots with the object, which will later be executed by the control framework.

In this configuration, the external camera provides the raw information that will be needed for multi-robot coordination, while the ProcessImage nodes refine said information into actionable data tailored to each individual robot. Together, they establish the perceptual foundation required for stable, collision-free approaches to the object.

On the Darwin-OP robots, vision operates in a more focused and situational manner. Once a robot has reached close range and is roughly aligned with the target, the central controller transmits the color to be tracked, and the robot switches into a local detection mode. In this mode, visual processing relies on the built-in ColorFinder module in combination with a slightly adapted version of the BallTracker module. Using these tools, the robot isolates the object in its field of view and determines its relative position with greater precision than the external camera can offer at short distances. From this perception, the robot derives the fine-grained motion adjustments needed and forwards them directly to its onboard control module, bypassing the central controller and minimizing communication delays.

In this setup, the onboard camera contributes the close-range accuracy and responsiveness that global vision alone cannot provide. By leveraging targeted vision modules, the robot maintains stable alignment and approach during the most delicate phase of interaction.

This dual-layered setup ensures robustness: the external camera offers global spatial awareness and prevents inter-robot collisions, while onboard vision provides precise, real-time adjustments during the final approach. Importantly, the central controller retains the ability to override or deactivate local autonomy, maintaining consistency within the overall coordination framework.

4.3 External Camera

In our framework, the external camera processing is structured around two ROS packages: ExternalCamera, responsible for acquiring and publishing the raw images, and ProcessImage, which interprets this data to extract meaningful information about the robots and the object.

The ExternalCamera node is responsible for interfacing with the overhead camera, configuring essential parameters such as frame rate, resolution, and image format, and then publishing the resulting image stream to a designated ROS topic. In this work, a single external camera was sufficient to cover the entire workspace, and therefore only one topic was required. Nevertheless, the design is flexible: multiple cameras could be integrated, each publishing to its own topic, thereby allowing independent or complementary visual streams to be associated with different robots or regions of the environment.

The ProcessImage node is responsible for interpreting the visual data published by the ExternalCamera. For each robot in the system, a dedicated ProcessImage instance is launched, allowing the perception pipeline to be tailored individually. The configuration for each robot is defined through a specific config file, which includes the color of the robot's "hat" and the target color, as well as additional parameters such as dilation/erosion iterations and other image-processing settings. This file can also specify fixed waypoints for the robot to approach and contains the corresponding robot's IP address on the network.

Using this configuration, the node identifies the robot and its designated object in the scene, estimates their relative orientation and distance, and formats the results into a concise output:

(IP) (Distance:Angle) (Pose) (Color)

where IP is the robot's network address, Angle represents the angular offset between the robot's heading and the object (0° meaning perfect alignment), Distance indicates the separation in pixels, Pose are the coordinates and orientation of the robot, and Color are the parameters that describe the target color. These structured outputs are then published to the appropriate topic, enabling the control framework to use them directly for trajectory planning and alignment.

By separating raw image acquisition from robot-specific interpretation, ProcessImage provides a flexible and modular bridge between global perception and targeted robot control.

4.3.1 Baseline Method

The first step in estimating the angle and distance between a robot and the object is to reliably identify both in the environment. Our initial approach for detection is intentionally simple and robust: the use of color filtering to isolate the relevant elements.

To achieve this, the image from the external camera is first converted from the RGB color space to HSV, which separates chromatic information (Hue) from intensity (Value) and saturation. This representation allows for more flexible filtering, since the Hue can be tuned to select the desired color, while adjustments to Value and Saturation help compensate for variations in illumination, reflections, and shadows.

Once the filter is applied, the pixels corresponding to the target color are grouped, and their centroid is computed to approximate the object's or robot's position in the scene. To reduce noise and improve the robustness of detection, morphological operations such as dilation and erosion are applied, producing cleaner, more coherent clusters.

Because the external camera is mounted overhead and oriented nearly perpendicular to the floor, the centroid of each cluster provides a good approximation of the object's true position on the ground plane. This simple pipeline offers a practical foundation for object detection, upon which more advanced methods can later be built.

While this baseline method already provides a reliable way to localize robots and objects on the ground, it is not without limitations. Variations in lighting, reflections from surfaces, or color similarities between different elements of the environment can still introduce errors. For this reason, several refinements and complementary strategies were later explored, which are presented in the following subsections.

4.3.2 Increase Visibility

One of the first challenges in visual detection was the difficulty of distinguishing the robots and the object from the environment. Both the Darwin-OP and the aluminum bar shared a metallic grey appearance, often reflecting light in ways that produced glare and blended them with the similarly colored floor. These conditions made color-based segmentation unreliable and led to inconsistent detection, particularly under varying illumination.

To illustrate this issue, we conducted an experiment using the baseline method. In this setup, the robot's orientation was approximated as the perpendicular to the width of the smallest bounding rectangle enclosing the largest detected centroid from the color filter. For the moment, we set aside the ambiguity of the rectangle's front–back symmetry (the "90° problem"), assuming the robot would be correctly oriented. In the figure 4.1, we can see one attempt to correctly obtain the distance and angle difference from the robot to the object, and in the plot 4.2, we can see that the results were unsatisfactory: the estimated angles lacked precision, varied considerably, and were not consistent across frames.

To address this limitation, we adopted a simple but effective visual enhancement strategy. High-contrast red paper strips were attached to each end of the aluminum bar. This modification made the bar stand out clearly in the camera feed, while the placement of two strips allowed the system to detect not only the bar's position more easily, but also its orientation at all times, as the two ends could now be independently identified.

A similar approach was used for the robots. To estimate their pose, colored paper markers were placed on the head, visible from the overhead camera. This allowed us to determine not only the robot's location but also its facing direction, information that is crucial for trajectory planning and coordination.

These minimal, low-cost visual markers provided a reliable baseline for object and robot localization during early development. They significantly improved detection stability without the need for complex feature extraction or sensor fusion techniques. For simplicity, and because at this stage only a single robot was used, we initially employed a red paper circle at the center of the bar rather than strips at the edges. Later refinements extended this idea by placing markers at both ends, enabling more robust orientation estimation.

In the figures 4.4 we can see how the improvements would look to the naked eye, and in 4.4 we can see the result of simple color filtering applied to the more vibrant colors.

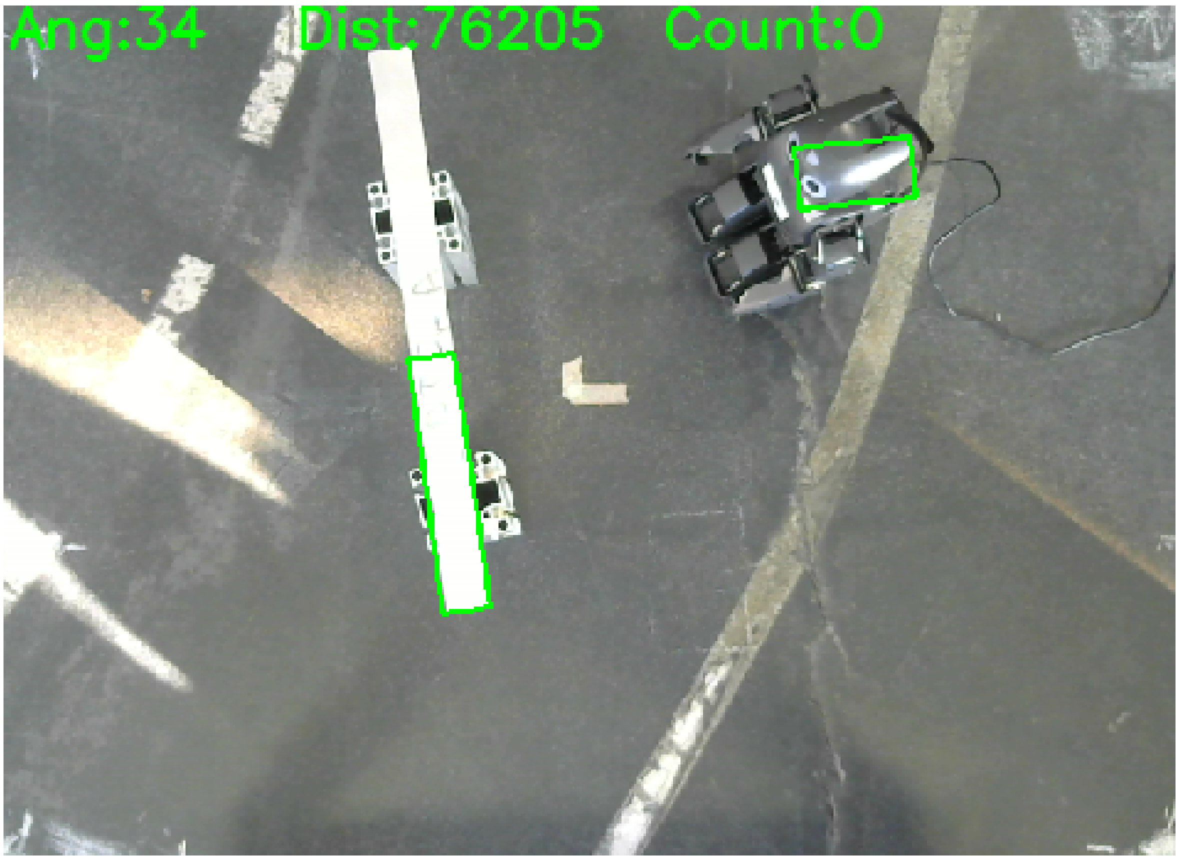


Figure 4.1: The result of the baseline method: Simply applying a color filter for the different shades of grey to differentiate between the robot, the object and the floor

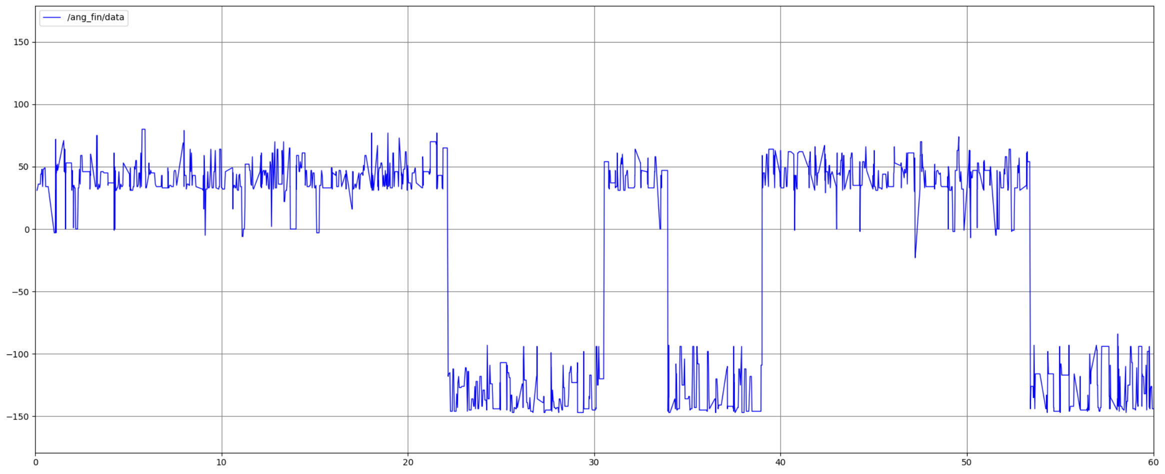


Figure 4.2: Variation of the angle between the robot and the object read in a interval of 60 seconds

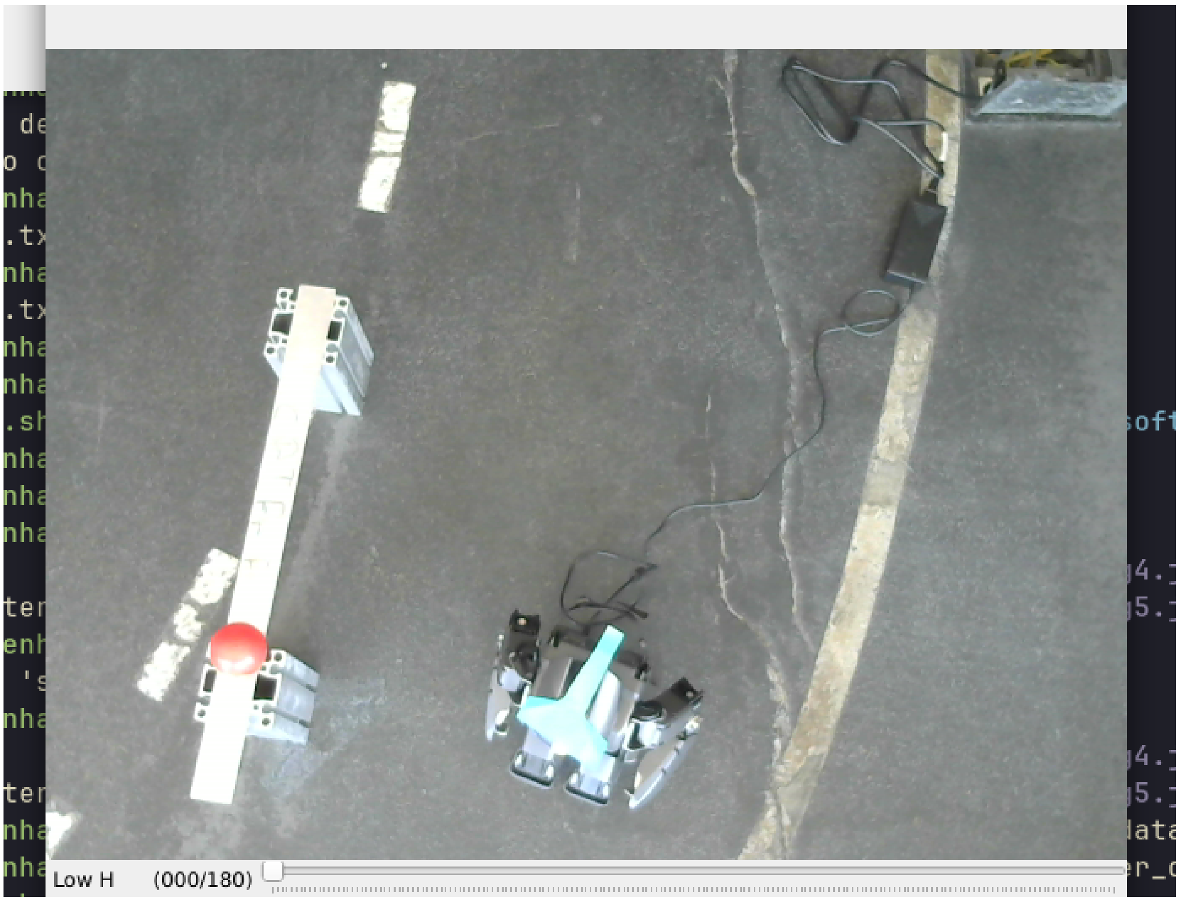
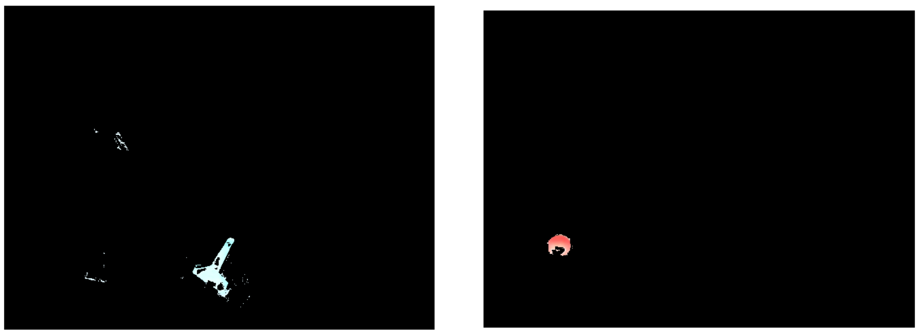


