



UNIVERSITY OF TRENTO

DEPARTMENT OF INDUSTRIAL ENGINEERING

MASTERS'S DEGREE IN MECHATRONICS ENGINEERING

~ · ~

ACADEMIC YEAR 2023–2024

# Leveraging Acquired Knowledge and Invariant Representations for Cable Manipulation

**Supervisor**

Prof. Matteo SAVERIANO

Prof. Dongheui LEE

**Graduate Student**

Erik MISCHIATTI

233242

FINAL EXAMINATION DATE: November 28, 2024

---

---

---

---

*The limit? Who decided that? If you have the time to fail ... you have to keep trying!*  
*Saitama*

---

## Abstract

Robotic manipulation has become a cornerstone of modern automation, with applications spanning manufacturing, logistics, and healthcare. The ability to handle complex tasks, such as assembling or manipulating objects with diverse physical properties, is a critical requirement for next-generation robotic systems. These tasks often involve rigid and deformable objects, demanding a combination of precision, adaptability, and robustness from the underlying framework.

This thesis addresses these challenges by presenting a robotic manipulation system designed to autonomously perform assembly and disassembly tasks. The project builds on modular assembly principles inspired by NIST standards and introduces an innovative trajectory learning approach based on kinesthetic demonstration and invariant representations. These methods ensure motion planning that is both robust and generalizable across varying scenarios.

The proposed system integrates cutting-edge hardware and software components, including a robotic arm, ROS-based communication for seamless integration, and Behavior Trees (BTs) for adaptive and modular task execution. Demonstration trajectories are captured, pre-processed, and reconstructed within an invariant space to maintain precision and flexibility.

Experimental results highlight the framework's ability to perform complex manipulation tasks with high accuracy, even under dynamically changing conditions. By providing a robust foundation for trajectory generalization and scalable system architectures, this work contributes to the advancement of robotic autonomy and underscores the growing potential of flexible automation solutions within the paradigm of Industry 4.0.

---

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 State of the Art</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.2 Perception . . . . .	3
1.2.1 Definition and Importance . . . . .	3
1.2.2 Technologies and Methods . . . . .	4
1.2.3 Critique and Gaps . . . . .	4
1.3 Planning . . . . .	5
1.3.1 Definition and Importance . . . . .	5
1.3.2 Technologies and Methods . . . . .	5
1.3.3 Critique and Gaps . . . . .	6
1.4 Execution . . . . .	6
1.4.1 Definition and Importance . . . . .	6
1.4.2 Technologies and Methods . . . . .	6
1.4.3 Critique and Gaps . . . . .	7
<b>2 Theoretical Foundations</b>	<b>9</b>
2.1 Introduction . . . . .	9
2.2 Perception . . . . .	9
2.2.1 Learning from Demonstration . . . . .	10
2.2.2 Feature Extraction . . . . .	13
2.2.3 Color and Shape Detection for Cable Connector Identification . . . . .	15
2.3 Planning . . . . .	17
2.3.1 Dynamic Movement Primitives (DMP) . . . . .	18
2.3.2 Bidirectional Invariant Representation . . . . .	22
2.3.3 Trajectory and Orientation Generalization . . . . .	25
2.4 Execution . . . . .	26
2.4.1 Behavior Trees . . . . .	26
<b>3 Materials and Equipment</b>	<b>29</b>
3.1 Introduction . . . . .	29
3.2 Custom Assembly Board for Robotic Manipulation . . . . .	29
3.2.1 Inventory of Experimental Equipment and Materials . . . . .	30
3.3 Hardware Equipment . . . . .	32
3.3.1 Robotic Arm (Franka Emika Panda) . . . . .	32
3.3.2 Camera (Intel RealSense D435i) . . . . .	32
3.4 Software Tools . . . . .	33



## CONTENTS

---

3.4.1	ROS (Robot Operating System) . . . . .	33
3.4.2	Aruco Libraries . . . . .	33
3.4.3	Other Software Tools . . . . .	33
<b>4</b>	<b>System Design and Development</b>	<b>35</b>
4.1	Introduction . . . . .	35
4.2	System Requirements and Design Goals . . . . .	35
4.2.1	Functional Requirements . . . . .	35
4.2.2	Design Goals . . . . .	36
4.2.3	Constraints . . . . .	37
4.3	Integration of Hardware Components . . . . .	38
4.3.1	Integration of the Robotic Arm and Sensors . . . . .	38
4.3.2	RealSense and Aruco Synchronization . . . . .	38
4.3.3	Custom Assembly Board and Object Manipulation . . . . .	39
4.3.4	Challenges in Hardware Integration . . . . .	39
4.4	Software Architecture and Communication . . . . .	40
4.4.1	ROS Node Management . . . . .	40
<b>5</b>	<b>Experimental Method</b>	<b>43</b>
5.1	Introduction . . . . .	43
5.2	Experimental Setup . . . . .	43
5.2.1	Configuration of the Environment . . . . .	43
5.2.2	Test Scenarios . . . . .	44
5.3	Experimental Procedure . . . . .	46
5.3.1	Preparation Phase . . . . .	46
5.3.2	Trajectory Recording and Processing Phase . . . . .	47
5.3.3	Execution Phase . . . . .	48
5.3.4	Post-Experiment Analysis . . . . .	49
5.4	Results and Observations . . . . .	49
5.4.1	Validation of Individual Modules . . . . .	49
5.4.2	Framework Performance in Task Execution . . . . .	52
	<b>Conclusions</b>	<b>55</b>
	<b>A 3D-Printed Tools for the Custom Assembly Board</b>	<b>57</b>
	<b>B ROS Node Graph</b>	<b>59</b>
	<b>Bibliography</b>	<b>63</b>
	<b>List of Figures</b>	<b>65</b>
	<b>List of Tables</b>	<b>67</b>

# Introduction

Modern industry increasingly relies on advanced robotics to optimize production processes, reduce costs, and improve safety and efficiency. The shift towards digitalization and automation has propelled innovation to new levels of human-machine interaction, where direct collaboration between human operators and robotic systems is crucial for achieving complex objectives. Current examples of this interaction include collaborative production in automotive factories, where human operators and robots work together to assemble complex components, and in the food industry, where robotic systems assist in selecting and packaging products while maintaining high hygiene standards.

This convergence of advanced technology and human labor also extends to other sectors such as logistics and hospitality, where autonomous mobile robots (AMRs) enhance efficiency and reduce operational costs by automating processes like food and beverage transportation or material handling without direct contact. These innovations not only optimize current operations but also pave the way for new applications of collaborative robotics (cobots) in processes requiring delicate and adaptable interactions, such as material assembly and complex component handling.

This evolution highlights a growing trend towards the use of robotic systems in configurations that require greater flexibility and adaptability. Consequently, recent developments in the manipulation of Deformable Linear Objects (DLOs) further illustrate how automation can be significantly enhanced through the integration of sensory feedback and predictive models. This allows robotic systems to dynamically adapt to operational complexities in industrial scenarios such as the production of switchgear and wiring, demonstrating a transformative impact on the effectiveness of robotic operations [5].

In this evolving context, the robotic manipulation of DLOs such as cables, ropes, and tubes presents significant challenges due to their inherently unpredictable nature and complex non-linear dynamics. Recent innovations in differentiable dynamic models for shape control of these objects have opened new frontiers in robotic trajectory programming, allowing for precise manipulations with minimal human intervention. Additionally, recent research has developed advanced models for estimating and tracking the 3D shape of DLOs from multiple 2D views, significantly improving robots' manipulation capabilities in industrial environments [6] [18] [10] [12] [37] [23].

Assembly and disassembly scenarios, where DLOs are employed, require precision in inserting flexible cables into tight spaces and careful management to avoid damage. These processes, which represent critical areas in industrial automation, benefit significantly from advanced detection and physical interaction capabilities, as demonstrated by technologies that allow robots to perceive and manipulate with increased precision [19], furthermore, innovations such as the real-time estimation of model parameters for DLO manipulation highlight how automation can dynamically adapt to operational needs, enabling more effective and efficient resource management in production contexts [8] [16].

The problem of production and assembly of wiring, still largely dependent on human labor, highlights the need for innovative robotic solutions capable of manipulating deformable objects

with precision, especially in environments characterized by tight spaces and numerous obstacles [4]. The research by Bostelman and Falco provides a detailed analysis of industrial manipulation technologies that facilitate autonomous assembly tasks, underscoring the importance of these developments for modern industry [5].

Incorporating these developments, research continues to push the boundaries of robotic technology, making machines increasingly capable of tackling tasks that were once considered too difficult or impossible for automation. For example, the semi-automatic approach to image marking for DLOs proposed by Caporali et al. shows how artificial intelligence can reduce human workload and improve the accuracy of robotic operations [7]. These advancements not only enhance the effectiveness of DLO manipulation but also demonstrate how automation can be significantly boosted through the integration of sensory feedback and predictive models, allowing robotic systems to dynamically adapt to operational complexities in industrial scenarios such as switchgear and wiring production.

This thesis aims to address these challenges by developing a robotic manipulation system capable of autonomously handling rigid and deformable objects. Leveraging advanced techniques such as kinesthetic demonstration, invariant representations, and Behavior Trees, this work contributes to the growing field of adaptive robotics, with a particular focus on industrial applications requiring precision and flexibility.

# Chapter 1

## State of the Art

### 1.1 Introduction

The "State of the Art" chapter aims to provide an overview of the current knowledge on the research topic, analyzing contributions from the literature and identifying both existing solutions and gaps that need to be addressed. The analysis focuses on similar or related works, with the goal of highlighting approaches, methods, and technologies relevant to the context of the thesis.

The chapter is structured into three main areas:

- **Perception**, which analyzes technologies and methods for environmental perception, including sensors and algorithms for object reconstruction and recognition.
- **Planning**, which focuses on techniques for task planning and optimization, particularly for robotic manipulation in dynamic environments.
- **Execution**, which focuses on methods for executing robotic tasks, from motion planning to control.

This structure, which will be revisited later in Chapter 2, allows for the identification of not only established knowledge but also open research areas, providing the necessary context to justify and position the contributions of the thesis.

### 1.2 Perception

#### 1.2.1 Definition and Importance

*"Perception is analyzing the sensing data that the robot gets, and outputting useful information for downstream tasks."*

This topic represents a fundamental component in robotic systems and the first step in our project, enabling them to acquire and interpret information about their surroundings. Through data obtained from vision devices or technologies that mimic other types of "senses," robots can identify objects, analyze their physical properties, and estimate their position and orientation. This process is particularly critical for manipulating complex objects, such as Deformable Linear

Objects. Manipulating these objects presents unique challenges, as their deformable nature and interactions with the environment require precise and robust perception to ensure operational success.

### 1.2.2 Technologies and Methods

Regarding deformable objects, perception technologies primarily rely on visual sensors such as RGB, RGB-D, and stereo cameras, often combined with advanced algorithms for segmentation, shape estimation, and tracking. RGB-D cameras, for instance, provide a three-dimensional representation of the scene, essential for geometric reconstruction tasks [6]. However, more economical and flexible configurations, such as 2D cameras in an *eye-in-hand* setup, are often preferred in industrial scenarios due to their simplicity of implementation and lower cost [16]. Additionally, stereo cameras and LIDAR are applied in situations requiring more accurate reconstruction of the shape and position of complex objects [23].

In the context of this project, a combined approach leveraging color and shape detection was implemented to identify the connector of a cable and estimate its position and orientation in 3D space. Using an RGB-D camera, the system segments the object based on color thresholds in the HSV color space and combines this information with depth data to calculate 3D coordinates of key points, that in our case is on the connector of the cable. This method ensures accurate detection even in cluttered environments, where traditional algorithms may struggle with occlusions or overlapping features. Additionally, advanced processing techniques, including exponential smoothing, are applied to stabilize the detected coordinates and mitigate noise in the depth data. The detected 3D axis of the connector is then visualized in real-time and used as input for downstream manipulation tasks. This approach highlights the effectiveness of combining geometric and color-based techniques to enhance perception capabilities in robotic systems.

From an algorithmic perspective, innovative approaches such as RT-DLO propose using graph-based representations to model DLOs [9]. These algorithms combine binary masks, generated through convolutional neural networks, with graph analysis that segments and represents DLOs in terms of nodes and edges. This allows for successfully managing complex intersections and generating skeletal representations ready for manipulation. Similarly, algorithms like mBEST utilize energy-minimization techniques to produce topologically accurate representations of DLOs, improving segmentation robustness in challenging environments [12][19].

For applications requiring dynamic manipulations, advanced models like the *Discrete Elastic Rod Model* (DER) incorporate deformations such as curvature, torsion, and stretching to accurately describe the physical behavior of DLOs [23]. Other approaches, such as the RT-Cable framework, adopt spatial representations that reduce dependency on absolute positions, focusing instead on the relative relationships between the cable and surrounding objects [18]. This strategy significantly simplifies the planning and execution of complex tasks like cable routing.

A key aspect of successful perception algorithms is the availability of high-quality datasets. Using synthetic data, generated through photorealistic simulations, has proven effective for training deep learning models, significantly reducing the costs of manual labeling [8]. However, to improve generalization to real-world scenarios, many approaches combine synthetic data with online adaptation techniques, updating the models during manipulation operations [6][38].

### 1.2.3 Critique and Gaps

Despite significant progress, several open challenges remain in the field of DLO perception. Generalization remains one of the main issues: models trained on synthetic data often struggle in

real-world scenarios, particularly when dealing with objects with physical properties that differ significantly from those simulated [8][38]. Additionally, robustness in dynamic and complex environments represents a barrier to large-scale industrial adoption. While algorithms such as RT-DLO and mBEST have improved the handling of intersections and topologically complex configurations [9][12], their performance may degrade in the presence of significant occlusions or rapid variations in environmental conditions.

Another limitation concerns the integration of additional sensors, such as tactile ones, which could provide complementary information to improve manipulation accuracy. Currently, most approaches rely exclusively on visual data, overlooking the potential of multi-sensory feedback. Finally, computational scalability remains a challenge for real-time applications involving many DLOs or particularly complex scenes.

## 1.3 Planning

### 1.3.1 Definition and Importance

The ability to plan and adapt is central to the functionality of robotic systems, enabling the reconstruction and execution of trajectories while ensuring precision and flexibility. This process involves leveraging techniques such as trajectory generalization, as well as advanced algorithms like Dynamic Movement Primitives (DMP) and the DHB-based invariant representation. These methodologies allow for the encoding and adaptation of motions to new contexts while preserving the essential characteristics of the original trajectories. In this thesis, planning is not just about achieving motion efficiency but also about ensuring robust and reliable task execution through the use of invariant-based frameworks.

### 1.3.2 Technologies and Methods

Invariant trajectory representations are becoming essential tools in robotic planning, especially for generalizing and recognizing demonstrated movements. Among the latest approaches, several innovative methodologies stand out:

**Dynamic Movement Primitives (DMPs).** DMPs represent one of the foundational frameworks for trajectory planning. They allow the encoding of complex movements as nonlinear dynamical systems, ensuring robustness and adaptability. Their key features include scalability, temporal invariance, and the ability to learn from demonstrations [1]. Recent advancements, such as the introduction of reversibility, have expanded their applicability, enabling bidirectional trajectory execution for tasks like assembly and disassembly [28]. Furthermore, the integration of coupling terms enables real-time adjustments to avoid obstacles or modify trajectories dynamically [22].

**DHB Model (Denavit-Hartenberg Bi-directional).** A Denavit-Hartenberg-inspired approach proposes a bidirectional representation to describe motion trajectories compactly and invariantly [20] [21]. This method decomposes trajectories into invariant components, enabling both recognition and accurate reproduction of movements. Key properties include:

- **Modularity and Robustness:** DHB representations separate position and orientation, simplifying integration into articulated systems.
- **Adaptability:** The ability to handle transformations such as scaling, rotation, and translation allows for greater generalization in dynamic scenarios.

This framework has proven particularly useful for complex manipulation tasks, such as recognizing and reproducing human motions in human-robot interaction contexts.

**Invariant Trajectory Representations.** Methods based on invariant representations, such as those introduced in [31], leverage spatio-temporal curvature to segment and analyze trajectories. These approaches offer robustness to viewpoint changes and transformations, making them ideal for recognizing and reproducing actions. In demonstration and knowledge transfer contexts, such representations simplify the process of mapping human movements to robotic systems, ensuring precision and adaptability.

**Trajectory Optimization.** A robust optimization-based approach has been developed to compute invariant trajectory representations, reducing noise and managing singularities. This method uses optimal control constraints to generate reliable trajectories even in noisy or dynamic scenarios [35]. Practical applications include adaptive motion planning and human intent recognition, where invariant descriptors facilitate transferring demonstrated tasks to new environments with minimal recalibration.

**Reversible Task Execution.** An additional advancement is the integration of logistic differential equations into DMPs to enable reversible trajectory execution [22]. This approach has proven particularly effective in assembly tasks like "peg-in-hole," where reversibility facilitates error recovery by allowing the robot to return to previous states [11].

### 1.3.3 Critique and Gaps

Despite significant progress, some open challenges remain in robotic planning:

- **Computational Complexity:** Processing invariant representations or optimizing trajectories in real-time can be demanding in highly dynamic environments.
- **Reliability in Noisy Environments:** Dependence on precise sensory data may introduce errors in unstructured scenarios or when sensors are affected by noise.
- **Applicability to Deformable Objects:** While many methods are optimized for rigid objects, there are limitations in generalizing to deformable objects or multi-contact manipulation.

## 1.4 Execution

### 1.4.1 Definition and Importance

Execution in robotic systems refers to the real-time implementation of planned actions, including trajectory control, motion coordination, and interaction with the environment. It represents the final stage of robotic operations, where abstract plans are transformed into precise physical actions. Execution is critical for ensuring task success, particularly in dynamic environments where external disturbances and uncertainties must be addressed.

### 1.4.2 Technologies and Methods

The effective execution of robotic tasks relies on a combination of advanced control strategies, sensor integration, and high-level decision-making frameworks. Several methodological approaches have been explored to meet these requirements:

**Behavior Trees (BTs).** Behavior Trees (BTs) are hierarchical frameworks that enable modular, scalable, and reusable execution of robotic tasks. BTs combine decision-making and action execution in a tree structure, where each node represents a behavior or a condition [13] [24]. This topic will be discussed in greater detail in the following chapters, as it is the approach adopted for the project.

**Force and Impedance Control.** Force-based methods, including impedance and admittance control, are fundamental for ensuring safe and effective interaction between robots and their environment. Variable impedance control, for instance, allows robots to dynamically adjust their stiffness or compliance depending on the task requirements [3]. These methods are particularly useful for tasks involving physical interactions, such as assembly or manipulation of deformable objects.

**Sensor Integration and Feedback.** Robotic execution increasingly relies on multi-sensory feedback to improve precision and robustness. For instance:

- **Tactile Sensing:** Provides detailed information about contact forces, enabling delicate object manipulation [15].
- **Visual Servoing:** Guides robotic actions using real-time visual data, ensuring accuracy in tasks such as pick-and-place operations [17].