Figure 4.3: Colors applied to the object and the robot for easier detection



(a) Robot's Blue Marker **(b)** Object's Red Marker

Figure 4.4: Result of color filtering red and blue, obtaining the robot and object markers

4.3.3 Get Robot's Pose

Using a simple rectangular strip or elongated polygon to estimate the robot's orientation poses a significant limitation: such markers only allow us to determine directionality within a 0–180° range, lacking the ability to distinguish between forward and backward orientations.

To overcome this, we opted for a more descriptive visual marker:an oriented polygon in the shape of an isosceles triangle, which was placed on top of the robot's head. This triangle provides a clear directional cue, enabling us to determine not only the robot's heading but also its absolute orientation from 0° to 360°. Since the robot typically maintains its head aligned with its body's forward direction, this triangle serves as a reliable indicator of the robot's global pose.

This approach, however, is not without challenges. The robot's head is not a uniform surface, and during locomotion, it changes its height, inclination, and sometimes even alignment slightly. Moreover, non-uniform lighting conditions or partial occlusions can cause inconsistencies in how the triangle appears in the captured image. In such cases, the system may incorrectly interpret the marker, leading to abrupt orientation errors—for example, flipping from an angle β to $\beta + 180°$.

To reduce these inconsistencies, we implemented a temporal smoothing strategy. The system tracks recent orientation values within a small time window and detects outliers by comparing the current estimate with the recent history. If an angle measurement differs significantly from the last few values, it is flagged as potentially inaccurate and temporarily discarded until further consistent measurements confirm a new trend. This stabilization mechanism helps avoid erratic jumps in orientation caused by momentary occlusions or lighting artifacts, providing smoother and more reliable estimates.

To calculate the orientation of the triangle (and thus the robot), we identify two key geometric points on the triangle:

- Point A is the center of the minimum-area oriented bounding rectangle that fully encloses the triangle. This gives a good approximation of the triangle's geometric center and is usually close to its centroid, especially for symmetric shapes like the isosceles triangle.
- Point B is the center of mass (COM) of the triangle, calculated by treating each pixel within the triangle as having equal mass. Because of the triangle's shape (with the base wider than the height), the COM is typically located closer to the base than the geometric center.

By connecting Point B to Point A, we obtain a vector that aligns with the triangle's principal direction. While this vector provides a consistent estimate of orientation, we found that the bounding rectangle itself offered greater stability for determining the main axis, even tho it was not as precise. The AB vector was then used to disambiguate directionality, indicating which end of the rectangle corresponded to the robot's forward heading. In the figure 4.5 we can see a full representation of the method used. In this case, the corresponding rectangle and the AB line (pink) are perfectly aligned.

This method, combining simple geometric reasoning with temporal smoothing, proved to be both efficient and effective. It significantly improved the consistency and precision of

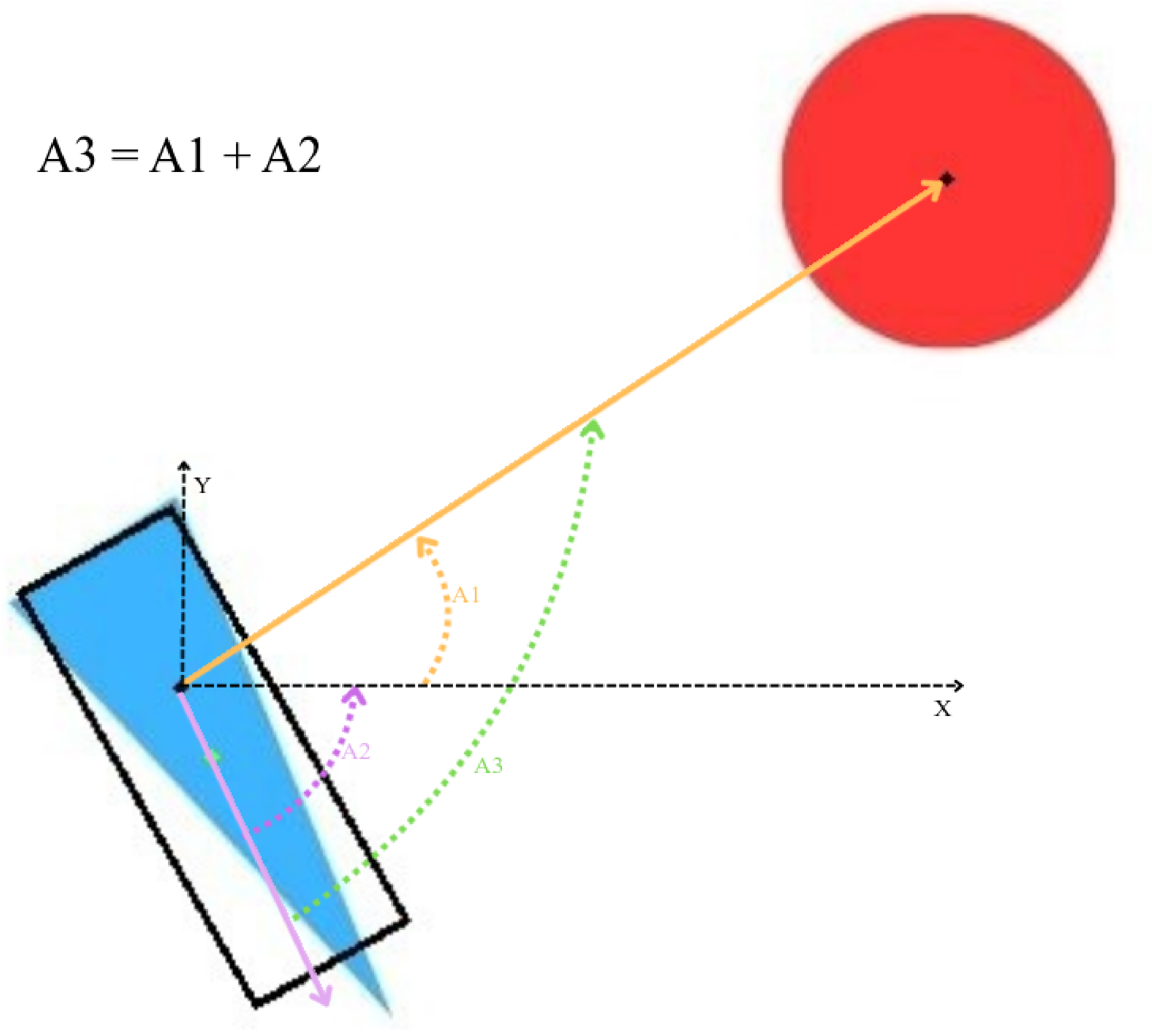


Figure 4.5: Representation of how the angle between the robot (with the marker on it's head), and the object is calculated

orientation estimation while keeping computational requirements modest.

As shown in the plot 4.6, although some fluctuations remain, the measured values are notably more stable and remain close to one another.

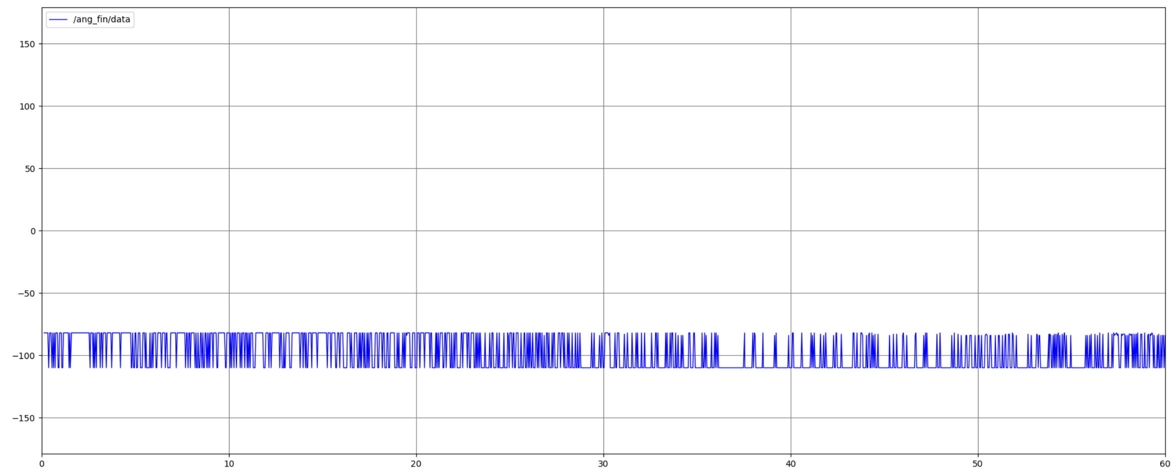
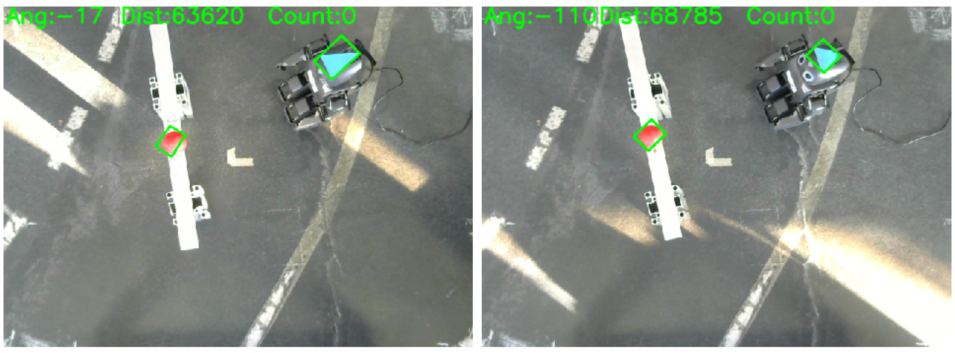


Figure 4.6: Plot of the values of the angle and the robot calculated in 60 seconds, using the adition of the blue triangle on top of the robot's head

4.3.4 Changing the polygon

While the isosceles triangle provided a theoretically sound and geometrically rich solution for orientation estimation, in practice it introduced additional complexities. Because it relied on precise shape recognition, the triangle marker proved sensitive to variations in lighting, partial occlusion, and perspective distortion. Even with temporal smoothing, the shape was sometimes fragmented or misidentified, particularly when the robot was in motion.

Although the method using the angle of the bounding box was semi-consistent, often switching only between two nearby values, these jumps could occasionally result in large orientation errors that effectively reset the robot's perceived heading. This instability is illustrated in the figure 4.7 below: the triangle appears in nearly identical positions, yet slight differences in illumination or head inclination cause the system to compute completely different orientation angles.



(a) Correct Read **(b)** Wrong Read

Figure 4.7: Triangle can help correctly read the angle between the robot and the object, but can also lead to an incorrect reading.

To improve robustness, we replaced the triangle with a simpler and more reliable visual marker: a pair of colored strips arranged in a T-shape. This configuration proved far easier to detect under varying conditions. The orthogonal lines of the T provided clear visual anchors, simplifying both segmentation and orientation calculation.

In this setup, the stem of the T indicates the backward direction of the robot, while the top bar confirms the marker's orientation. Because the T is asymmetric, it eliminates the 180° ambiguity that plagued symmetric shapes, while still maintaining computational simplicity.

This modification greatly increased tolerance to environmental noise, shadows, and small changes in the robot's position or head angle. As a result, orientation estimates became significantly more stable and accurate, an especially critical improvement when robots operated near each other and the object.

The shift from the triangle to the T-marker illustrates a key trade-off: sacrificing some geometric richness in favor of a solution that is simpler, more robust, and better suited to real-world conditions. The following plot 4.8 demonstrates this improvement, showing orientation values that remain nearly linear with minimal variation.