The combination of tactile and visual feedback is particularly important for managing uncertainties in dynamic environments.

**Model Predictive Control (MPC).** Model Predictive Control (MPC) has emerged as a powerful tool for robotic execution, offering real-time optimization of control inputs based on future state predictions. This approach is particularly suited for dynamic tasks that require simultaneous consideration of constraints, such as collision avoidance and energy efficiency [26].

**Optimized Trajectories.** Optimized trajectories represent a key aspect of execution, especially when calculated in real-time during a task. Optimization techniques, such as those applied in Model Predictive Control, allow continuous adjustment of trajectories to adapt to dynamic conditions and physical constraints [35]. This approach integrates with advanced control to ensure precision and efficiency.

### 1.4.3 Critique and Gaps

Despite significant progress, several challenges remain in the execution of robotic tasks:

- **Scalability in Complex Environments:** Real-time execution frameworks, such as BTs and MPC, can become computationally intensive in scenarios with high-dimensional state spaces.
- **Robustness to Noise:** Sensory noise and environmental uncertainties can degrade the performance of control algorithms, particularly in unstructured settings.
- **Integration of Advanced Sensors:** While multi-sensory feedback enhances precision, it also increases system complexity and requires advanced data fusion techniques.





## Chapter 2

# Theoretical Foundations

### 2.1 Introduction

Robots are becoming increasingly essential in a variety of fields, from industrial automation to service applications. To operate effectively in dynamic and unstructured environments, they must be capable of perceiving their surroundings, planning appropriate actions, and executing these actions reliably. Achieving this level of autonomy requires a systematic integration of perception, planning, and execution, with each component playing a crucial role in enabling robots to perform complex tasks.

This chapter provides a comprehensive exploration of these three pillars of robotics. The Perception section focuses on enabling robots to observe and interpret their environment, particularly through Learning from Demonstration (LfD). By learning directly from human demonstrations, robots can acquire new skills and behaviors without the need for extensive manual programming. This capability is critical for adapting to diverse and evolving tasks.

The Planning section introduces techniques for transforming the data gathered through perception into actionable plans. It highlights the use of Dynamic Movement Primitives (DMPs) for encoding and generalizing trajectories, and the Denavit-Hartenberg inspired Bidirectional Representation (DHB) for robust motion description and reconstruction. Together, these methods ensure that robots can plan actions that are both flexible and precise, even in dynamic environments.

Finally, the Execution section delves into the implementation of action plans using Behavior Trees (BTs). These provide a modular and hierarchical framework for real-time decision-making and action control, enabling robots to respond dynamically to changes in their environment while ensuring scalability and robustness.

By integrating these components, this chapter lays the foundation for designing robotic systems capable of seamless interaction with their environment and performing complex tasks with autonomy and reliability.

### 2.2 Perception

The primary goal of Perception is to enable robots to observe and interpret the environment and the actions demonstrated by a human operator. This chapter focuses on the paradigm of Learning from Demonstration (LfD), an approach that allows robots to acquire complex behaviors through the observation of human demonstrations, eliminating the need for explicit programming.

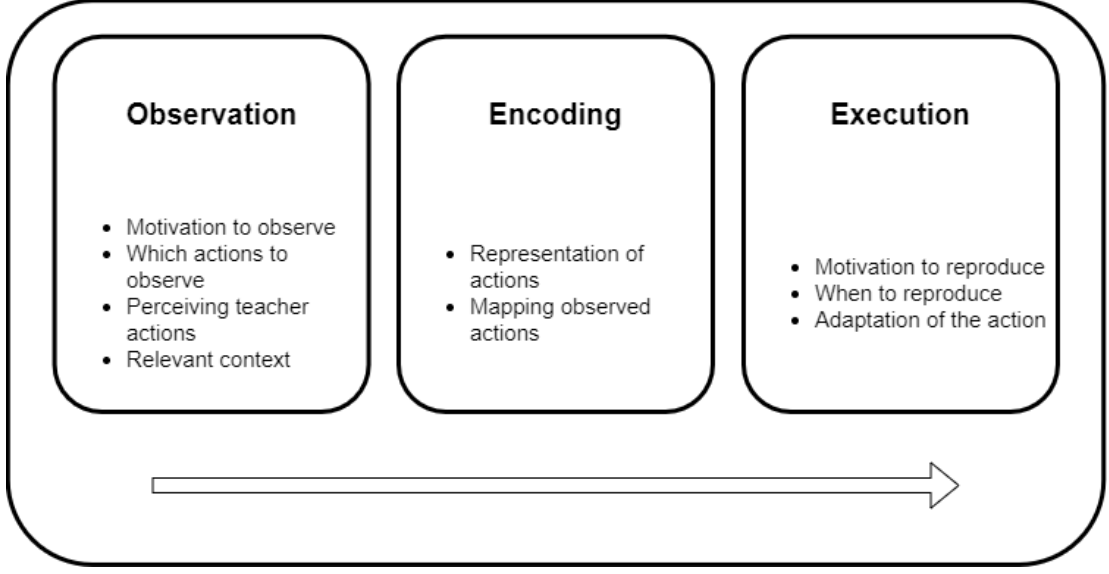


Figure 2.1:

Figure 2.2: The three main phases in imitation

Additionally, it explores the integration of sensory data with advanced processing techniques to facilitate object recognition and localization, essential for downstream manipulation tasks.

### 2.2.1 Learning from Demonstration

*“Imitation is defined as the process by which an agent learns a behavior by observing the execution of that behavior by a teacher.” [2]*

Learning by Imitation, also known as **Learning from Demonstration (LfD)**, involves recognizing demonstrated actions, encoding them in a meaningful format, and autonomously reproducing them. LfD is valuable as it allows robots to acquire complex behaviors without extensive manual programming, facilitating quicker adaptation to new tasks and environments.

In robotics, LfD enables robots to autonomously replicate human actions, effectively reducing the time and complexity required to teach robots, especially in industrial assembly, where flexibility and precision are critical. Through observing human demonstrations, robots can generalize the behaviors learned, applying them to various situations with a level of adaptability that traditional programming often lacks.

LfD has become a powerful tool for tasks such as robotic assembly, where robots have traditionally been programmed to follow predefined trajectories for specific tasks like peg insertion, bolt screwing, or component assembly. Rather than requiring rigid programming for each task, robots can now learn directly from demonstrations, adapting fluidly to changes in the environment or task requirements. By observing, encoding, and reproducing actions, robots can autonomously execute tasks, integrating learned behaviors with their physical capabilities and adapting them to current operating conditions.

Demonstration	Ease of demonstration	High DOFs	Ease of mapping
Kinesthetic Teaching	✓		✓
Teleoperation		✓	✓
Passive Observation	✓	✓	
Motion-Sensor Demonstration	✓	✓	

Table 2.1: Comparison of learning-from-demonstration methods

### Robotic Assembly

LfD allows robots to acquire complex behaviors through observation, removing the need for explicit programming. In robotic assembly, LfD has proven to be an efficient method for enabling robots to perform tasks such as peg-in-hole insertions, bolt screwing, and component assembly without extensive reprogramming. Traditionally, assembly line robots were programmed to follow predefined trajectories, but with LfD, they can learn these behaviors directly from demonstrations, providing greater adaptability and faster setup times [39].

### Key Concepts in Learning from Demonstration

The process of learning from demonstration involves three fundamental stages: **Observation**, **Encoding** or representation, and **Execution** or reproduction. During the observation phase, the robot records and analyzes the actions demonstrated by the teacher, including the task’s context and objectives. In the representation phase, these actions are converted into an internal model that the robot can use to autonomously execute the task. Finally, in the reproduction phase, the robot replicates the observed behavior, adapting it to its physical capabilities and the specific conditions of its environment [2].

### Demonstration Approach

There are several methods by which demonstrations can be provided to a robot. In many cases, the robot can be guided through a task physically or via teleoperation, where an operator remotely controls the robot while it learns the desired behaviors. Demonstrations can also be captured through motion sensors or motion-capture systems, which are then translated into robotic actions.

Table 2.1 provides a summary of the main similarities and distinctions between these approaches, focusing on the ease of demonstration, capacity to manage a high number of degrees of freedom, and the simplicity of mapping the demonstrations onto the robot’s configuration or operational space. In the following sections, each demonstration method is examined in detail. [39] [32]

**Kinesthetic Demonstration** Kinesthetic Demonstration is one of the most effective methods for teaching robots new tasks. In this Learning from Demonstration (LfD) approach, a human instructor physically guides the robot through the required motions. During this process, the robot’s actuators are placed in a passive mode, allowing the demonstrator to manipulate its joints directly. The robot records both the trajectory and the forces applied throughout the task, creating a set of training data that it can later use to autonomously reproduce the action. This approach is particularly beneficial for complex assembly tasks, such as peg-in-hole operations, where precision and adaptability are essential.

Kinesthetic teaching is widely used with robotic manipulators, including lightweight industrial robots, due to its intuitive nature and minimal training requirements for users. This method

relies solely on the robot’s onboard hardware and does not require external sensors, interfaces, or additional inputs. Recording demonstrations directly on the robot also eliminates the correspondence problem, simplifying the machine-learning process and enabling the robot to focus on accurate motion reproduction.

A key advantage of kinesthetic teaching is its hands-on approach, which allows the demonstrator to gain an intuitive sense of the robot’s physical limitations within its environment. This direct interaction with the robot is often more immersive than using simulated or virtual demonstrations. Furthermore, proprioceptive feedback from motor encoders allows the robot to “sense” its own motion by registering joint-angle data at each degree of freedom.

Despite its effectiveness, kinesthetic teaching has several limitations. The quality of the demonstration can depend on the dexterity and smoothness of the human user; even with skilled demonstrators, the recorded data often require post-processing, such as smoothing, to ensure consistency. Additionally, kinesthetic teaching is primarily applicable to robotic manipulators, as these robots have an intuitive form factor that facilitates physical guidance. On other platforms, such as legged robots or robotic hands, kinesthetic demonstration may be challenging or less effective.

In certain implementations, the demonstrator can control aspects of the task execution, such as starting and stopping the recording, through tools like buttons on the robot’s cuff. This setup enables the recording of specific manipulation primitives—characterized by start and end poses and action sequences—creating modular components that can be processed using frameworks like Dynamic Movement Primitives (DMPs) to form a structured model of the learned task.

**Teleoperation** Teleoperation is a versatile method in LfD where a human operator remotely controls a robot to perform tasks, typically using a control device or interface to guide the robot’s actions. During teleoperated demonstrations, the robot records data from its own sensors, capturing movements directly on the robot to ensure precise mapping without needing intermediate transformations. This approach enables the robot to learn complex tasks efficiently by observing human input in real-time.

The teleoperation process can be executed with various input devices, including control boxes, haptic interfaces, and virtual-reality systems, which allow for remote interaction with the robot. Unlike kinesthetic demonstrations, teleoperation does not require the user to be physically near the robot, making it suitable for remote settings and opening opportunities for crowdsourced data collection from multiple demonstrators.

Teleoperation has been applied in a range of applications, such as:

- **Assembly tasks:** The robot mirrors the human’s assembly motions, tracking pose information in real-time to replicate the demonstrated movements precisely.
- **Collaborative tasks:** Teleoperation is useful for tasks that require both dynamic and communicative input, enabling the robot to perform spatial positioning, grasp preshapes, and pick-and-move operations.
- **Humanoid and complex robotic control:** Systems like NASA’s Robonaut use full-immersion teleoperation, allowing the robot to transmit visual and auditory data to a human operator wearing a helmet for sensory feedback. Although this offers high fidelity control, it can be challenging due to the intensity and precision required from the operator.

One unique aspect of teleoperation is the ability to deliver “hints” to the robot. Operators may provide guidance by repeating tasks or highlighting key aspects of a skill. For example, operators can use vocal cues or gestures to emphasize elements within a task, accelerating the

robot’s learning process. Studies have shown that integrating vocal cues into the robot’s weight functions can enhance feature relevance, aiding in segmentation and manipulation tasks.

In addition to direct control, teleoperation in LfD includes understanding the human demonstrator’s intent. By analyzing vocal tone or body language, robots can learn to interpret the affective intent behind commands, going beyond literal task replication to capture goal-oriented behavior. This understanding enhances the robot’s ability to recognize the objectives behind demonstrated tasks, paving the way for social and cognitive learning, where robots understand human intentions and even imperfectly demonstrated examples.

Despite its advantages, teleoperation has some limitations, such as the additional training required for operators to use specific input devices effectively, and the availability of the necessary hardware.

**Passive Observation** Passive Observation is a demonstration approach where the robot learns by observing the human perform a task without any direct interaction. In this method, the demonstrator executes the task using their own body, sometimes with the assistance of additional sensors to facilitate tracking. The robot remains a passive observer throughout the process, recording movements for later analysis and imitation. This approach, often referred to as imitation learning, is particularly simple for the human demonstrator as it requires minimal training.

Passive Observation is especially suited to robots with many degrees of freedom or non-anthropomorphic robots, where kinesthetic teaching would be challenging. However, this approach introduces technical challenges, such as the need to translate human movements into actions executable by the robot and handling issues like occlusions, rapid movement, and sensor noise. Despite these challenges, Passive Observation has been successfully applied in a variety of tasks, including collaborative furniture assembly, autonomous driving, table-top tasks, and knot tying.

**Motion-Sensor Demonstration** Motion-Sensor Demonstration is a method where the complex limb movements of a human demonstrator are captured using advanced tracking devices, such as optical or magnetic markers, gloves with LEDs and tactile sensors, or motion sensors. This approach allows for higher precision than computer vision, reducing issues like visual overlap and improving data accuracy. It relies on marker-based tracking to capture fine movements for complex tasks, such as robotic assembly.

In Motion-Sensor Demonstration, tracking devices capture position and orientation data to record full-body or limb-specific movements. For example, a glove with LEDs can monitor wrist orientation, while tactile sensors identify contact points with objects. This method is particularly effective for segmenting complex motion patterns into discrete or continuous actions, which is useful for tasks like peg-in-hole assembly and tasks requiring force control. Motion-Sensor Demonstration often involves manual segmentation and labeling to provide reference data for the robot’s learning process.

### 2.2.2 Feature Extraction

Once a demonstration is provided, it is crucial for the robot to extract relevant features from the observed behavior. Feature extraction enables the robot to generalize the demonstrated behavior to different contexts or goals, providing greater flexibility in task execution. [39] [32] [34] [29]

### Dynamic Movement Primitives (DMP)

Dynamic Movement Primitives (DMPs) are a key concept in Learning from Demonstration. DMPs offer a framework for encoding demonstrated behaviors into reusable movement patterns, which can be adapted to different start and goal positions. They provide a flexible way to model complex trajectories by decomposing the task dynamics into attractor systems that guide the robot’s motion. This section offers a brief introduction to DMPs, with a more detailed discussion provided in a dedicated chapter later in this thesis.

### Denavit-Hartenberg inspired Bidirectional Representation (DHB)

The Denavit-Hartenberg inspired Bidirectional Representation (DHB) is a foundational approach in Learning from Demonstration for describing and reproducing rigid body motions in an invariant and bidirectional manner. DHB offers a compact and robust representation that can adapt to various spatial configurations, maintaining invariances in properties such as rotation, translation, linear and angular scaling, and temporal variations. These attributes make DHB particularly suitable for recognizing and generating trajectories, even in complex and variable scenarios.

The bidirectional nature of the DHB representation allows for seamless conversion of trajectories between Cartesian and invariant spaces, ensuring accuracy in reconstructing the original trajectory and enabling robots to adapt flexibly to different interaction scenarios. Unlike other unidirectional representations or those requiring higher-order derivatives, DHB ensures numerical robustness and reduces sensitivity to noise, proving ideal for human-robot interaction applications in realistic environments. This section introduces the principles of DHB, with a more detailed discussion provided in a dedicated chapter later in this thesis.

### Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) are probabilistic models widely used in Learning from Demonstration to encode and generalize demonstrated trajectories. An HMM is a statistical model used to describe a Markov process with unobserved (hidden) states, and it can be considered the simplest form of a dynamic Bayesian network. In the HMM framework, the observable outputs depend on the hidden states, which themselves are characterized by probabilities that define transitions between states. This structure allows HMMs to manage temporal sequences and spatial variables through a probabilistic approach, making it robust for segmenting and recognizing movement patterns in robotic tasks.

HMMs are particularly useful in modeling trajectories where human demonstrations consist of sequences of positions and velocities, represented as a continuous HMM with states encoded by Gaussian Mixture Regression (GMR). Each Gaussian distribution within the HMM is associated with a center and covariance matrix based on positional and velocity data, and these distributions are weighted to capture spatial and sequential information effectively. This approach allows HMMs to encapsulate both spatial and temporal variability, making them suitable for modeling complex motion patterns observed in demonstrations.

To improve stability in motion reproduction, several extensions to the standard HMM have been developed. Some approaches incorporate additional control mechanisms to ensure precise tracking of nonlinear dynamic movements, allowing robots to closely follow learned trajectories. Other adaptations focus on extracting and segmenting motion sequences from recorded trajectories, effectively modeling variability across spatial and temporal dimensions. These enhancements make HMMs versatile tools, as they can not only recognize specific motion patterns but also generate new motions based on previously learned behaviors, functioning as generative models.

### Gaussian Mixture Models (GMM) and Gaussian Mixture Regression (GMR)

Gaussian Mixture Models (GMMs) are probabilistic models used for clustering and density estimation, making them valuable tools in Learning from Demonstration for representing and generalizing demonstrated movements. A GMM is composed of two main categories of parameters: mixture component weights, which define the proportion of each Gaussian distribution in the model, and component means and variances (or covariances), which characterize the shape and position of each Gaussian component in the data space. One advantage of GMMs is that they do not require prior knowledge about the classification of data points, allowing the model to automatically learn the underlying structure in a dataset.

In robotic applications, GMMs are effective for motion modeling due to their robustness to noise. However, in high-dimensional spaces or when data samples are noisy or limited, GMMs with full covariance matrices can be prone to overfitting, where the model becomes too closely aligned with the sample data and loses generalizability. To address this, a technique called semi-tied GMM is often used. This approach decomposes the covariance into a common latent feature matrix and a component-specific diagonal matrix, forcing the mixture components to align along a set of common coordination patterns. These patterns can then be reused across different parts of the skill-learning process, enhancing the model’s robustness and adaptability.

Combined with Gaussian Mixture Regression (GMR), GMMs allow robots to predict plausible trajectories based on the statistical properties of the demonstrated data, adapting movements to different initial and goal conditions. This combination of GMM and GMR is particularly advantageous for tasks that require flexibility and real-time adaptability, as it enables robots to generalize learned movements efficiently.