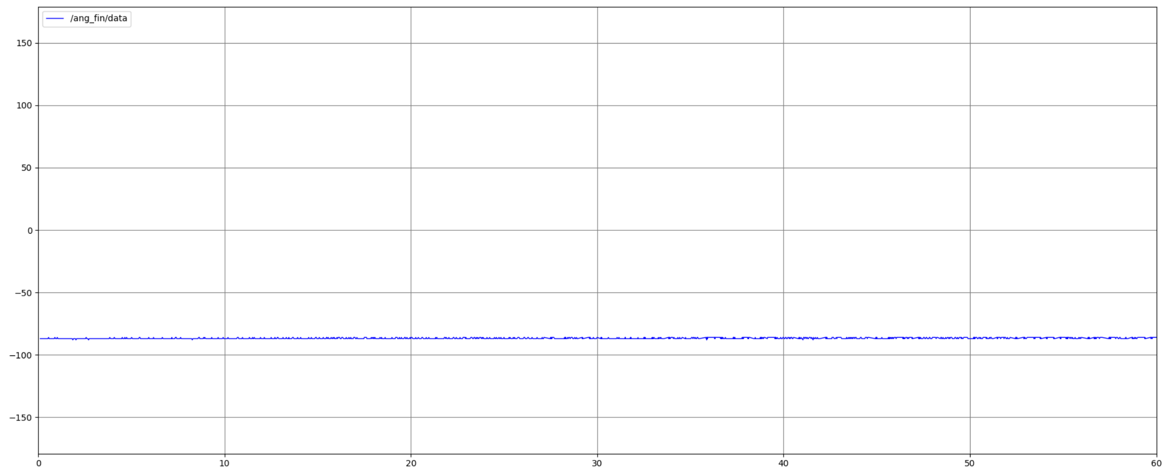


Figure 4.8: Plot of the values of the angle and the robot calculated in 60seconds

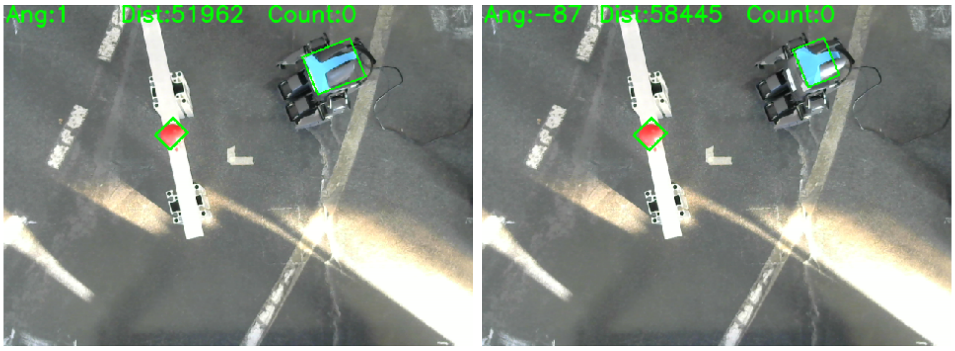
4.3.5 Create more contrast

The simple T-marker already provided a significant improvement in both consistency and precision of orientation detection. However, in certain situations parts of the T would become partially obstructed by the robot's "ears," or its visibility would be reduced due to reflections caused by the circular metallic surface of the robot's head where the strip was attached.

These issues sometimes produced orientation errors of approximately 90°, as the top bar of the T remained visible while the stem was partially lost. In such cases, the bounding rectangle enclosing the marker would align only with the bar, disregarding much of the stem, and the computed orientation angle would fail.

In the figure 4.9 , we can observe an example of this effect.

To further increase precision and robustness, we designed a simple paper "hat" that could be placed on the robot's head. The hat featured a darker decagonal base, on top of which the lighter T-marker was fixed. This configuration created higher contrast, reduced glare from the metallic head surface, and ensured that the T was better aligned for detection in most circumstances.



(a) Correct Read **(b)** Wrong Read

Figure 4.9: T shape can help correctly read the angle between the robot and the object, but can also lead to an incorrect reading.

While the smaller T that fit on the hat was occasionally subject to minor deviations—typically within ±10° due to lighting effects, these were not significant. The increased robustness and protection against large orientation errors outweighed this drawback, making the hat-mounted T-marker our preferred and final solution.

In the figure 4.10, we can see the end result of this system.

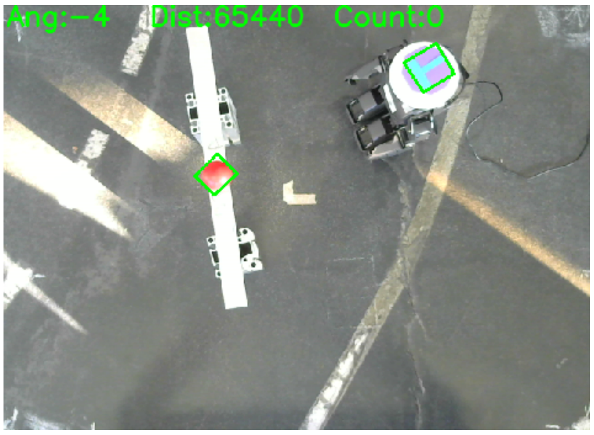


Figure 4.10: The robot with the "hat"

The following plot 4.11 illustrates the resulting orientation measurements, highlighting the improved consistency and the relatively small variations observed.

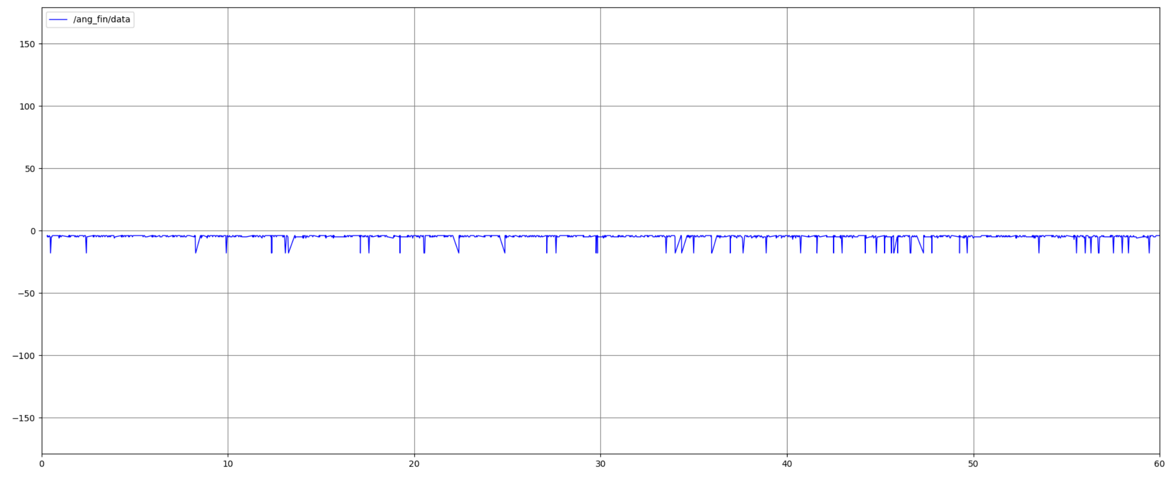


Figure 4.11: Plot of the values of the angle and the robot calculated in 60seconds

4.4 Internal Camera

While the external overhead camera provides a broad, global view of the workspace, its effectiveness decreases when a robot approaches the target object. At close distances, occlusions and limited resolution make external vision less reliable for fine alignment. For this reason, the Darwin-OP's internal camera is employed as a complementary perception source, enabling precise, local detection when the robot is near its goal.

This part of the framework was inspired by the BallFollowing/Football demo provided by Robotis, which showcases the use of onboard vision for real-time object tracking. The demo relies on two core modules—ColorFinder, which isolates pixels matching a predefined color, and BallTracker, which estimates the position of the detected object in the image. While the original implementation was designed for tracking a ball on the ground, our task required adapting the method for detecting and approaching an elevated object. With these modifications, the same lightweight and efficient approach could be reused, ensuring reliable close-range tracking while keeping computational requirements minimal.

4.4.1 Baseline

The Football demo, developed by Robotis, provides a practical showcase of the Darwin-OP's onboard vision and locomotion capabilities. In brief, the demo enables the robot to locate a ball, approach it, and ultimately execute a kick. With the exception of the kicking action itself, this baseline behavior closely matches the requirements of our autonomous mode and was therefore adopted as a foundation for our work.

At the start of the demo, the robot activates the ColorFinder module, which by default searches for red pixels in the images captured by the onboard camera. If no target is detected, the robot slowly pans its head from side to side in an attempt to bring the ball into view. Should this also fail, the robot halts and waits.

Once the ball is detected, the robot adjusts its head to align the centroid of the detected "red" region as closely as possible with the center of the camera's image. This ensures that the object remains in view while providing a reference for motion control. The head pan angle is interpreted as the horizontal offset of the ball, which dictates how much the robot should rotate its body to face it directly. The head tilt angle provides a rough estimate of distance: the lower the tilt required to see the ball, the closer it is to the robot's feet.

Using these two cues, the robot approaches the ball, continuously correcting its trajectory to keep the head pan near zero (object centered) and reducing the tilt as it gets closer. When both angles indicate that the object is directly in front and near the robot, and the centroid lies close to the center of the image (slightly below to account for the downward camera angle), the system determines that the ball is within kicking range. The robot then checks the pan value to decide which foot to use and executes the kick, after which it resumes searching for the ball.

This process works reliably and illustrates the effectiveness of lightweight onboard vision. More importantly, it demonstrates a control loop that is highly similar to the behavior desired for our autonomous mode: continuous perception-action cycles that guide the robot toward an object without requiring input from the external controller.

4.4.2 Adaptation for our case

When considering how to adapt the Football demo to our project, one advantage immediately stood out. In our framework, this autonomous mode is only activated when the robot is already close to and roughly aligned with the target object. This greatly reduces the risk of the robot failing to detect the object during the initial search, a major drawback of the baseline demo. With visibility ensured, we could focus on adjusting the perception-to-action loop for our task.

The first step was to evaluate whether the existing pan and tilt parameters, originally tuned for a ball on the ground, were suitable for detecting and approaching the aluminum bar.

- Pan adjustment: In the demo, the head pan is sufficient to guide the robot into a position where it can kick with either foot. In our case, however, grasping the bar requires both arms to be engaged simultaneously. If the robot approaches at a slight angle, one arm may miss the object entirely. To prevent this, we narrowed the acceptable pan range, ensuring the robot is more strictly aligned before attempting a grasp.
- Tilt adjustment: The tilt values in the baseline demo assume the ball lies on the floor. For an elevated object like the bar, this would result in the robot pushing the bar forward instead of approaching it correctly, as the projection in the image would misrepresent the distance. To account for this, we increased the minimum tilt angle considered "close enough," ensuring the robot only initiates the pickup when the object is at the appropriate height and range.

Finally, the kicking motion of the original demo was replaced by a pick-up sequence, specifically designed for lifting the bar. The details of how this motion was achieved will be discussed in the Control section.

With these targeted modifications, the Football demo was successfully repurposed into an effective autonomous mode for our project, capable of guiding the robot to reliably align with and grasp the object.

4.4.3 Integration into the Framework

Having made the necessary adaptations, we now have at our disposal a reliable Autonomous Mode that functions effectively at close range, precisely where the external camera becomes less reliable due to reduced accuracy or occlusions caused by the robot itself.

Nevertheless, this mode still suffers from limitations at longer distances, where it may fail to correctly identify the object. This can occur either because of background interference from similar colors or simply because the object lies outside the robot's field of view. For this reason, the transition to Autonomous Mode is not handled locally but instead triggered by the central controller, which bases its decision on information obtained from the ExternalCamera.

Activation is conditioned by two factors:

- Proximity: the robot must be within a predefined distance threshold of the object;
- Alignment: the robot's orientation relative to the object must fall within an acceptable angular range.

When both conditions are satisfied, and the robot is not picking up/putting down the object, the robot switches into Autonomous Mode. From this point onward, local vision processing ensures fine-tuned adjustments of orientation and distance, enabling the robot to prepare for stable manipulation of the object.

This hybrid approach leverages the complementary strengths of the two vision sources: the global perspective of the external camera for long-range navigation and collision avoidance, and the local precision of the onboard camera for short-range alignment and final approach.

4.4.4 Results and Observations

The use of this module proved to be highly effective.

In the following table 4.2, we can see that although the external module alone was usually not enough to place the robot in an ideal position to pick up the object, it would place it good enough to allow for the Autonomous Mode to precisely adjust as needed.

Table 4.1: Results of applying both the vision modules when the robot is in various positions. Each entry corresponds to (d, θ), where d is the distance(pixels) and θ is the angle (degrees)

Beginning	Before Autonomous Mode	Before Picking up Object
(24798, 70º)	(4721, 9º)	(2255, -3º)
(25310, -71º)	(4563, 10º)	(1897, 0º)
(87986, -90º)	(4890, 14º)	(2015, 5º)
(87899, 90º)	(4590, -7º)	(1987,-1º)
(134356, 176º)	(4988, 15º)	(2119,12º)

Table 4.2: Distance and Resulting Angle when the system passed to Autonomous Mode, and when decided it was close enough to pick up the object.

Although it occasionally exhibited minor overshooting, it consistently enabled the robot to approach the object with far greater reliability. This significantly reduced the risk of colliding with the object and causing it to fall, or stopping in a position unsuitable for initiating the pick-up motion.

In practice, the Autonomous Mode successfully addressed the main shortcoming of the external camera setup: its lack of precision at close distances. This limitation arose both from occlusion, where the robot's own body partially blocked the object, and from the inherent scale of pixel-based measurements, where small shifts in image coordinates correspond to disproportionately large changes in physical distance.

By contrast, the local onboard vision offered the fine-grained accuracy necessary to carry out the final alignment and approach. While it retained clear disadvantages in long-range perception and in maintaining awareness of the broader environment, it complemented the external camera perfectly by filling in this critical gap. Together, the two systems provided a balanced and robust perception framework across both global and local scales.

4.5 Final Conclusions

With this two-module solution, we established a framework capable of extracting the essential information from the environment to guide the robots reliably toward the target object. The modular design not only ensures clarity in responsibilities, separating global perception from local fine-tuning, but also provides a natural path for scalability. In principle, the framework can accommodate additional external cameras for wider coverage, or be extended to manage a larger number of robots with minimal adaptation. Despite retaining certain limitations, most notably the reliance on color filters, which can fail when confronted with background interference or changing illumination, and the persistent possibility of occlusions, the system proved sufficiently robust and dependable for the scope of this project. In practice, it consistently delivered the necessary perceptual accuracy to enable stable robot approaches and laid a strong foundation for subsequent work in cooperative object manipulation. Looking ahead, the framework could be further strengthened by reducing its reliance on handcrafted color filters, for instance through the integration of depth cameras or learning-based detection methods that are more resilient to environmental variability. Additionally, combining external and onboard vision with proprioceptive data from the robots themselves would create a richer, multi-modal perception system, ultimately enabling more reliable and adaptable collaboration in increasingly complex scenarios.