#### 2.2.3 Color and Shape Detection for Cable Connector Identification

One of the core challenges in robotic manipulation tasks is accurately identifying and localizing objects within complex and dynamic environments. In the context of this project, a customized perception system was developed to detect the cable connector based on its color and shape, allowing for precise estimation of its position and orientation in 3D space. This process combines color segmentation, depth analysis, and axis detection using advanced computer vision techniques and real-time processing pipelines.

As shown in Algorithm 1, the ImageDepthProcessor pipeline integrates multiple stages, including HSV-based segmentation, ArUco marker detection, and pose estimation. This approach ensures robust and real-time perception capabilities for detecting cable connectors and their 3D orientation.

#### Color Segmentation and Depth Analysis

The detection process begins with color segmentation, which isolates the connector region from the scene. The approach leverages the HSV (Hue, Saturation, Value) color space for robust color-based filtering. This is particularly advantageous over RGB for this task due to its resilience to variations in lighting conditions. The following steps outline the segmentation process:

1. **HSV Thresholding:** The system applies predefined lower and upper bounds to segment pixels matching the connector’s color. These bounds are dynamically adjustable through a real-time trackbar interface, allowing for fine-tuning during calibration.

$$\text{Mask}_{\text{connector}} = \text{cv2.inRange}(\text{HSV Image}, \text{Lower Bound}, \text{Upper Bound})$$



2. **Contour Extraction:** Using the binary mask obtained from the thresholding step, contours are extracted to identify candidate regions corresponding to the connector. The largest detected contour is considered the primary region of interest.
3. **Depth Mapping:** The pixel coordinates of the segmented connector are projected into 3D space using depth data from the RGB-D camera. The depth value at each pixel is deprojected into Cartesian coordinates based on the intrinsic parameters of the camera:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (u - c_x) \cdot Z / f_x \\ (v - c_y) \cdot Z / f_y \\ Z \end{bmatrix}$$

where  $(u, v)$  are the pixel coordinates,  $Z$  is the depth value, and  $f_x, f_y, c_x, c_y$  are the intrinsic parameters.

### Connector Axis Detection and Orientation Estimation

Once the connector region is localized in 3D space, its orientation is determined by estimating its principal axis. This process is critical for guiding downstream manipulation tasks:

1. **3D Axis Detection:** The system identifies the two most significant points (e.g., end caps) on the connector using 3D geometry, forming a virtual axis. This axis provides an orientation reference in the workspace.
2. **Angle Calculation:** The angle of rotation about the base frame is computed using the vector formed by the detected axis. The angle  $\theta$  in the XY plane is determined as:

$$\theta = \arctan 2(\Delta Y, \Delta X)$$

where  $\Delta X$  and  $\Delta Y$  are the differences in the X and Y coordinates of the two end points.

3. **Smoothing for Stability:** To enhance robustness against noise, an exponential smoothing filter is applied to both position and orientation data:

$$S_t = \alpha \cdot X_t + (1 - \alpha) \cdot S_{t-1}$$

where  $\alpha$  is the smoothing coefficient,  $X_t$  is the current value, and  $S_{t-1}$  is the smoothed value from the previous iteration.

### Real-Time Visualization and Integration

To facilitate debugging and provide feedback during operation, the system includes real-time visualization of the detection results. The connector's 3D axis is overlaid on the video stream using OpenCV, and its orientation is displayed numerically on the frame. This feedback is essential for evaluating system performance and making adjustments during calibration.

Additionally, the detected 3D position and orientation are published as ROS topics, enabling integration with the robot's control system. These data serve as input for manipulation tasks, such as grasping or assembly, ensuring precise interaction with the connector.

### Advantages and Challenges

This approach combines geometric and color-based methods to achieve robust detection in cluttered and dynamic environments. The use of depth data enhances the system's ability to resolve occlusions and overlapping features, which are common in real-world scenarios. However, challenges remain in ensuring reliability under extreme lighting conditions or with objects of similar colors in proximity.

---

#### Algorithm 1: ImageDepthProcessor Workflow

---

**Input:** RGB frame  $rgb_{frame}$ , Depth frame  $depth_{frame}$   
**Output:** Grasping point  $P_{grasp}$ , Connector axis  $Axis_{3D}$ , Smoothed orientation  $\theta_{smooth}$

**Initialization**

- 1 Initialize ROS node and publishers for cable pose, connector axis, and debug frames;
- 2 Load camera intrinsics and setup trackbars for HSV segmentation;
- 3 Initialize ArUco detector and transformation utilities;
- 4 **while** frames are synchronized between RGB and Depth **do**
  - Step 1: Convert Input Frames**
  - 5  $rgb_{frame} \leftarrow$  Convert raw RGB frame to OpenCV format;
  - 6  $depth_{frame} \leftarrow$  Convert raw depth frame to depth array;
  - Step 2: HSV Segmentation for Connector Detection**
  - 7 Convert  $rgb_{frame}$  to HSV color space;
  - 8 Apply HSV thresholds to extract connector mask  $mask_{connector}$ ;
  - 9 Identify contours and calculate bounding boxes for largest regions;
  - 10 Compute 3D coordinates of connector centers using camera intrinsics;
  - 11 Store valid points  $Points_{3D} \leftarrow [P_1, P_2, \dots]$ ;
  - Step 3: ArUco Marker Detection and Pose Estimation**
  - 12 Detect markers in  $rgb_{frame}$  and estimate poses ( $rvecs, tvecs$ );
  - 13 Apply exponential smoothing to stabilize detected poses;
  - 14 Publish smoothed marker poses  $Pose_{marker}$ ;
  - Step 4: Compute Connector Axis and Orientation**
  - 15  $Axis_{3D} \leftarrow$  Find principal axis between  $Points_{3D}$ ;
  - 16  $\theta \leftarrow$  Compute orientation angle from  $Axis_{3D}$ ;
  - 17  $\theta_{smooth} \leftarrow$  Apply smoothing to stabilize  $\theta$ ;
  - 18 Publish axis and orientation  $Axis_{3D}, \theta_{smooth}$ ;
  - Step 5: Visualize and Publish Results**
  - 19 Annotate  $rgb_{frame}$  with axis and grasping point;
  - 20 Publish debug frames and poses via ROS;
- 21 Save updated HSV parameters and shutdown gracefully;
- 22 **return** Grasping point  $P_{grasp}$ , Smoothed orientation  $\theta_{smooth}$ ;

---

## 2.3 Planning

The Planning section addresses the transformation of perceived data into executable action plans, ensuring that robots can generalize learned behaviors to new contexts and adapt to variability in the environment or task requirements. This chapter introduces three fundamental tools that contribute to flexible and adaptive planning:

- **Dynamic Movement Primitives (DMPs):** A flexible framework that allows for modeling and reproducing complex trajectories in a stable manner, ensuring generalization and adaptability.

- **Denavit-Hartenberg inspired Bidirectional Representation (DHB):** A bidirectional representation that enables the description of trajectories in an invariant space, eliminating dependencies on rotations, translations, or temporal variations.
- **Trajectory and Orientation Generalization:** A mathematical approach to adapting trajectories and orientations learned in one context to new start and goal configurations, enabling task execution in variable spatial and orientational setups.

### 2.3.1 Dynamic Movement Primitives (DMP)

*“Dynamic Movement Primitives (DMPs) are a versatile and flexible framework that allows robots to encode and reproduce complex movements in a stable and generalizable way.” [1]*

Dynamic Movement Primitives (DMPs) are a robust and flexible framework widely used in robotics for encoding and reproducing smooth, nonlinear trajectories learned from demonstrations. Their key strength lies in their ability to generalize across different goals, allowing robots to adapt movements to new situations. Furthermore, DMPs ensure global asymptotic stability, making them well-suited to dynamic and unstructured environments where task requirements may change unpredictably.

The DMP framework consists of two main components: a transformation system and a canonical system. The transformation system generates the trajectory, while the canonical system governs the temporal evolution of the movement, ensuring that the movement progresses smoothly towards the goal.

**Mathematical Formulation of DMPs** The DMP framework is built upon a system of differential equations designed to encode complex motor behaviors while ensuring stability and flexibility. This system can generalize learned movements to new goal states and handle variability in the task execution.

**Standard DMP Formulation** The standard formulation of DMPs, as introduced by [33], is based on a second-order dynamical system responsible for generating the desired movement trajectory. The system comprises two key components:

- **Transformation System:** A second-order system that encodes the movement dynamics and controls the trajectory.
- **Canonical System:** A first-order system that controls the evolution of the phase variable, ensuring smooth progression towards the goal.

The transformation system is mathematically expressed as:

$$\tau^2 \ddot{y} = \alpha_z (\beta_z (g - y) - \tau \dot{y}) + f(x)$$

Where:

- $\alpha_z$  and  $\beta_z$  are positive gains controlling convergence towards the goal  $g$ ,
- $\tau$  is a temporal scaling factor,
- $f(x)$  is the nonlinear forcing term that encodes the learned trajectory from demonstrations.

The canonical system modulates the speed of the movement via the phase variable  $x$ , which drives the forcing term  $f(x)$ . This ensures that the system smoothly converges to the goal state while maintaining flexibility and stability. This structure allows for online adjustments to the movement, providing smooth transitions even in dynamic environments [33].

The transformation system can also be represented in state-space form as:

$$\begin{aligned}\tau\dot{z} &= \alpha_z\beta_z(g - y) - \alpha_z z + g_f(x)(g - y_0)f_s(x) \\ \tau\dot{y} &= z\end{aligned}$$

Where  $z = \tau\dot{y}$  represents the velocity, and  $y_0$  is the initial position. The phase variable  $x$  is used to avoid direct dependency on time, and the temporal scaling factor  $\tau$  determines the duration of the movement.

The forcing term  $f_s(x)$  is typically represented as a weighted sum of Gaussian basis functions, ensuring that the trajectory learned from demonstrations is accurately captured:

$$f_s(x) = \phi(x)^T w$$

Where  $\phi(x) = [\phi_1(x), \dots, \phi_N(x)]^T$  are Gaussian functions, and  $w$  are the weights that determine the contribution of each function.

**Advanced DMP Formulation with Reversibility** Building upon the standard DMP formulation, more recent work has extended the framework to include reversibility, allowing for complex bidirectional movements. This is particularly useful in tasks like robotic assembly, where robots must reverse actions, such as in insertion and extraction operations. The advanced formulation presented by [28] decouples stiffness and damping parameters from the temporal scaling factor, enabling smoother transitions between forward and backward motion.

The advanced transformation system is expressed as:

$$\tau^2\ddot{y} = \alpha_z\beta_z(g - y) - \alpha_z\tau\dot{y} + g_f(x)(g - y_0)f_s(x)$$

Where:

- $g_f(x)$  is a gating function ensuring the forcing term vanishes smoothly as the movement approaches the goal,
- $f_s(x)$  modulates the learned trajectory.

This formulation provides enhanced flexibility in handling more complex movement trajectories, such as those requiring bidirectional behavior, making it particularly applicable to sophisticated robotic manipulation tasks [28].

### Properties of DMPs

The proposed DMP formulation retains all desirable properties of the original DMP. One of the key advantages of DMPs is their global asymptotic stability at the target. Additionally, the stable coordination of multiple Degrees of Freedom (DoFs) is similar to the original DMP formulation, using a common canonical system for all DoFs, ensuring smooth and consistent movement even in complex tasks requiring multiple coordinated actions.

Another important property of DMPs is their ability to scale spatially and temporally. When there is a change in the initial/target position or the temporal scaling parameter  $\tau$ , the generated trajectories remain qualitatively similar or topologically equivalent. This scaling property is

verified through simulations, demonstrating that for different initial/target poses or temporal scaling, the trajectories produced by the proposed DMP coincide with those of the original formulation.

DMPs also exhibit robustness to perturbations, such as external disturbances. In such cases, the phase stopping mechanism is employed, which results in the slowdown or even halt of the trajectory generation by modifying the canonical system's evolution. This can be expressed as:

$$\tau \dot{x} = \frac{h(x)}{1 + |ad(t)|}$$

Where  $d(t)$  represents the external disturbance, and  $a$  is a positive constant. Additionally, other forms of phase stopping, such as sigmoid stopping, can also be implemented.

Another important feature of DMPs is their ability to incorporate coupling terms, which allow the dynamical system to modify its trajectory based on external signals without the need for replanning. These coupling terms have been applied to adjust the trajectory in real-time based on external forces, enforce position or joint limits, and avoid obstacles. For instance, to avoid limits, a repulsive force can be added at the velocity level:

$$\begin{aligned}\dot{y} &= z - \gamma(y_L - y)^3 \\ \dot{z} &= \ddot{y} - D(z - \dot{y}) - K(y - y_L)\end{aligned}$$

Where  $y_L$  is the limit, and  $\gamma$  controls the effect of the repulsive force.

Obstacle avoidance can also be integrated as follows:

$$f_o = \gamma R y' \phi e^{-\beta \phi}$$

Where  $R$  is a rotation matrix, and  $\phi$  is the angle between the obstacle position  $p_o$  and the trajectory of the DMP. This allows the robot to dynamically adjust its movement to avoid obstacles during task execution.

### Applications of DMPs in Robotic Tasks

DMPs have been successfully applied across a wide range of robotic tasks, particularly in situations requiring precise and adaptable movement control. One notable application is in robotic assembly tasks, such as peg-in-hole operations, where DMPs encode demonstrated trajectories and adapt them to new goals in real-time.

During the teaching phase, robots record human demonstrations, capturing key information such as position, orientation, and forces during the task. This information is then encoded into the DMP framework, enabling robots to autonomously reproduce the task while generalizing it to different goals and conditions. DMPs provide stability and robustness, even in the presence of disturbances or changes in the environment, ensuring that the robot can execute tasks accurately and adaptively [11].

**Generalization and Adaptability in DMPs** A significant advantage of DMPs is their ability to generalize movements to different goal states or starting conditions. This capability is particularly valuable in scenarios where the robot must perform the same task in varying contexts, such as grasping objects of different sizes or shapes. The forcing term  $f(x)$  in the DMP formulation enables robots to adapt the learned movement to these new conditions while maintaining the core characteristics of the original trajectory [1].

This generalization feature has been applied successfully in tasks such as object manipulation and pick-and-place operations, where robots must handle variability in object size, weight, or position. DMPs allow robots to perform these tasks without requiring extensive reprogramming or additional demonstrations.

**Cooperation in Human-Robot Collaboration** DMPs have also been leveraged in cooperative human-robot tasks, where robots collaborate with humans to perform shared tasks such as co-manipulation. In these scenarios, the robot can adjust its movements in real-time based on human input, ensuring smooth and safe interactions. The flexibility and adaptability of DMPs make them particularly suitable for these applications, where precise coordination between human and robot actions is essential.

For example, in co-manipulation tasks, the robot can adjust its trajectory based on the force exerted by the human, allowing for seamless collaboration. This capability is enabled by the real-time adaptability of DMPs, which can respond dynamically to changes in the task or environment, providing a natural and intuitive interaction between human and robot [22].

### Advantages and Limitations of DMPs

Dynamic Movement Primitives (DMPs) offer several advantages, making them a powerful tool for robotic motion generation. However, as with any framework, they also present certain limitations that researchers continue to address.

**Advantages of DMPs** DMPs provide numerous benefits in robotic motion control, which are essential for dynamic and adaptive robotic tasks:

- **Generalization:** DMPs allow robots to adapt learned movements to new goals and contexts with minimal adjustment. This generalization is particularly valuable in tasks where robots must repeat similar actions across variable scenarios, as it reduces the need for reprogramming.
- **Stability:** The underlying dynamical system of DMPs ensures global asymptotic stability, which guarantees smooth convergence to the target position even in complex, high-dimensional tasks. This stability is especially beneficial in environments with unpredictable conditions.
- **Real-time Adaptation:** DMPs enable real-time adjustments to motion, which allows robots to respond to changes in the environment, disturbances, or modifications in task requirements. The phase stopping mechanism and coupling terms provide robustness to external forces and disturbances.
- **Modularity:** The modular structure of DMPs allows complex tasks to be achieved by sequencing or blending simpler movement primitives. This modularity provides flexibility in programming diverse tasks and combining different motion components to generate sophisticated behaviors.
- **Ease of Learning from Demonstration (LfD):** DMPs are well-suited for learning from demonstration, as they can encode trajectories from human-guided demonstrations, capturing essential kinematic features. This enables robots to learn and replicate complex movements without extensive manual coding.

**Limitations of DMPs** Despite their strengths, DMPs also present limitations that can constrain their use in certain applications:

- **Limited Flexibility in High-Dimensional Spaces:** While DMPs are effective for tasks with lower degrees of freedom (DoF), their scalability to high-dimensional systems remains challenging. Managing the complexity and interactions between multiple DoFs can become computationally expensive.

- **Dependency on Predefined Goal States:** DMPs require well-defined goal states for accurate trajectory generation. In cases where the goal state is not clearly specified or needs to be dynamically determined, additional components are often required to enhance flexibility.
- **Forcing Term Complexity:** The forcing term  $f(x)$  requires careful tuning of weights and Gaussian functions to capture the nuances of complex trajectories accurately. Improper tuning can result in suboptimal motion generation, limiting the accuracy and adaptability of the DMP.
- **Limited Task Reversibility:** Although advanced DMP formulations with reversibility exist, reversing learned tasks efficiently remains a challenge. This limitation can be critical in tasks like assembly/disassembly, where bidirectional movement is required.

Researchers continue to explore solutions to address these limitations, such as integrating machine learning models for adaptive goal-setting, optimizing the design of Gaussian basis functions in the forcing term, and developing hybrid systems that combine DMPs with other motion generation techniques for more complex, multi-DoF tasks.

### 2.3.2 Bidirectional Invariant Representation

The Bidirectional Invariant Representation offers a powerful framework for describing rigid body motions in a manner that is invariant to transformations such as rotations, translations, scaling, and time variations. This approach has been particularly effective in applications such as gesture recognition and reproduction, where robustness to variations in viewpoint and execution speed is crucial [20].

#### What is an Invariant Representation?

An invariant representation is a mathematical approach for describing objects or motions in a way that is resistant to specific transformations, such as rotation, translation, scaling, or temporal variations. This concept is foundational in fields such as robotics, computer vision, and pattern recognition, where it is essential to recognize actions, gestures, or objects despite changes in position, orientation, or size.

In robotics, invariant representations allow systems to interpret actions consistently across different viewpoints or environmental conditions, a requirement for effective human-robot interaction (HRI). For instance, in gesture recognition, the position and movement of a human’s arm may vary in absolute terms, but an invariant representation abstracts these motions to capture the essential structure of the action, enabling recognition independent of initial pose or speed [31][30][36].

**Principles of Invariant Representations** The core principle behind invariant representations is to transform data from its original space into a feature space that filters out non-essential information, preserving only the attributes relevant to the task at hand. For example, curvatures and spatio-temporal curvature can be employed as invariant descriptors to capture critical aspects of motion. Spatio-temporal curvature, in particular, is effective in identifying “dynamic instants” (i.e., significant changes in speed or direction), which are essential for interpreting complex gestures and motions.