CHAPTER 5

Control Framework

Building upon the perception layer established through vision, the second fundamental component of collaboration is control. Whereas vision determines what is happening in the environment, control governs how the robots respond—translating spatial information into coordinated, stable, and purposeful motion.

5.1 Initial Problem

The goal of this stage is to transform the perception provided by the vision system into reliable robot motion. The robots must be capable of approaching the object individually, maintaining balance, and avoiding interference with one another, to prepare for the cooperative transport. This requires both macro-level coordination (high-level commands) and micro-level precision (joint control and posture adjustment).

Key challenges addressed in this section include:

- Macro vs. Micro Control: High-level actions such as walking forward or rotating must be combined with more fine-grained control of joints for tasks like aligning arms or shifting posture. Balancing these layers of control is essential for smooth execution.
- Postural Stability: Any upper-body movement, such as raising the arms in preparation for grasping, alters the center of mass. Ensuring stability requires synchronized control between the torso, arms, and legs.
- Collision Avoidance Between Robots: As both robots approach the same object, their paths must remain coordinated to prevent collisions or blocking each other's approach. The global perspective from the external camera supports this coordination.
- Action Synchronization Across Multiple Robots: Both robots must align the timing and execution of their actions (e.g., walking pace, rotations, or initiating a lift) to ensure smooth coordination and prevent mismatched behaviors.

This section presents the control strategies and communication mechanisms used to ensure that each robot can move safely and accurately toward the object, preparing the ground for cooperative manipulation in later stages.

5.2 Framework

When discussing the control framework, a structure emerges that closely parallels the vision system: it is composed of two primary components. The first runs on the central controller, implemented through ROS packages, while the second executes directly on the Darwin-OP robots themselves.

On the central controller, the ROS packages serve as the high-level decision-making layer. They aggregate information from the external camera and status updates from each robot, then plan actions at a macro scale. This includes tasks such as determining the appropriate direction of movement, deciding when to initiate the Autonomous Mode, and handling communication to transmit commands to the robots.

On the Darwin side, the control framework is organized into specialized modules for robot control, walking, communication, and motion/joint actuation.

- The robot controller module acts as the central node, interpreting incoming commands from the communication module and delegating them to the appropriate subsystems.
- The communication module, as the name implies, manages the exchange of data, receiving high-level commands from the central controller while also reporting back real-time information about the robot's internal state.
- The walking module encapsulates locomotion, including forward steps, rotations, and autonomous decisions based on feedback from the internal camera when Autonomous Mode is active.
- Finally, the motion/joint control module executes predefined motions such as the object pick-up sequence, and, when required, enables fine-grained joint adjustments as instructed by the central controller. Although this last functionality was less frequently used in our project, it remains an important capability of the framework.

This modular division between centralized planning and distributed low-level execution ensures that high-level coordination is preserved across robots, while each individual robot maintains the autonomy and precision needed for reliable action.

5.3 Central Controller

5.3.1 ControlDarwinNode

This ROS node serves as the interface between perception and action: it receives positional data from the ProcessImage nodes and translates it into executable commands for the robots. The information is published in the format <(IP) (Distance:Angle) (Pose) (Color)>, where IP specifies the target robot, Distance represents the distance in pixels to the object, Angle indicates the angular deviation relative to the robot's forward orientation, Pose has the coordinates and current orientation of the robot, and Color has the HSV parameters that define the target color of the robot. Based on this input, the node determines the appropriate command and transmits it to the corresponding robot over a TCP/IP connection.

In addition to automated commands derived from vision, this node also supports user-specified commands. These can be published to a dedicated ROS topic in the format

<IP Command>, where Command follows the structure of a standard robot command, as described below. This feature provides flexibility for testing, debugging, and manually overriding automated behavior when necessary.

Commands

The transmitted commands fall into two main categories: simple commands and complex commands.

- Simple commands operate at a low level, directly adjusting the position of a single joint or a symmetrical pair of joints. They are typically used for fine-tuning, such as correcting posture, adjusting arm alignment, or carrying out precise calibration steps.
  - Format: j:JointID:BothJoints:Increment
  - j: indicates that the command refers to a joint action rather than a complex motion.
  - JointID: identifies the joint to be moved.
  - BothJoints: specifies whether the command should apply to only the selected joint (1), to both symmetrical joints (2), or to a single joint in absolute mode (0).
  - Value/Increment: defines either the incremental adjustment to apply, or the absolute value for the joint position (in the BothJoints = 0 case).
- Complex commands coordinate multiple joints to perform higher-level motions, such as walking, rotating, or executing the pick-up sequence.
  - Format: MotionID:TimeToExecute
  - MotionID: a single character that identifies the intended action (e.g., 'f' for walking forward, 'f' for turning left, 'x' for initiating the pick-up motion).
  - TimeToExecute: the duration of the action in seconds. In practice, this value is usually capped at 2 seconds to avoid error accumulation and ensure stable execution.

By unifying vision-driven inputs with user-level overrides, this node provides a robust and flexible control mechanism. It ensures that robots can respond both to autonomous perception and to operator interventions, while maintaining a consistent communication structure across the system.

Macro Control

After gathering information from the different ProcessImage nodes, this control node determines the most appropriate action to send to each robot. Its primary priority is alignment: the robot is first instructed to rotate until its orientation relative to the object falls within a predefined threshold. This ensures that, once the robot is close enough, the object is already within the field of view of its internal camera.

Only after achieving adequate alignment does the node issue forward-walking commands, gradually reducing the distance to the target. If at any point the alignment drifts outside the acceptable range, rotation commands are reissued to correct orientation before further advancing.

This cycle continues until the robot reaches a distance threshold from the object. At that point, the node commands the robot to switch into Autonomous Mode, handing over fine-grained control to onboard vision. As with all complex motions, activation of the Autonomous Mode is constrained to a maximum of two seconds, providing regular opportunities for macro-level corrections and preventing uncontrolled drift.

5.4 Darwin Control

Before addressing the control architecture and motion strategies, it is important to outline the basic operational procedures of the Darwin-OP platform. This section provides a brief overview of essential setup tasks, such as powering the robot, establishing connections, and accessing its internal systems, to help future users quickly configure the platform and focus on higher-level development rather than initial troubleshooting.

To operate the Darwin-OP robot, one can either connect it directly to a power adapter or use a rechargeable lithium-polymer battery, which typically provides up to 30 minutes of autonomous operation under standard loads. Swapping the battery is straightforward: with the robot connected to the charger, the battery can be safely removed and replaced with a charged one before disconnecting the power cable, ensuring uninterrupted readiness.

The LED indicator located on the robot's forehead serves as a simple yet effective diagnostic tool. By default, a green LED signifies that the robot is powered on and idle, while a yellow LED indicates that an active process or user-defined program is running. Upon startup, the robot automatically enters demo mode unless configured otherwise, and this is generally accompanied by a yellow LED.

To power down the robot, the user may press the chest button for approximately 3 seconds to initiate a graceful shutdown. In the case of a system lock-up or emergency, holding the button for 7 seconds will trigger a forced shutdown. During either procedure, the forehead LED will begin flashing green. Once it ceases blinking and remains off, it is safe to press the power button again and disconnect the power source or remove the battery.

Because the Darwin-OP runs a full Linux operating system, it can function as a standalone computer, and makes it a lot easir to transfer information with. Users may connect a monitor via HDMI, a keyboard via USB, and directly interact with the system without needing a remote terminal. This is especially useful when setting up Wi-Fi for the first time. Rather than connecting via Ethernet, identifying the robot's IP address using a network discovery tool (e.g., Fing or Angry IP Scanner), and configuring Wi-Fi manually through SSH, the user can instead access the GUI or terminal locally and configure networking settings with ease.

This direct access also proves handy for quick debugging or minor code adjustments. The robot comes pre-installed with essential development tools, including the Vi text editor and g++ compiler, enabling on-the-fly code editing and compilation directly on the device.

5.4.1 Key-Hardware Components of Darwin-OP

After configuring the basic connection and setup of the Darwin-OP, the next step involves accessing its key hardware components, namely, the onboard camera and servo motors. These elements form the foundation of both perception and actuation in the system. The following sections describe how to use the built-in API to interface with the robot's camera to acquire visual data, and present two distinct approaches for controlling its servos, each suited to different levels of precision and autonomy.

Use Camera

To capture images using the Darwin-OP's built-in camera, the following steps are typically performed:

- Instantiate a LinuxCamera object – This class handles all operations related to image acquisition. It provides access to image size, format, color channels, and internal buffers.
- Convert the raw RGB data to JPEG format – The Darwin-OP framework includes a built-in function (jpeg_utils::compress_rgb_to_jpeg) to efficiently compress the image into JPEG, which reduces the size for storage or transmission.
- Store or transmit the image – The resulting JPEG buffer can either be saved locally on the robot or sent to an external controller (e.g., a laptop) using an SSH connection or another communication protocol.

Control Joints Movement

To interact with the servos (i.e., Dynamixel motors), there are two main approaches: One approach is to directly comunicate with them using the CM730 API, as the CM730 module provides direct read/write access to the motor registers.

This method offers fine-grained control and communicates directly with the servo hardware, allowing faster and more precise interaction when real-time feedback is important.

- To read a joint's current position (from hardware), use: cm730->ReadWord(joint_id, MX28::P_PRESENT_POSITION_L, value_ptr, 0);
- To write a target angle (goal position), use: cm730->WriteWord(joint_id, MX28::P_GOAL_POSITION_L, value, 0);
- To change specific control parameters, such as the proportional gain (P-Gain), use: cm730->WriteByte(joint_id, MX28::P_P_GAIN, 8, 0);

This other approach is to use the MotionManager/MotionModule modules that also come built-in to more easily integrate with the robot's motion modules and ensures synchronization across subsystems (e.g., walking, head tracking). However, it does not provide real-time feedback from the actual servo hardware.

- Create and initialize a Motion Manager instance with the preferred configurations;
- Create and initialize a MotionModule instance, and add it to MotionManager, like for example the Walking/Head module, or a custom one created by us.
- We acess the m_Joint parameter of the instance, and call the SetValue(joint_id, value) function to set a desired joint angle.
- Call Process() on the MotionManager instance to apply the changes to the servos.
- Use GetValue(joint_id) the same way we used SetValue(...) to retrieve the most recently commanded value for that joint.

Comparison

While both approaches allow for joint control, they serve different purposes:

- MotionManager is ideal for high-level, coordinated control. It tracks commanded positions internally but does not reflect the physical motor position, which may differ due to mechanical resistance or power issues.
- CM730 allows direct access to the Dynamixel registers, enabling real-time feedback and low-level control, independent of the high-level motion system.

This makes CM730 particularly useful for diagnostics, sensor fusion, and precise feedback loops, while MotionManager is better suited for structured motion execution and integration with walking or gesture frameworks. In the end, in our project we opted to use mostly CM730.

5.4.2 Improve Locomotion

One recurring challenge during locomotion arose from the robot's default foot design. The Darwin-OP's soles are made of smooth metal plates, which, although mechanically robust, provide little traction on standard laboratory flooring. As a result, the robot frequently experienced minor slippage during normal walking and much more noticeable sliding during in-place rotations. This effect was exacerbated on surfaces that were not perfectly level, often creating asymmetries in rotational performance depending on the turning direction.

Such slippage was particularly problematic for the control framework, as it introduced inconsistencies when attempting to compute reliable average angular and linear velocities. The resulting displacements varied significantly across repetitions, making it difficult to predict motion outcomes with precision.

To quantify the extent of this problem, a series of experiments was conducted. The robot was instructed to rotate for 2 seconds in both directions, and the resulting translations and rotations were measured and averaged. The procedure was then repeated with 4-second rotation intervals. A similar approach was applied to forward walking motions, testing both 2-second and 4-second durations.

These experiments not only allowed us to evaluate the repeatability of the robot's movements, but also provided average velocity estimates. Such measurements were later used to parameterize higher-level control strategies, enabling the robot to perform longer motions more reliably.

Table 5.1: Results of motion experiments without any soles. Each entry corresponds to (x, y, θ), where x and y are translations (cm) and θ is rotation (degrees).

Command	**Trial 1**	**Trial 2**	**Trial 3**
(Rotate right 4s)	(-6, 8,-30º)	(-8, 11, -30º)	(-11, 5, -20º)
(Rotate right 2s)	(-2, 3, -30º)	(-8, 11, 20)	(-6, 7, -30º)
(Rotate left 4s)	(0, -9,220º)	(0, -6,270º)	(3, -7,220º)
(Rotate left 2s)	(1, 3,70º)	(3, 9,120º)	(5, 8,90º)
(Forward 4s)	(11, 40,25º)	(13, 44,5º)	(21, 36,50º)
(Forward 2s)	(1, 28,40º)	(4, 27,45º)	(0, 30, 35º)

The results reveal clear asymmetries and inconsistencies in the Darwin-OP's locomotion performance. When executing rotational commands, the robot consistently turned more effectively to the left than to the right. In fact, the measured rotations to the right were often very small, sometimes close to zero, despite the command duration being sufficient to expect a much larger angular displacement. Moreover, the angular values obtained for 2-second and 4-second rotations were surprisingly similar, rather than scaling proportionally; ideally, a 4-second command should result in roughly double the rotation of a 2-second one. This lack of linearity makes it difficult to predict motion outcomes solely based on command duration.

Translation during rotation further compounded the issue. Although some lateral displacement is not unusual for bipedal robots, the expectation is that it should remain minimal and relatively consistent. In practice, however, the robot exhibited significant variability in its translational drift while turning, making it harder to model and compensate for in the control layer. This inconsistency posed a greater challenge than the mere presence of drift itself, as it undermined the repeatability of motion primitives.

Forward walking presented fewer asymmetries, but it was still affected by drift and angular deviations. The robot rarely moved in a perfectly straight line, and deviations grew more pronounced over longer walking intervals. Nevertheless, walking motions were somewhat more predictable than rotations, which proved to be the most problematic primitive.