**Types of Invariance in Robotics** In robotic gesture recognition and motion reproduction, invariant representations are designed to ensure that systems can handle:

- **Rotational and translational invariance:** Essential in tasks where the robot observes actions from different viewpoints. Rototranslation invariance is achieved by using descriptors that maintain their value under changes in reference frame, supporting stable recognition across multiple angles and perspectives. This principle is particularly valuable in HRI, where it eliminates the dependency on specific spatial configurations of the user relative to the robot sensor.
- **Temporal invariance:** Critical for actions performed at varying speeds. By normalizing time-dependent features, invariant representations enable consistent interpretation of gestures regardless of the speed of execution. Temporal alignment techniques, such as dynamic time warping, enhance this capability by synchronizing similar actions with different temporal characteristics, thus aiding in action recognition across different speeds.
- **Scaling invariance:** Useful in scenarios where gestures or movements vary in size. Linear scaling invariance allows invariant representations to handle variations in amplitude or spatial extent, facilitating gesture recognition in users of different body sizes or with gestures that vary in range.

**Applications in View-Invariant Action Recognition** Recent approaches to action recognition, especially in the domain of computer vision, emphasize view-invariant methods to ensure robust recognition from various angles and perspectives. By defining actions as sequences of "dynamic instants" and "intervals," researchers have demonstrated that spatio-temporal curvature can effectively represent human actions as consistent patterns, even when observed from different views. For example, changes in trajectory curvature can indicate specific action segments (e.g., picking up, lifting, or releasing), which are central to view-invariant recognition frameworks.

### Principles of Bidirectional Invariant Representation

The fundamental principle behind this framework is to transform motion trajectories into a space that is invariant to common transformations, while still retaining the ability to reconstruct the original motion. This is achieved by mapping the original motion in Cartesian space to an invariant space through a set of transformations that preserve the underlying structure of the movement but remove any dependency on the specific coordinate frame or scale.

In the context of gesture recognition and reproduction, this allows for the recognition of gestures even when they are performed with variations in speed, orientation, or position. The bidirectional nature of the representation means that not only can gestures be recognized from their invariant representation, but they can also be reconstructed in Cartesian space with high fidelity.

### Mathematical Formulation of DHB

The mathematical formulation of the Bidirectional Invariant Representation is inspired by the Denavit-Hartenberg (DHB) convention, commonly used in robotics to describe the configuration of rigid body systems. In this approach, rigid body motions are represented using a minimal set of six parameters: three for the position and three for the orientation.



**Definition of Position and Orientation Frames:** To achieve an invariant description of motion, two separate frames are defined: one for the position and one for the orientation. The position at each time  $t$  is represented by a vector  $p(t)$ , while the orientation is described by a minimal rotation vector  $r(t)$ , which captures the rotation as an axis-angle pair.

**Axis Assignment within Frames:** - The axes of the position frame are defined as:

$$\begin{aligned} x_p(t) &= \frac{p(t + \Delta t) - p(t)}{\|p(t + \Delta t) - p(t)\|}, \\ y_p(t) &= \frac{x_p(t) \times x_p(t + \Delta t)}{\|x_p(t) \times x_p(t + \Delta t)\|}, \\ z_p(t) &= x_p(t) \times y_p(t). \end{aligned}$$

- For the orientation frame, we use a normalized rotation vector:

$$x_r(t) = \frac{r(t)}{\|r(t)\|},$$

with  $y_r(t)$  and  $z_r(t)$  defined analogously through cross-products.

**Invariant Position and Orientation Values:** - The invariant values describing position and orientation between consecutive frames are computed as:

$$\begin{aligned} m_p(t) &= \|p(t + \Delta t) - p(t)\|, \\ m_r(t) &= \|r(t + \Delta t) - r(t)\|. \end{aligned}$$

These invariants capture the translational and rotational motion components between frames, preserving essential motion details while removing dependency on absolute positioning.

**Rotational Alignment Angles:** - To align subsequent frames, the angles  $\theta_1$  and  $\theta_2$  are calculated for both position and orientation frames:

$$\begin{aligned} \theta_1^p &= \arctan \left( \frac{x_p(t) \times x_p(t + 1)}{x_p(t) \cdot x_p(t + 1)} \cdot y_p(t) \right), \\ \theta_2^p &= \arctan \left( \frac{y_p(t) \times y_p(t + 1)}{y_p(t) \cdot y_p(t + 1)} \cdot x_p(t + 1) \right), \end{aligned}$$

with similar expressions for the orientation frame's angles  $\theta_1^r$  and  $\theta_2^r$

**Invariance Properties:** The DHB representation ensures multiple invariance properties that are critical for tasks involving gesture recognition and robotic motion reproduction:

- **Invariance to rotation and translation:** The representation is unaffected by the initial position or orientation, allowing for robust recognition across varying viewpoints.
- **Temporal invariance:** By normalizing temporal aspects, DHB supports gesture recognition at varying speeds.
- **Scalability:** DHB's scaling invariance accommodates gestures or movements of different sizes, useful when interacting with users of different body dimensions.

These invariance properties make the Bidirectional Invariant Representation a powerful tool for human-robot interaction, where variations in user execution, perspective, and environment are common. The ability to reconstruct motion from invariant values further enhances its application potential in tasks that require both recognition and precise reproduction of gestures [20].

### 2.3.3 Trajectory and Orientation Generalization

Generalization is a critical aspect of planning that enables robots to adapt learned trajectories and orientations to new starting and goal conditions. This capability is particularly valuable in dynamic environments or tasks where the spatial and rotational configuration of objects can vary. This section introduces the mathematical framework used for trajectory and orientation generalization, ensuring adaptability and precision.

#### Generalization of Trajectory

The process of trajectory generalization involves scaling and transforming a learned trajectory to align with new start and goal positions. The following equations define the generalization process:

$$\tilde{x}_{td}(t) = s_L(t)R(t)(x_{td}^d - x_1^d) + x_{\text{start}}(t)$$

Where:

- $\tilde{x}_{td}(t)$ : Generalized position trajectory at time  $t$ ,
- $x_{td}^d$ : Original learned trajectory,
- $x_1^d$ : Initial position of the learned trajectory,
- $x_{\text{start}}(t)$ : New start position,
- $R(t)$ : Rotation matrix aligning the learned trajectory with the new goal,
- $s_L(t)$ : Scaling factor defined as:

$$s_L(t) = \frac{\|x_{\text{goal}}(t) - x_{\text{start}}(t)\|}{\|x_{Td}^d - x_1^d\|}$$

Where  $x_{\text{goal}}(t)$  represents the new goal position, and  $x_{Td}^d$  is the final position of the learned trajectory.

Additionally, the generalized velocity trajectory is computed as:

$$\mathbf{v}_{\text{gen}}(t) = s_L(t)R(t)\mathbf{v}_{\text{learn}}(t)$$

Where  $\mathbf{v}_{\text{learn}}(t)$  is the learned velocity trajectory.

#### Generalization of Orientation

Orientation generalization is achieved by scaling Euler angles based on the relative difference between the start and goal orientations. The process is defined as follows:

$$\Delta e_{\text{original}} = e_{\text{goal}} - e_{\text{start}}$$

$$\Delta e_{\text{new}} = e_{\text{goal,new}} - e_{\text{start}}$$

The scaling factor for each axis is computed as:

$$s = \frac{\Delta e_{\text{new}}}{\Delta e_{\text{original}}}$$

The generalized orientation trajectory is obtained by scaling the difference between the trajectory and the start orientation:

$$\begin{aligned}\Delta e_{\text{traj}}(t) &= e_{\text{traj}}(t) - e_{\text{start}} \\ \Delta e_{\text{scaled}}(t) &= \Delta e_{\text{traj}}(t) \cdot s \\ e_{\text{scaled}}(t) &= e_{\text{start}} + \Delta e_{\text{scaled}}(t)\end{aligned}$$

This method ensures that the orientation trajectory adapts to new conditions while maintaining the relative shape of the original trajectory.

## 2.4 Execution

The Execution chapter focuses on the implementation and control of the generated action plans, with particular emphasis on real-time adaptability and modular execution. In this phase, robots must translate plans into concrete behaviors, dynamically reacting to changes in the environment or unforeseen events. Behavior Trees (BTs) serve as the central framework described in this chapter, offering a hierarchical and modular structure for managing robotic actions. Their tree-like architecture allows the decomposition of complex behaviors into simpler sub-behaviors, ensuring scalability, reusability, and immediate response to changes.

The chapter highlights the advantages of BTs over other control architectures, such as Finite State Machines (FSMs), emphasizing their flexibility and integration with platforms like ROS. It also describes how BTs support real-time control by combining environmental conditions and predefined actions to ensure robust and responsive execution.

### 2.4.1 Behavior Trees

The Behavior Tree (BT) is a powerful control architecture widely used in robotics, artificial intelligence, and video games. BTs allow autonomous systems to execute actions in a modular and hierarchical fashion, enabling better scalability, flexibility, and reusability compared to traditional control systems such as Finite State Machines (FSMs) [14]. This section will introduce the key components and structure of BTs, followed by an exploration of their advantages and applications in robotics.

#### Key Components and Structure

A Behavior Tree consists of two main types of nodes: control flow nodes and execution nodes. Control flow nodes dictate the flow of execution, while execution nodes represent actions or conditions that the robot must evaluate or perform. The BT starts at the root node and executes nodes based on a recursive tick system, propagating from parent to child nodes.