Despite these shortcomings, the experiments provided valuable average velocity estimates that could be integrated into the control framework. Although the primitives themselves lacked perfect repeatability, their mean behavior was sufficiently predictable to be combined with visual feedback, ensuring that higher-level tasks such as object approach and alignment remained feasible.

To address the inconsistencies observed in locomotion, a physical modification was introduced to the robot's design.

A 2 mm layer of neoprene rubber was affixed to the soles of the Darwin-OP's feet with the goal of increasing friction against the ground, as we can see in the figure 5.1. This adjustment aimed to reduce unwanted slippage and thereby improve the repeatability and stability of the robot's movements.

The addition of the rubber layer had a significant impact on performance. Although minor asymmetries between left and right rotations persisted, and the relationship between 2-second and 4-second motion intervals was not perfectly linear, the overall variability was substantially reduced. Rotations became more consistent, with measured angular velocities showing far less deviation between trials, while forward walking showed improved stability and reduced drift.

In essence, the neoprene soles enhanced the robot's interaction with the floor, mitigating the most problematic effects of the original smooth metal design. The following results illustrate this improvement, demonstrating smaller discrepancies across trials and more predictable motion outcomes compared to the unmodified configuration.

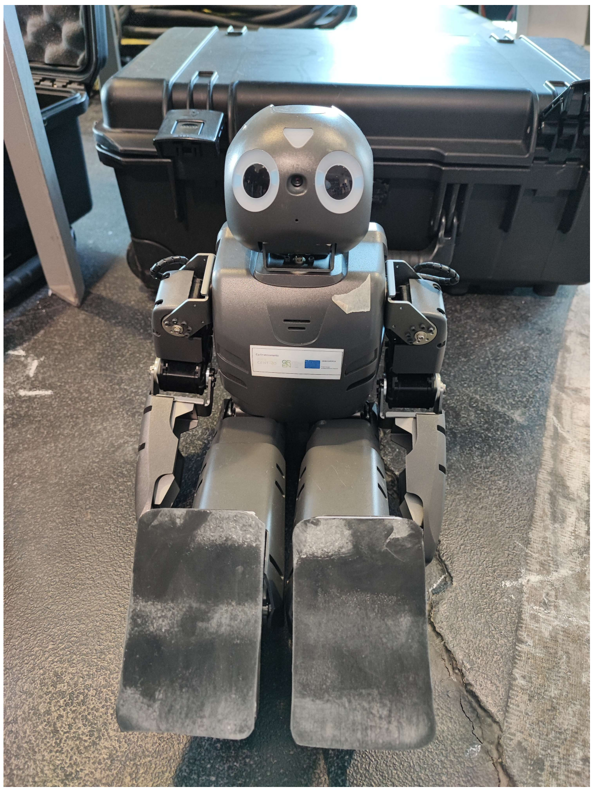


Figure 5.1: Picture of the Darwin-OP with the neoprene rubber feet.

Table 5.2: Results of motion experiments with neoprene soles. Each entry corresponds to (x, y, θ), where x and y are translations (cm) and θ is rotation (degrees).

Command	**Trial 1**	**Trial 2**	**Trial 3**
(Rotate right 4s)	(-21, 14, -105º)	(-22, 15, -120º)	(-26, 15, -120º)
(Rotate right 2s)	(-6, 14, -75º)	(-11, 14, -75º)	(-4, 11, -70º)
(Rotate left 4s)	(14, 4, 140º)	(22, 4, 140º)	(17, 6, 130º)
(Rotate left 2s)	(5, 10, 65º)	(4, 11, 90º)	(10, 18, 120º)
(Forward 4s)	(-3, 63, -15º)	(-4, 55, 0º)	(6, 56, 0º)
(Forward 2s)	(1, 30, 10º)	(-2, 26, 20º)	(-1, 15, 10º)

Another material tested was carpet.

As with the neoprene rubber, small pieces of carpet were fixed to the soles of the robot's feet with the goal of increasing friction against the ground surface. The intention was the same: to reduce slippage during both walking and rotation, and thereby improve the consistency of the robot's locomotion. The following results summarize the robot's performance with this modification.

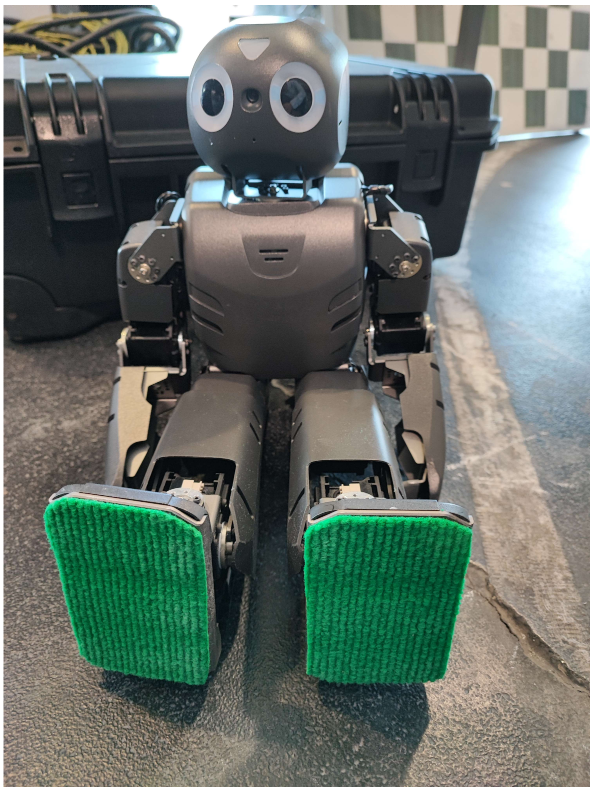


Figure 5.2: Picture of the Darwin-OP with the green carpet feet.

Table 5.3: Results of motion experiments with carpet soles. Each entry corresponds to (x, y, θ), where x and y are translations (cm) and θ is rotation (degrees).

Command	**Trial 1**	**Trial 2**	**Trial 3**
(Rotate right 4s)	(7, -13, -130º)	(3, -12, -140º)	(5, -13, -110º)
(Rotate right 2s)	(5, -6, -65º)	(4, -6, -90º)	(7, -5, -90º)
(Rotate left 4s)	(0, 8, 140º)	(-1, 8, 140º)	(0, 9, 145º)
(Rotate left 2s)	(4, 1, 110º)	(5, 4, 83º)	(4, 0, 90º)
(Forward 4s)	(45, -2, -10º)	(50, -2, -20º)	(42, 1, -5º)
(Forward 2s)	(27, 2, -5º)	(22, 1, -10º)	(28, 1, -15º)

The introduction of carpet soles further improved the consistency of the robot's locomotion compared to both the bare metal and the neoprene rubber modifications. When examining the rotation trials, the results show not only reduced variability between repeated tests, but also a significant mitigation of the asymmetry observed earlier. In particular, the tendency of the robot to rotate more effectively to the left than to the right, which was still somewhat visible with neoprene, became much less pronounced with carpet. Forward walking also benefited: translation values were much closer across repeated trials, and the resulting angular deviations were smaller and more stable, typically staying within $\pm 15°$.

Overall, the carpet soles provided the highest level of consistency among all tested configurations. While minor deviations remain unavoidable due to the nature of humanoid locomotion on flat surfaces, this modification offered the most balanced performance, making it a strong candidate for reliable motion planning and control.

One other advantage the friction of these new soles provide us is the ability to more easily increase the opening angle of the legs of the robot. Before, increasing this value too much would be risky, as it greatly increases the risk of slipping sideways, which in our environement, with some inclinitation and not much friction, would not be very safe. With this new soles however, we can increase this range without as much risk. This is beneficial because opening a bit the legs allows us to lower our center of mass, while still mantaining it in the pelvis area of the robot.

5.5 Darwin Framework

Similar to the rest of the project, the control code running on the Darwin-OP is also organized into distinct modules, each designed to encapsulate a specific functionality while remaining flexible enough for adaptation.

One of the most important components is the Communication module. This module was created to establish and manage the TCP/IP communication channel between the robots and the central controller. Although the Darwin already comes with a built-in API that enables this functionality, our custom implementation provided greater flexibility and allowed us to tailor the communication layer to our specific needs. It includes functions for creating and managing a TCP/IP server, connecting to that server, and sending or receiving messages reliably. Because of this modular design, the Communication module could also be imported by the ControlDarwin ROS package, enabling code reuse both on the robot and on the central controller.

Another key addition is the MyJoints module, developed to simplify and streamline the use of the Darwin's API for low-level joint control, especially when interacting with the CM730 interface. Instead of directly writing to memory via CM730 functions, which can be cumbersome, this module introduces an abstraction layer that retains the benefits of direct memory access while improving usability. It also provides higher-level functions for executing custom complex motions, such as transitioning into the posture required to pick up the object.

In addition to these custom-built modules, two modules were adapted from Robotis's official Darwin-OP framework. The first is MyWalking, a modified version of the standard Walking module. This module controls the robot's joints during locomotion but was specifically adapted in our work to allow the arms to move independently of the walking pattern. This capability was essential for tasks involving object transportation, as it enabled the robot to carry an object while maintaining stable gait.

The second adapted module is MyWalker, derived from the original BallFollower module. Our version was extended to handle both autonomous behaviors and high-level commands from the central controller. When operating in Autonomous Mode, MyWalker uses the centroid of the detected object in the robot's onboard camera feed to decide its movement (e.g., alignment and approach). In normal operation, it interprets incoming commands such as "forward" or "turn right" and maps them to appropriate walking actions. This design leverages the MotionManager framework of the Darwin, ensuring smooth execution of locomotion primitives.

Finally, the MyControl module acts as the primary orchestrator of the onboard control system. It coordinates the other modules, using the Communication module to exchange information with the central controller, while delegating motion execution to either the MyJoints or MyWalker modules depending on the command received. By serving as the integration point, MyControl ensures a coherent flow from high-level decisions to low-level actuation.

5.5.1 Changing gait parameters

As the MyWalking mode was done with the Walking official Robotis module for walking as a baseline, we could still use the tools developed for said module in ours. One of those tools was the Walking Tuner. As the name implies, this was used to be able to configure parameters for the Darwin-OP's gait when walking. Some of the parameters included the period of the steps, the height they should reach at the maximum, arm swing gain, step forward/back ratio, hip height, etc. All of these parameters can make the gait more smooth in different situtations, and it was very usefull in our project as well. As the robot needs to carry some weight in it's arms, it was useful to use the parameter Hip Pitch, to rotate the torse slightly backwards, in hope of balancing with the object better. We also lowered the hip height during walking, as well as the step height, lower the center of gravity and increase stability. We increase the Y-offset (distance between feet - left/right) slightly as well, thanks to the feet soles talked about previously, to slightly decrease the height of the COM. Lastly, our change to the MyWalking module is similar to the parameters Arm Swing Gain, which determines how much should the arms move to help balance during locomotion. In our case, this value will be 0, and the arms will also have a different default value.

As the MyWalking module was developed using the official Robotis walking controller as its foundation, we were able to maintain compatibility with several of the tuning and diagnostic tools provided by Robotis. Among these, the Walking Tuner proved to be particularly valuable. As its name suggests, this tool allows the user to configure and fine-tune various gait parameters, enabling the robot to achieve smoother and more stable locomotion under a wide range of conditions.

The Walking Tuner exposes a range of low-level parameters that directly influence gait dynamics, including step period, maximum foot lift height, arm swing gain, step forward/backward ratio, hip height, and others. Careful adjustment of these variables can significantly improve balance, efficiency, and stabilit, especially when the robot operates under atypical conditions such as carrying additional payloads or performing cooperative tasks.

In the context of this work, tuning these parameters was essential to compensate for the additional weight distribution introduced by the object carried in the robot's arms. For instance, the hip pitch angle was slightly adjusted backward to counterbalance the forward-shifted center of mass caused by the payload. We also lowered the hip height and reduced the step height, both of which help decrease the overall center of gravity and improve static and dynamic stability during locomotion. Similarly, the Y-offset (lateral distance between the feet) was increased slightly, a change made feasible by the improved traction provided by the modified foot soles described earlier, further contributing to a lower and more stable center of mass.

Finally, while the original walking module uses arm swing gain to dynamically balance the robot during motion, in our implementation this parameter was effectively disabled (set to zero) since the arms are used for carrying the object and must remain stable. Instead, we defined a new default arm posture optimized for object transport, preventing unwanted oscillations that could compromise grip or balance.

These adjustments, though subtle, had a significant cumulative effect on walking stability. The robot's gait became more predictable and resistant to disturbances, allowing it to maintain balance more effectively while manipulating objects, a critical requirement for collaborative tasks.

5.5.2 Kinematics

To facilitate the process of moving the Darwin-OP into specific and repeatable positions, such as picking up/putting down the object, kinematic modeling was employed—more specifically, inverse kinematics (IK) on the MyJoints module. Instead of manually tuning individual joint angles, IK provides a more intuitive way of defining motions by specifying desired end-effector positions and allowing the solver to compute the corresponding joint configurations.

For the lower body, inverse kinematics was applied to the legs in order to control the position of the hips relative to the ground. To solve the inverse kinematics problem, we further simplified the system to a planar RRR arm model, as our primary objective was to adjust the vertical position while maintaining a constant value along the forward–backward axis. The lateral (right–left) displacement was not considered critical in this context, since the robot's center of mass would remain aligned with the pelvic region regardless of small deviations in that direction, assuming that both legs would make simetrical movements. By setting a target hip height, we were able to more easily configure stable postures, ensuring consistency during walking sequences and when transitioning into manipulation poses. This not only simplified the tuning process but also provided a systematic way of compensating for small variations in ground contact.

For the upper body, IK was developed to generate smoother and more natural trajectories for the arms when reaching toward the object. In this case, we kept the Shoulder Roll joint constant, modifying the problem into a classic RR planar arm configuration. This abstraction introduced a minor loss of precision due to slight misalignments in the robot's physical structure, specifically, the elbow joint sits slightly lower than the shoulder joints in the neutral position. However, this offset was minimal (less than 2 cm) and therefore considered negligible for the purposes of our calculations, allowing us to safely proceed with the simplified model. Rather than moving each joint independently, the IK formulation allowed the hands to follow a continuous path in Cartesian space, improving the accuracy of grasping while maintaining balance. This was particularly important for coordinated pickup motions, where both arms must move symmetrically and remain synchronized with the rest of the body's posture.

By incorporating inverse kinematics into both leg and arm control, the robot's motions became more adaptable and easier to configure, reducing the time needed for trial-and-error tuning while increasing the repeatability of key actions.

5.6 Cooperation

5.6.1 Synchronization

As the framework for controlling each robot individually was already established, the development of the cooperative control framework was designed as an extension of that foundation. This new layer builds upon the existing communication and motion control architecture to enable synchronized, collision-free operation between multiple humanoid robots.

In this configuration, a new coordination module was introduced. Unlike the single-robot version, which only needs to receive the distance and direction of a robot relative to the target object, this module processes the positions and orientations of all participating robots simultaneously. This design choice was crucial for two reasons: first, to accurately predict and prevent potential collisions between robots; and second, to ensure that both agents approach the object in a consistent and coordinated manner. Relying solely on directional data, without spatial context, would be insufficient for maintaining safe and synchronized trajectories.

Beyond spatial coordination, this module also addresses the challenge of temporal synchronization. Due to hardware limitations, the internal clocks of the Darwin-OP robots occasionally reset, making time-based synchronization unreliable. Additionally, since each robot operates on its own onboard computer, the central controller has limited access to its internal scheduling and thread management, further complicating real-time coordination.

To mitigate these issues, a lightweight synchronization strategy was implemented:

- Instead of transmitting commands to the robots immediately after computation, the central controller stores them temporarily in a queue managed by the coordination module.
- At fixed, predefined intervals, the module releases the queued commands to all robots simultaneously, ensuring that actions begin in near-perfect temporal alignment.

This mechanism provides a practical form of synchronization, reducing inconsistencies caused by communication latency or unsynchronized execution. It also complements the pre-existing design constraint in which every motion command is limited to a maximum duration of two seconds, ensuring that both robots operate within predictable time windows. Together, these strategies create a robust foundation for synchronized dual-robot control, enabling cooperative behaviors such as simultaneous approach, lifting, or transport of shared objects.

5.6.2 Colision Avoidance

Once temporal synchronization between robots was achieved, the next step was to prevent potential collisions during simultaneous motion. Although numerous sophisticated algorithms exist for multi-agent trajectory planning, such as potential fields, velocity obstacles, or predictive trajectory optimization, we opted for a lightweight, deterministic approach consistent with the overall philosophy of our system.

This solution leverages the same mechanism used for synchronization. Since all commands are queued before being sent to the robots, the system can perform a preliminary trajectory validation step to ensure that no conflicts exist between the planned motions of the agents.

The process begins with the synchronization queue, which temporarily holds all trajectory commands awaiting execution. Each robot can have at most one pending trajectory in this queue. A secondary validation queue is then maintained for trajectories confirmed to be collision-free. Before a trajectory is promoted to this validation queue, it undergoes a simple but effective set of checks:

The starting and ending points of each trajectory are evaluated against those of all other robots, with a safety threshold applied to account for positional uncertainty. The paths between these points are tested for intersections to ensure that the motion of one robot does not cross into the predicted path of another.

This approach remains computationally efficient because, within our framework, all robot commands correspond to simple primitives, either linear translations or in-place rotations. For linear motions, trajectory intersections can be evaluated directly by testing the overlap of line segments between start and end points. For rotations, the collision threshold is slightly increased, since in practice the robot's base does not rotate around a perfect fixed point and may drift slightly due to surface friction.

If a trajectory passes validation, it is transferred from the synchronization queue to the validated queue, from which it will later be executed. Trajectories that fail validation remain in the synchronization queue until conditions change or the conflict is resolved. This two-tiered system ensures that only collision-free, time-aligned commands are sent to the robots, allowing the control framework to maintain safety and coordination without relying on heavy computational models.

5.7 Conclusion

Together, these modules form a cohesive control architecture that balances modularity, adaptability, and efficiency. By extending and customizing existing Robotis frameworks while introducing new components tailored to the project's requirements, the system ensures robust communication, reliable locomotion, and seamless integration with both macro-level commands from the central controller and local autonomy within the robot. This layered approach not only facilitated the implementation of cooperative tasks in this work but also provides a scalable foundation for future extensions, such as adding new behaviors, refining motion primitives, or integrating more advanced decision-making strategies. Some of the changes helped improve the stability of the robots on both locomotion and when executing motions in place. However, they were not enough, and we were not able to achieve cooperative object transportation with the 2 robots.

CHAPTER 6

Conclusions and Future Work

6.1 Summary of Achievements

This section highlights what was accomplished throughout the thesis. You can structure it either chronologically (by chapter) or by topic (Vision, Control, Experiments). Points to include: Successful development of a two-layer vision framework combining external and onboard cameras. Implementation of a modular control system with distributed processing between the central controller and the robot. Integration of inverse kinematics to improve motion precision and repeatability. Improvements in locomotion stability through hardware adaptation (e.g., rubber or carpet soles). Validation of system performance through experiments involving autonomous object approach and manipulation preparation. (Optional) End this section with a small reflection on how the modular and scalable design supports future multi-robot extensions.

Throughout this dissertation, a complete framework for the perception and control of humanoid robots was developed, enabling them to autonomously approach and prepare for the manipulation of a shared object. The work combined both global and local perception strategies with modular control architectures, resulting in a flexible system capable of integrating multiple robots and cameras with minimal reconfiguration.

In the vision subsystem, two complementary approaches were implemented: an external camera providing global environmental awareness, and onboard cameras offering precise, short-range perception. This dual-vision setup effectively mitigated the limitations inherent to each method individually, improving positional accuracy at long range while maintaining fine alignment close to the object. A series of visual markers and filtering techniques were designed to overcome issues of reflectivity and color similarity, ensuring reliable detection of both robots and objects even under suboptimal lighting conditions.

In the control subsystem, the project established a robust communication framework based on TCP/IP, enabling synchronized command execution between the central controller and each robot. Custom modules were implemented for motor actuation, walking, and motion control, while the inclusion of an autonomous local control mode allowed each robot to operate independently once near its target. The integration of inverse kinematics in this subsystem also improved motion control and repeatability. Experiments confirmed the system's effectiveness in reducing errors in approach and alignment, as well as improving motion stability through gait parameter tuning and mechanical enhancements such as modified foot soles.

Overall, this work successfully demonstrated a modular, extensible architecture for humanoid robot coordination, serving as a solid foundation for future research in collaborative manipulation and multi-robot control.

6.2 Discussion and Limitations

Provide a critical assessment of the system's performance. Discuss what worked well and what limitations remain. Points you can include: The framework's modularity and scalability as a strength. The dependence on color filtering and its limitations under varying lighting conditions. The occlusion problem when robots or objects block the external camera's view. Precision constraints of the Darwin-OP hardware and internal camera resolution. Challenges in achieving perfect synchronization between robots during cooperative actions. This section demonstrates that you understand the boundaries of your work — a key aspect for a strong conclusion.

While the proposed framework achieved stable and reliable performance in enabling humanoid robots to detect, approach, and align with an object, several limitations and challenges remain that highlight opportunities for improvement.

From a vision standpoint, the system's reliance on color-based filtering proved both effective and restrictive. Although the use of color markers significantly improved detection robustness, this approach remains sensitive to variations in lighting, reflections, and background color similarities. Under changing illumination or when the markers were partially occluded, the detection accuracy could decrease, affecting subsequent localization and control. Additionally, since the external camera operated from a fixed position, depth perception was limited, which could lead to small spatial estimation errors, especially when the robots or the object moved outside the optimal visual range.

In terms of control, the modular architecture provided a clear structure for coordinating motion commands, but some challenges arose from synchronization and mechanical constraints. Because each Darwin-OP robot runs on an independent onboard controller, ensuring perfect timing alignment between them was difficult, particularly given occasional clock resets and minor network delays. This could result in slight desynchronization between the robots' actions, especially during dynamic or long-duration sequences. Although we were able to slightly increase the stability of the robots while executing motions in place or during locomotion, it was not enough to allow for cooperative object transportation.

Finally, while the current system handled single-object approach tasks effectively, scalability and adaptability remain open challenges. The framework has the potential to accommodate additional robots or sensory inputs, but this would require further optimization in communication efficiency, real-time coordination and better control modules. Similarly, while the control system supports preprogrammed and modular motions, it lacks adaptive feedback mechanisms capable of responding to unexpected disturbances or dynamic object movement.

Despite these limitations, the framework established a strong experimental foundation. The results demonstrated that with minimal sensing and computational resources, stable and coordinated humanoid behavior can be achieved, paving the way for more advanced implementations that integrate learning-based control, sensor fusion, and collaborative manipulation.

6.3 Future Work

The work developed in this dissertation opens several promising directions for future research, spanning both perception and control. Each of the proposed components, vision, communication, and motion coordination, can be further refined and extended to enhance the overall robustness and autonomy of the system.

In terms of vision, future work could focus on improving the accuracy and reliability of the perception modules. The current approach, based primarily on color filtering, can be expanded through the integration of additional classical computer vision techniques such as contour detection, edge-based tracking, or background subtraction to mitigate the effects of lighting variation and partial occlusions. Alternatively, a more modern route would involve adopting deep learning-based detection models, such as convolutional neural networks or YOLO-style architectures, to allow for more robust, generalizable object recognition. Such methods would reduce the need for manual configuration of color parameters while maintaining real-time processing capabilities.

Moreover, future research could explore sensor fusion between external and onboard cameras, combining global and local perspectives in a more unified framework. This could involve using visual–inertial odometry or integrating depth sensors to achieve better 3D localization and orientation estimation in dynamic environments.

Regarding control, one major avenue for expansion lies in exploring heterogeneous robot cooperation, involving robots of different sizes or morphologies. This would introduce additional challenges in maintaining balance, synchronizing movement, and distributing forces when manipulating or transporting shared objects. Addressing these challenges would contribute to a deeper understanding of adaptive coordination in humanoid teams.

Further improvements could also be made by introducing learning-based control strategies. Reinforcement learning or imitation learning could be used to optimize gait parameters, improve balance under load, and enable the robots to autonomously refine their cooperative behaviors through experience. Integrating such adaptive methods would reduce the need for manual tuning and could lead to smoother, more energy-efficient motion.

Finally, the overall system could benefit from a transition toward more decentralized coordination, where each robot maintains partial autonomy while still collaborating under a shared objective. This would increase scalability and resilience, allowing additional robots to be seamlessly integrated into the cooperative framework.

References

[1] A. Fitriana, K. Mutijarsa, and W. Adiprawita, «Color-based segmentation and feature detection for ball and goal post on mobile soccer robot game field», Cited by: 20, 2017. DOI: 10.1109/ICITSI.2016.7858232. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85015937857&doi=10.1109%2fICITSI.2016.7858232&partnerID=40&md5=5a4f10fc90d5fc208c6864839a02eabc.

[2] L. Xie and C. Deng, «Object detection of nao robot based on a spectrum model», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10956 LNAI, pp. 327–338, 2018, Cited by: 0. DOI: 10.1007/978-3-319-95957-3_36. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85051853711&doi=10.1007%2f978-3-319-95957-3_36&partnerID=40&md5=bb647a594d9217215f6bd08c2ed92b80.

[3] J. Bruce, T. Balch, and M. Veloso, «Fast and inexpensive color image segmentation for interactive robots», Cited by: 347, vol. 3, 2000, pp. 2061–2066. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0034446417&partnerID=40&md5=328ddff81578b6d2f8af7acd279858d6.

[4] H. Pylkkö, J. Riekki, and J. Röning, «Real-time color-based tracking via a marker interface», Cited by: 6, vol. 2, 2001, pp. 1214–1219. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0034860877&partnerID=40&md5=18de2fad372023ad8491859412fa0348.

[5] U. A. Manawadu, S. Keito, and N. Keitaro, «Object recognition and pose estimation from rgb-d data using active sensing», Cited by: 2, vol. 2022-July, 2022, pp. 165–170. DOI: 10.1109/AIM52237.2022.9863241. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85137705393&doi=10.1109%2fAIM52237.2022.9863241&partnerID=40&md5=70a63a455842098105a098bb4f9305f9.

[6] Ç. Yetişenler and A. Özkurt, «Multiple robot path planning for robot soccer», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3949 LNAI, pp. 11–23, 2006, Cited by: 10. DOI: 10.1007/11803089_2. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33746725929&doi=10.1007%2f11803089_2&partnerID=40&md5=a7439e12f3c9cacef756134d24725179.

[7] S. Mondal, N. F. Sharon, K. M. Tabassum, U. H. Muna, and N. Alam, «Development of a low-cost real time color detection capable robotic arm», Cited by: 2, 2023. DOI: 10.1109/ICCIT60459.2023.10441038. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85187402024&doi=10.1109%2fICCIT60459.2023.10441038&partnerID=40&md5=b54a5195158ee60e0f5a1906565b9af6.

[8] D. Noh, J. Choi, and S. Baek, «Artificial marker-aided localization for service robots in visually repetitive environments», Cited by: 0, 2025. DOI: 10.1109/ICCE63647.2025.10929959. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105006548569&doi=10.1109%2fICCE63647.2025.10929959&partnerID=40&md5=8c40977b61a17ce7dd3d6cf14920972a.

[9] D. Prado, S. Morais, J. V. Camargos, *et al.*, «Navigation system for a service robot for robocup@work», Cited by: 0, 2025. DOI: 10.1109/CROS66186.2025.11064857. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105012119337&doi=10.1109%2fCROS66186.2025.11064857&partnerID=40&md5=e5d384e04f3a221de611c93bd07e3657.

[10] C. Wang, Y. Zhang, D. Dong, X. Liu, J. Li, and H. Liu, «Research on target detection algorithm based on nao robot», 2023, pp. 146–150. DOI: 10.1109/ICIPCA59209.2023.10257679. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85174426192&doi=10.1109%2fICIPCA59209.2023.10257679&partnerID=40&md5=4755dadcc0b953b5150e22d2adc17252.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, «You only look once: Unified, real-time object detection», Cited by: 43526; All Open Access, Green Open Access, vol. 2016-December, 2016,

pp. 779–788. DOI: 10.1109/CVPR.2016.91. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84986308404&doi=10.1109%2fCVPR.2016.91&partnerID=40&md5=79a23b9cc271b6f314eea447fb88cc7d.

[12] T. Zhong, Y. Gan, Z. Han, H. Gao, and A. Li, «A lightweight object detection network for industrial robot based yolov5», 2023, pp. 4685–4690. DOI: 10.1109/CAC59555.2023.10449979. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85189290516&doi=10.1109%2fCAC59555.2023.10449979&partnerID=40&md5=95fa6d6976ab6bf1b7a7eac2e84f41d4.

[13] Q. Ding, E. Zhang, Z. Liu, X. Yao, and G. Pan, «Text-guided object detection accuracy enhancement method based on improved yolo-world», *Electronics (Switzerland)*, vol. 14, no. 1, 2025. DOI: 10.3390/electronics14010133. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85214531846&doi=10.3390%2felectronics14010133&partnerID=40&md5=b6ad8ee445a108ca188814d048701a69.

[14] S. Biswas, A. Nandy, and A. K. Naskar, «Object detection with yolo model on nao humanoid robot», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14301 LNCS, pp. 495–502, 2023, Cited by: 0. DOI: 10.1007/978-3-031-45170-6_51. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85177813545&doi=10.1007%2f978-3-031-45170-6_51&partnerID=40&md5=b8d4969aaf720feda32b298f75bdcb03.

[15] F. Sun, C. Wang, T. Zheng, and H. Liu, «An improved object detection method based on nao robot», 2023, pp. 1184–1188. DOI: 10.1109/ICPECA56706.2023.10076034. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85152223728&doi=10.1109%2fICPECA56706.2023.10076034&partnerID=40&md5=f3300146f130b36016aa3693eb45e310.

[16] A. Efendi and C.-Y. Huang, «Robot arm technology in detection and manipulation in smart pharmacies: A review», *International Journal of Humanoid Robotics*, vol. 22, no. 5, 2025, Cited by: 0. DOI: 10.1142/S0219843625300016. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105014099676&doi=10.1142%2fS0219843625300016&partnerID=40&md5=f0aa8a59b5de8fdf56b3acf863e6186b.

[17] X. Wang, Y. Tang, X. Liu, J. Wang, J. Cao, and R. Sun, «Research on robot target classification and localization based on improved mask r-cnn», *Concurrency and Computation: Practice and Experience*, vol. 37, no. 21-22, 2025, Cited by: 0. DOI: 10.1002/cpe.70247. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105013458391&doi=10.1002%2fcpe.70247&partnerID=40&md5=8f12ecae6ffdc52f520d1d05962477ac.

[18] H. Ge, P. Sun, and Y. Lu, «A new dataset, model, and benchmark for lightweight and real-time underwater object detection», *Neurocomputing*, vol. 651, 2025, Cited by: 1. DOI: 10.1016/j.neucom.2025.130891. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105010556519&doi=10.1016%2fj.neucom.2025.130891&partnerID=40&md5=a6e8bd55e155b722e770bada71da4626.

[19] J. Duan, L. Zhuang, Q. Zhang, J. Qin, and Y. Zhou, «Vision-based robotic grasping using faster r-cnn–grcnn dual-layer detection mechanism», *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 239, no. 6-7, pp. 950–964, 2025, Cited by: 2. DOI: 10.1177/09544054241249217. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85193755729&doi=10.1177%2f09544054241249217&partnerID=40&md5=719984620396dc659901a6ff1fdcb15d.

[20] M. M. Abo-Zahhad and M. Abo-Zahhad, «Real time intelligent garbage monitoring and efficient collection using yolov8 and yolov5 deep learning models for environmental sustainability», *Scientific Reports*, vol. 15, no. 1, 2025, Cited by: 2; All Open Access, Gold Open Access. DOI: 10.1038/s41598-025-99885-x. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105004443624&doi=10.1038%2fs41598-025-99885-x&partnerID=40&md5=dcb0f3261701776dfdf24d443a61a80d.

[21] E. Kee, J. J. Chong, Z. J. Choong, and M. Lau, «Enhancing intelligent robot perception with a zero-shot detection framework for corner casting», *Electronics (Switzerland)*, vol. 14, no. 9, 2025, Cited by: 0; All Open Access, Gold Open Access. DOI: 10.3390/electronics14091887. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105004862027&doi=10.3390%2felectronics14091887&partnerID=40&md5=f312072f81a44e3f3732ada7c035c7b8.

[22] M. T. R. Sahed, S. Rufsun, M. T. I. Aronno, and M. S. R. Chowdhury, «Object classification by a 7-dof robotic arm using mobilenet-ssd and feedback position mapping», Cited by: 0, 2025. DOI: 10.1109/QPAIN66474.2025.11171666. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-105019047005&doi=10.1109%2fQPAIN66474.2025.11171666&partnerID=40&md5=16d518583f352c7bc383b677b19c19e4.

[23] D. Manju, B. Gollapalli, P. Sudheer Benarji, K. Pooja, and A. Made, «Performance comparison of mobilenet ssd and yolo v4 in object detection», Cited by: 0, 2024, pp. 141–147. DOI: 10.1109/ICCCMLA63077.2024.10871895. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85219582430&doi=10.1109%2fICCCMLA63077.2024.10871895&partnerID=40&md5=7a07875ac22da125a85d186d632bb0d2.

[24] P. Sengaphone, J. M. De Leon, G. L. Augusto, *et al.*, «Implementation of single shot multibox detector (ssd) algorithm for object detection», Cited by: 1, 2024, pp. 702–707. DOI: 10.1109/ICBIR61386.2024.10875914. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-86000020999&doi=10.1109%2fICBIR61386.2024.10875914&partnerID=40&md5=636df38299516b548e0bad0de3dc1f10.

[25] V. Kumar, V. Goel, A. Amoriya, and A. Kumar, «Object detection using ssd», Cited by: 3, 2023, pp. 743–748. DOI: 10.1109/ICAC3N60023.2023.10541704. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85195777608&doi=10.1109%2fICAC3N60023.2023.10541704&partnerID=40&md5=0f5d99ae02f546e0dfaa1591c1e2e16a.

[26] T. Xu, K. Kühnlenz, and M. Buss, «A view direction planning strategy for a multi-camera vision system», Cited by: 4, 2008, pp. 320–325. DOI: 10.1109/ICINFA.2008.4608018. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-54249136743&doi=10.1109%2fICINFA.2008.4608018&partnerID=40&md5=21469df1da7bad5b832abd842fc4d681.

[27] Y. Aizawa, T. Suzuki, and K. Kobayashi, «Improvement of multiple robots' self-localization by using perspective positional information», Cited by: 0, vol. 2017-November, 2017, pp. 776–780. DOI: 10.23919/SICE.2017.8105662. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85044169947&doi=10.23919%2fSICE.2017.8105662&partnerID=40&md5=4eec47e627b4074eb1b35d799278a71f.

[28] C.-H. Chang, S.-C. Wang, and C.-C. Wang, «Vision-based cooperative simultaneous localization and tracking», Cited by: 24, 2011, pp. 5191–5197. DOI: 10.1109/ICRA.2011.5980505. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84871683054&doi=10.1109%2fICRA.2011.5980505&partnerID=40&md5=b95a0a16f418c48a375f01261299071e.

[29] M. R. Verbryke and C. L. R. McGhan, «Dual-arm manipulation with a humanoid robot using a modular control architecture», Cited by: 1, 2018. DOI: 10.2514/6.2018-5209. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85056194530&doi=10.2514%2f6.2018-5209&partnerID=40&md5=91ba15ed91e723eb5a52e41824741664.

[30] O. Khatib, L. Sentis, and J.-H. Park, «A unified framework for whole-body humanoid robot control with multiple constraints and contacts», *Springer Tracts in Advanced Robotics*, vol. 44, pp. 303–312, 2008, Cited by: 72. DOI: 10.1007/978-3-540-78317-6_31. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-39449112411&doi=10.1007%2f978-3-540-78317-6_31&partnerID=40&md5=2bca3206b99fa5891fefd173b3cc8f87.

[31] N. Gorges, A. J. Schmid, D. Osswald, and H. Wörn, «A framework for creating, coordinating, and executing skills on a humanoid robot», Cited by: 3, 2007, pp. 385–391. DOI: 10.1109/ICHR.2007.4813898. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-67649684027&doi=10.1109%2fICHR.2007.4813898&partnerID=40&md5=f375cadb223f8249e4f3b1e9526d1763.

[32] Z. Jiang, Y. Wang, S. Wang, S. Bi, and J. Chen, «Motion planning and control with environmental uncertainties for humanoid robot», *Sensors*, vol. 24, no. 23, 2024. DOI: 10.3390/s24237652. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85211814628&doi=10.3390%2fs24237652&partnerID=40&md5=3e03fa28e25c8ca91323fe54e79fb7de.

[33] D. Lim, M.-J. Kim, J. Cha, and J. Park, «Mob-net: Limb-modularized uncertainty torque learning of humanoids for sensorless external torque estimation», *International Journal of Robotics Research*, vol. 44, no. 1, pp. 96–128, 2025. DOI: 10.1177/02783649241260428. [Online]. Available: https://www.

scopus.com/inward/record.uri?eid=2-s2.0-85213523833&doi=10.1177%2f02783649241260428&partnerID=40&md5=b0f3757108df745de1683b390cc85bbb.

[34] K. Kim, J.-Y. Lee, S. Kim, *et al.*, «Coordinated task execution by humanoid robot», Cited by: 2, 2009, pp. 3496–3503. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77951114286&partnerID=40&md5=9443324003982b9510e9d396b5fcc138.

[35] T. Asfour, D. Ly, K. Regenstein, and R. Dillmann, «Coordinated task execution for humanoid robots», *Springer Tracts in Advanced Robotics*, vol. 21, pp. 259–267, 2006, Cited by: 6. DOI: 10.1007/11552246_25. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33845398249&doi=10.1007%2f11552246_25&partnerID=40&md5=d6cab41ee9ba55ea7b797ab9d3b61c90.

[36] M.-W. Han and G. Timofte, «Control of a humanoid robot based on the zmp method», 1 PART 1, Cited by: 0, vol. 17, 2008. DOI: 10.3182/20080706-5-KR-1001.3205. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-79961018626&doi=10.3182%2f20080706-5-KR-1001.3205&partnerID=40&md5=72b5db13ecc435bfdb4636ffaf6f527f.

[37] E. Taşkiran, M. Yilmaz, Ö. Koca, U. Seven, and K. Erbatur, «Trajectory generation with natural zmp references for the biped walking robot suralp», Cited by: 19, 2010, pp. 4237–4242. DOI: 10.1109/ROBOT.2010.5509792. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77955788907&doi=10.1109%2fROBOT.2010.5509792&partnerID=40&md5=4246fb310ded9bda6133bc3063497fbd.

[38] M. Vukobratović, B. Borovac, V. Potkonjak, and M. Jovanoviç, «Dynamic balance of humanoid systems in regular and irregular gaits: An expanded interpretation», *International Journal of Humanoid Robotics*, vol. 6, no. 1, pp. 117–145, 2009, Cited by: 11. DOI: 10.1142/S0219843609001668. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-70349392820&doi=10.1142%2fS0219843609001668&partnerID=40&md5=ca18e01731500ca98870f3593c855573.

[39] M. VUKOBRATOVIĆ and B. BOROVAC, «Zero-moment point — thirty five years of its life», *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 157–173, 2004, Cited by: 1882. DOI: 10.1142/S0219843604000083. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85104874228&doi=10.1142%2fS0219843604000083&partnerID=40&md5=5d7045aecb1be5ff0a1cb97f8d17291f.

[40] O. Kurt and K. Erbatur, «Biped robot reference generation with natural zmp trajectories», Cited by: 20, vol. 2006, 2006, pp. 403–410. DOI: 10.1109/AMC.2006.1631693. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33751042919&doi=10.1109%2fAMC.2006.1631693&partnerID=40&md5=3a83e17845d50781467546bb4af2a0f8.

[41] D. Gouaillier, V. Hugel, P. Blazevic, *et al.*, «Mechatronic design of nao humanoid», Cited by: 487, 2009, pp. 769–774. DOI: 10.1109/ROBOT.2009.5152516. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099230379&doi=10.1109%2fROBOT.2009.5152516&partnerID=40&md5=1b00e361c8d524475b5d2b50bedacc26.

[42] I. Ha, Y. Tamura, and H. Asama, «Development of open platform humanoid robot darwin-op», *Advanced Robotics*, vol. 27, no. 3, pp. 223–232, 2013, Cited by: 27. DOI: 10.1080/01691864.2012.754079. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84876409122&doi=10.1080%2f01691864.2012.754079&partnerID=40&md5=8fe43774d724fa77de325d8fd6f13cca.

[43] K. Imanishi and T. Sugihara, «Autonomous biped stepping control based on the lipm potential», Cited by: 12, vol. 2018-November, 2018, pp. 593–598. DOI: 10.1109/HUMANOIDS.2018.8625011. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85062298452&doi=10.1109%2fHUMANOIDS.2018.8625011&partnerID=40&md5=ea277cfab5b16c1c9c75c1d457556c2d.

[44] R. Zhang, M. Zhao, and C.-L. Wang, «Standing push recovery based on lipm dynamics control for biped humanoid robot», Cited by: 9, 2018, pp. 1732–1737. DOI: 10.1109/ROBIO.2018.8664792. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85064129721&doi=10.1109%2fROBIO.2018.8664792&partnerID=40&md5=cf4f74f83b5ec4195f5640a68e55dec4.

[45] D. Wang, W. Cao, B. Zhang, and M. Mukai, «Motion planning for a robotic wheelchair with slerp mpc local planner», Cited by: 1, 2023, pp. 925–930. DOI: 10.23919/SICE59929.2023.10354119. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85182606586&doi=10.23919%2fSICE59929.2023.10354119&partnerID=40&md5=434abd58e944a305fd56b9f6de233c4e.

[46] W. Cai, Q. Li, S. Huang, H. Zhu, Y. Yang, and M. Zhao, «Squat motion of a bipedal robot using real-time kinematic prediction and whole-body control», *IET Cyber-systems and Robotics*, vol. 4, no. 4, pp. 298–312, 2022, Cited by: 2; All Open Access, Gold Open Access. DOI: `10.1049/csy2.12073`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85144301843&doi=10.1049%2fcsy2.12073&partnerID=40&md5=28663866c7fb113ab7ca680b460c903a`.

[47] J. Bai, S. Zhong, J. Gao, Y. Zhang, and Z. Chen, «Dynamic locomotion of bipedal robots on steep slope through wbc and mpc», Cited by: 1, 2024, pp. 1049–1054. DOI: `10.1109/ICARM62033.2024.10715800`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85208054602&doi=10.1109%2fICARM62033.2024.10715800&partnerID=40&md5=9e1e0d7ca0e7d19307e097f32ef57b7d`.

[48] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, «Safe and fast tracking on a robot manipulator: Robust mpc and neural network control», *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020, Cited by: 194; All Open Access, Green Open Access. DOI: `10.1109/LRA.2020.2975727`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85081544523&doi=10.1109%2fLRA.2020.2975727&partnerID=40&md5=c0641f89de7a8e857009c253a3882a36`.

[49] P. Fan, J. Peng, S. Ding, *et al.*, «Trajectory-attracted adaptive tracking control for robotic systems based on a hybrid guiding vector field in flexible environments», *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 14 808–14 817, 2025, Cited by: 0. DOI: `10.1109/TASE.2025.3564644`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-105004075930&doi=10.1109%2fTASE.2025.3564644&partnerID=40&md5=c1133c35e09c09fef2e164e4e2cbdc72`.

[50] T. Rybus, «The obstacle vector field (ovf) method for collision-free trajectory planning of free-floating space manipulator», *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 70, no. 2, 2022, Cited by: 5; All Open Access, Gold Open Access. DOI: `10.24425/bpasts.2022.140691`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85129559633&doi=10.24425%2fbpasts.2022.140691&partnerID=40&md5=c2c09782949b6b36d4834035652b0675`.

[51] T. Nierhoff, S. Hirche, and Y. Nakamura, «Laplacian trajectory vector fields for robotic movement imitation and adaption», Cited by: 0, vol. 22, 2014, pp. 205–213. DOI: `10.1007/978-3-319-07058-2_24`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-84927667209&doi=10.1007%2f978-3-319-07058-2_24&partnerID=40&md5=60a193628e4be3d2d1f39844510fda6d`.

[52] C.-F. Juang, C.-H. Lu, and C.-A. Huang, «Navigation of three cooperative object-transportation robots using a multistage evolutionary fuzzy control approach», *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3606–3619, 2022. DOI: `10.1109/TCYB.2020.3015960`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85130768160&doi=10.1109%2fTCYB.2020.3015960&partnerID=40&md5=5505b9b165f4b3fa2e7cb29d90dc2ba8`.

[53] M. Wang and M. Schwager, «Distributed collision avoidance of multiple robots with probabilistic buffered voronoi cells», Cited by: 37, 2019, pp. 169–175. DOI: `10.1109/MRS.2019.8901101`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85075636690&doi=10.1109%2fMRS.2019.8901101&partnerID=40&md5=80dfda14db71959e88303b726d1b3449`.

[54] D. K. M. Zade and S. H. Semnani, «Temporary goal method: A solution for the problem of getting stuck in motion planning algorithms», *Iranian Conference on Electrical Engineering, ICEE*, no. 2024, 2024, Cited by: 0. DOI: `10.1109/ICEE63041.2024.10668270`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85208637011&doi=10.1109%2fICEE63041.2024.10668270&partnerID=40&md5=fbb66726af347fc7f5cb5ba996819b27`.

[55] W. Chen and Y. Zhu, «Hierarchical multi-robot pursuit with deep reinforcement learning and navigation planning», Cited by: 1, 2024, pp. 1267–1273. DOI: `10.1109/YAC63405.2024.10598424`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85200785934&doi=10.1109%2fYAC63405.2024.10598424&partnerID=40&md5=f8860ee84b097ef18c8ec23589ab4245`.

[56] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, «Chance constraint based multi agent navigation under uncertainty», Cited by: 3; All Open Access, Green Open Access, vol. Part F132085, 2017. DOI: `10.1145/3132446.3134917`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85038433627&doi=10.1145%2f3132446.3134917&partnerID=40&md5=fcbbe532a3dad43f78fa18d93aab338e`.

[57] E. Hourdakis, G. Chliveros, and P. Trahanias, «Orca: A physics based, robotics simulator able to distribute processing across several peers», Cited by: 1, 2013. DOI: 10.1109/ISR.2013.6695664. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84893283642&doi=10.1109%2fISR.2013.6695664&partnerID=40&md5=05d76597161e336aef905014f4ec8e63.

[58] M. R. Mohd Romlay, A. Mohd Ibrahim, S. F. Toha, P. De Wilde, I. Venkat, and M. S. Ahmad, «Obstacle avoidance for a robotic navigation aid using fuzzy logic controller-optimal reciprocal collision avoidance (flc-orca)», *Neural Computing and Applications*, vol. 35, no. 30, pp. 22 405–22 429, 2023, Cited by: 14. DOI: 10.1007/s00521-023-08856-8. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85167521134&doi=10.1007%2fs00521-023-08856-8&partnerID=40&md5=3ea313711db6c565f626414574d4dd4d.

[59] L. Hawley and W. Suleiman, «Control framework for cooperative object transportation by two humanoid robots», *Robotics and Autonomous Systems*, vol. 115, pp. 1–16, 2019, Cited by: 29; All Open Access, Green Open Access. DOI: 10.1016/j.robot.2019.02.003. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85061650648&doi=10.1016%2fj.robot.2019.02.003&partnerID=40&md5=6cfd7c04ae9518c4ec898ba066ba725f.

[60] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, «Robotics: Modelling, planning and control», *Advanced Textbooks in Control and Signal Processing*, vol. 0, pp. 1–623, 2009, Cited by: 1823. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84969566030&partnerID=40&md5=2814a554c3051b04e6991a2992de662a.

[61] Z. Hu, Z. Zhao, L. Zhang, *et al.*, «Collaborative object transportation by multiple robots with onboard object localization algorithm», 2019, pp. 2344–2350. DOI: 10.1109/ROBIO49542.2019.8961823. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85079062678&doi=10.1109%2fROBIO49542.2019.8961823&partnerID=40&md5=45a1c407e89e317958ae7c99271f8914.

[62] G. Eoh, J. D. Jeon, J. S. Choi, and B. H. Lee, «Multi-robot cooperative formation for overweight object transportation», 2011, pp. 726–731. DOI: 10.1109/SII.2011.6147538. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84863141808&doi=10.1109%2fSII.2011.6147538&partnerID=40&md5=35ced714b053945e410bd4d621844fd4.

[63] M.-H. Wu, A. Konno, S. Ogawa, and S. Komizunai, «Symmetry cooperative object transportation by multiple humanoid robots», 2014, pp. 3446–3451. DOI: 10.1109/ICRA.2014.6907355. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84929170559&doi=10.1109%2fICRA.2014.6907355&partnerID=40&md5=928f0228e496be2a4152c86f3d4967b7.

[64] M.-H. Wu, S. Ogawa, and A. Konno, «Symmetry position/force hybrid control for cooperative object transportation using multiple humanoid robots», *Advanced Robotics*, vol. 30, no. 2, pp. 131–149, 2016. DOI: 10.1080/01691864.2015.1096212. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84955200696&doi=10.1080%2f01691864.2015.1096212&partnerID=40&md5=fedf72c9d790d5faeb3ac1f80f6b66df.

[65] M.-H. Wu, A. Konno, and M. Uchiyama, «Cooperative object transportation by multiple humanoid robots», Cited by: 10, 2011, pp. 779–784. DOI: 10.1109/SII.2011.6147547. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84857567298&doi=10.1109%2fSII.2011.6147547&partnerID=40&md5=b8608b9a3a5fdb7b59e9d3a697df3512.

[66] A. Florin and A. Silvia, «Synchronizing robot motions in cooperative tasks», *Control Engineering and Applied Informatics*, vol. 13, no. 1, pp. 43–48, 2011, Cited by: 5. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-79957935308&partnerID=40&md5=5234912fa22b0b86cb0afb3b670e6fcb.

[67] S. Saeedvand, H. Mandala, and J. Baltes, «Hierarchical deep reinforcement learning to drag heavy objects by adult-sized humanoid robot», *Applied Soft Computing*, vol. 110, 2021, Cited by: 22. DOI: 10.1016/j.asoc.2021.107601. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85107951694&doi=10.1016%2fj.asoc.2021.107601&partnerID=40&md5=9b3360ca5aa39923ec3143810b049799.

[68] H. Jeon, D.-W. Kim, and B.-Y. Kang, «Deep reinforcement learning for cooperative robots based on adaptive sentiment feedback», *Expert Systems with Applications*, vol. 243, 2024, Cited by: 10. DOI: 10.1016/j.eswa.2023.121198. [Online]. Available: https://www.scopus.com/inward/record.

uri?eid=2-s2.0-85179581967&doi=10.1016%2fj.eswa.2023.121198&partnerID=40&md5=6348a234da140e88953e3c43fed25614.

[69] P.-H. Kuo, W.-C. Yang, P.-W. Hsu, and K.-L. Chen, «Intelligent proximal-policy-optimization-based decision-making system for humanoid robots», *Advanced Engineering Informatics*, vol. 56, 2023, Cited by: 12. DOI: 10.1016/j.aei.2023.102009. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85159572331&doi=10.1016%2fj.aei.2023.102009&partnerID=40&md5=7bb428dce81a393b721e0522f052e8f8.

[70] Y. Inoue, T. Tohge, and H. Iba, «Cooperative transportation system for humanoid robots using simulation-based learning», *Applied Soft Computing Journal*, vol. 7, no. 1, pp. 115–125, 2007, Cited by: 17. DOI: 10.1016/j.asoc.2005.05.001. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-33750936227&doi=10.1016%2fj.asoc.2005.05.001&partnerID=40&md5=f993408e33e2e58cf689191650e9f82d.

[71] H. Zheng, J. Jiang, P. Wei, G. Long, and C. Zhang, «Competitive and cooperative heterogeneous deep reinforcement learning», Cited by: 10, vol. 2020-May, 2020, pp. 1656–1664. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85096640325&partnerID=40&md5=9d8ffddf37d81574a5b5b9678476e5d9.

[72] J. Baltes, I. Akbar, and S. Saeedvand, «Cooperative dual-actor proximal policy optimization algorithm for multi-robot complex control task», *Advanced Engineering Informatics*, vol. 63, 2025, Cited by: 3. DOI: 10.1016/j.aei.2024.102960. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85210773405&doi=10.1016%2fj.aei.2024.102960&partnerID=40&md5=8fe1afbc813fd39ce6ade8a2098cb1a4.

[73] J. K. Gupta, M. Egorov, and M. Kochenderfer, «Cooperative multi-agent control using deep reinforcement learning», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10642 LNAI, pp. 66–83, 2017, Cited by: 803. DOI: 10.1007/978-3-319-71682-4_5. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85036632670&doi=10.1007%2f978-3-319-71682-4_5&partnerID=40&md5=dea7e5bf59b66560ef3feba655f3254f.

[74] F. Xie, Z. Guo, T. Li, Q. Feng, and C. Zhao, «Dynamic task planning for multi-arm harvesting robots under multiple constraints using deep reinforcement learning», *Horticulturae*, vol. 11, no. 1, 2025, Cited by: 22; All Open Access, Gold Open Access. DOI: 10.3390/horticulturae11010088. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85216017146&doi=10.3390%2fhorticulturae11010088&partnerID=40&md5=06868122808965b5c8043a2ff3f3130d.

[75] L. Deng, W. Gong, and L. Li, «Multi-robot exploration in unknown environments via multi-agent deep reinforcement learning», Cited by: 8, vol. 2022-January, 2022, pp. 6898–6902. DOI: 10.1109/CAC57257.2022.10055585. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85151126373&doi=10.1109%2fCAC57257.2022.10055585&partnerID=40&md5=91e7a407d2bf7f9893ecb9da62a40e63.

[76] H.-C. Chen, S.-A. Li, T.-H. Chang, H.-M. Feng, and Y.-C. Chen, «Hybrid centralized training and decentralized execution reinforcement learning in multi-agent path-finding simulations», *Applied Sciences (Switzerland)*, vol. 14, no. 10, 2024, Cited by: 3; All Open Access, Gold Open Access. DOI: 10.3390/app14103960. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85194377474&doi=10.3390%2fapp14103960&partnerID=40&md5=1a70c4638a345d20389f60bbfc27f10a.

[77] H. Yun, «Multi-agents on a muti-layer network a maddpg-based method», Cited by: 0, 2024, pp. 1456–1459. DOI: 10.1109/CISCE62493.2024.10653107. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85204706020&doi=10.1109%2fCISCE62493.2024.10653107&partnerID=40&md5=a60676782d1a60a438ed4fefb02afce4.

[78] L. Sun, H. Chen, Z. Guo, T. Wang, A. Ding, and H. Ma, «Oma-qmix: Exploring optimal multi-agent reinforcement learning framework in multi-action spaces», Cited by: 0, 2024, pp. 8194–8199. DOI: 10.23919/CCC63176.2024.10662294. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85205497096&doi=10.23919%2fCCC63176.2024.10662294&partnerID=40&md5=9443126c4a223438160f29cbc9ae3301.

[79] L. Zhang, Y. Sun, A. Barth, and O. Ma, «Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning», *IEEE Access*, vol. 8, pp. 184 109–184 119, 2020, Cited by: 58; All Open Access, Gold Open Access. DOI: 10.1109/ACCESS.2020.3025287. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85102809910&doi=10.1109%2fACCESS.2020.3025287&partnerID=40&md5=2b15648f68179d96e955961daf9cba88.

APPENDIX A

Additional content