As shown in Table 2.2, the Behavior Tree nodes have different outcomes based on their execution flow:

- **Sequence Node:** This node executes its children in order until one returns failure or is running. If all children succeed, the sequence node returns success.
- **Fallback (Selector) Node:** This node checks its children sequentially until one returns success or is running. It is typically used to implement behaviors where alternative actions must be tried until one succeeds.

- **Action Nodes:** These nodes execute tasks or actions, such as moving the robot or grasping an object. They return success, failure, or running based on the state of the action.
- **Condition Nodes:** These nodes check whether a condition holds (e.g., whether an object is detected) and return success or failure.
- **Decorator Nodes:** These modify the behavior of other nodes, such as retrying a node a set number of times or inverting its result.

Node type	Symbol	Succeeds	Fails	Running
Fallback	?	If one child succeeds	If all children fail	If one child returns Running
Sequence	$\rightarrow$	If all children succeed	If one child fails	If one child returns Running
Parallel	$\Rightarrow$	If $\geq M$ children succeed	If $> N - M$ children fail	else
Action	text	Upon completion	If impossible to complete	During completion
Condition	text	If true	If false	Never
Decorator	$\diamond$	Custom	Custom	Custom

Table 2.2: Behavior Tree node types and their respective outcomes.

The modular structure of BTs allows behaviors to be divided into smaller, reusable sub-behaviors. Each sub-behavior can be independently designed and tested, which enhances the system’s maintainability and readability [14].

### Advantages and Applications in Robotics

BTs offer significant advantages over traditional control architectures like FSMs. These advantages include:

- **Modularity:** BTs promote the creation of reusable and interchangeable behaviors, allowing developers to easily modify or extend the system.
- **Reactivity:** The tick-based execution ensures that BTs can react to dynamic changes in the environment, providing a level of adaptability that is essential in unstructured settings.
- **Scalability:** Unlike FSMs, BTs do not suffer from the state explosion problem. The hierarchical nature of BTs allows complex behaviors to be composed of smaller, simpler sub-behaviors, making it easier to manage large systems.
- **Human-Readable Structure:** The tree structure of BTs is easy to visualize, which simplifies debugging and comprehension for developers.

In robotics, BTs have been successfully implemented in various applications, including robotic manipulation, navigation, and multi-agent coordination. For example, BTs are commonly used in pick-and-place operations, where a robot must grasp and move objects in a dynamic environment. The modularity of BTs allows these operations to be defined in a flexible manner, facilitating the addition of fallback behaviors in case of failure. Moreover, in multi-agent systems, BTs have proven effective in managing complex behaviors across teams of robots, enabling real-time decision-making and execution.

### Implementation of Behavior Trees in ROS

Behavior Trees have been integrated with the Robot Operating System (ROS), which provides a robust platform for building complex robotic applications. There are several libraries available for implementing BTs in ROS, such as *py\_trees* and *BehaviorTree.CPP*. These libraries offer pre-defined nodes and tools for integrating BTs with ROS topics, services, and actions [25].

A common implementation strategy involves the following steps:

- **Defining the Tree Structure:** The tree structure is defined using a combination of action, condition, and control flow nodes. These nodes are linked to specific ROS functionalities, such as subscribing to topics or invoking actions.
- **Creating Custom Nodes:** Developers can create custom nodes to interface with the robot's hardware, such as sensors and actuators. These nodes can be written in C++ or Python and integrate directly with ROS messages.
- **Execution of the Tree:** Once the tree is defined, it is continuously ticked by the BT engine, allowing real-time decision-making based on sensor inputs and action feedback.

This approach facilitates the integration of BTs into robotic systems, allowing for flexible, modular, and reactive control strategies. Furthermore, ROS enables seamless communication between the behavior tree nodes and the robot's hardware components, enhancing the adaptability of the system in dynamic environments.

## Chapter 3

# Materials and Equipment

### 3.1 Introduction

The study of robotic manipulation has grown increasingly significant in recent years, as robotics systems are required to perform complex and precise tasks across diverse environments and applications. Achieving reliable manipulation capabilities requires a careful balance of hardware, software, and experimental design to ensure flexibility, control, and adaptability in real-world settings.

This chapter details the setup and configuration of a custom experimental environment specifically designed to support robotic manipulation research. This environment includes a modular assembly board, a robotic arm and various tools for manipulation. These components work together to simulate a variety of assembly tasks that involve both rigid and deformable objects, with particular focus on an Ethernet cable as a representative deformable object.

The goal of this setup is to enable the robotic arm to perform manipulation tasks autonomously while maintaining precision and control. Key aspects covered in this chapter include the design and purpose of the custom assembly board, the selection and arrangement of experimental materials, the hardware equipment utilized, and the software tools required to operate the system. Each section provides a comprehensive description of the components involved, alongside their roles and technical specifications.

### 3.2 Custom Assembly Board for Robotic Manipulation

Robotic manipulation is a field that requires precise engineering solutions to enhance the ability of robots to interact with the environment and autonomously manipulate objects. One of the fundamental aspects of this manipulation is the creation of modular surfaces that allow the robotic system to perform a variety of operations in a flexible and efficient manner.

The basic idea behind the design of the board is inspired by similar solutions adopted in the field of robotic assembly, as described by the National Institute of Standards and Technology (NIST).

NIST has developed multiple modular platforms for robotic assembly that facilitate the manipulation and grasping of objects through flexible configurations that support a wide range of tasks [27]. These platforms allow the simulation of different working conditions, offering a controlled yet highly adaptable environment for testing robotic manipulation algorithms and methodologies.

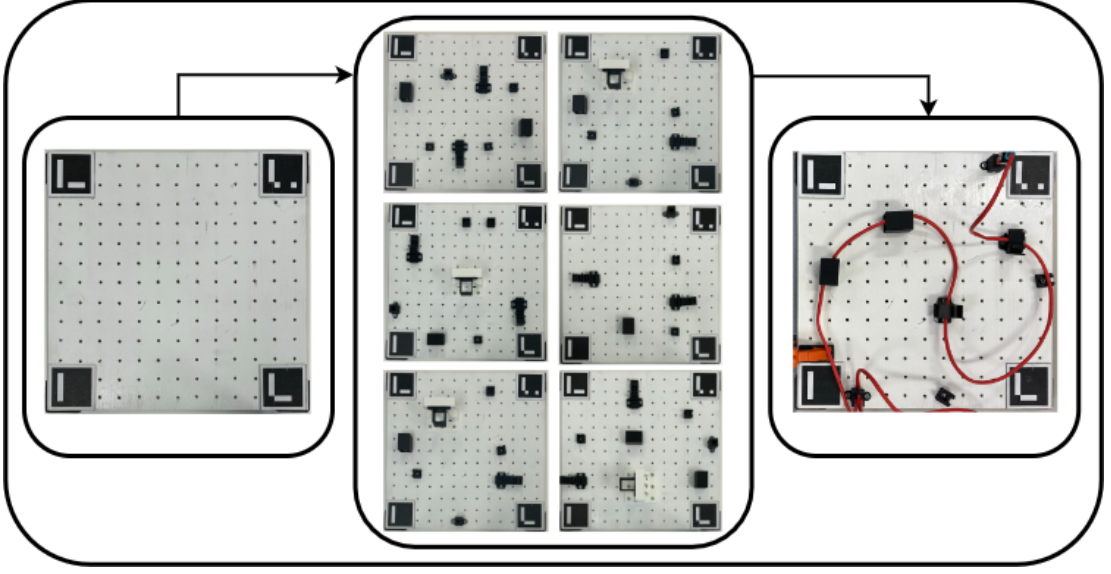


Figure 3.1: Modular assembly board configurations: base setup (left), various tool arrangements (center), and cable manipulation scenario (right).

Building on these principles, I have designed a custom solution aimed at supporting the manipulation of both deformable and rigid objects within a demonstration learning context. The design of the board and its technical specifications are outlined below.

### 3.2.1 Inventory of Experimental Equipment and Materials

This section summarizes, in the Table 3.1, the key materials and components employed during the experimental phase. Each component plays a specific role in testing and validating the robotic manipulation system.

#### Board Design and Purpose

The basic idea is to draw inspiration from one of the boards presented by NIST [27], and adapt it to better suit the project’s needs. This board was designed to serve as a versatile platform for robotic manipulation tasks, with the primary goal of simulating a range of assembly scenarios in a controlled and adaptable environment. The design of the board is modular, and unlike the examples taken as inspiration, it offers greater versatility, allowing multiple configurations that support different manipulation objectives, such as grasping, sorting, and assembling both rigid and deformable objects.

As shown in Figure 3.1, a more flexible approach was adopted, allowing the creation of multiple patterns with the help of a hole matrix, enhancing the robot’s ability to adapt to different conditions. As illustrated, the board can be configured in various ways to meet the needs of each experiment. These configurations were chosen to test the robot’s ability to manipulate objects of varying shapes, sizes, and materials, providing a robust platform for evaluating robotic grasping and manipulation strategies.

### 3.2. CUSTOM ASSEMBLY BOARD FOR ROBOTIC MANIPULATION

Component	Description	Material
Board	400 mm x 400 mm x 20 mm board with 169 holes	Wood
Aruco Marker	4 placed on the corners, 4 placed on the connector of the cable	Paper (printed)
Feet Supports	4 supports with rubber anti-slip bearings	Rubber
Grasped Objects	Various shapes (tubes, clips, holders.)	Plastic
Ethernet Cable	Flexible deformable cable	Rubber
Connectors	Electrical connectors for assembly	Plastic, Metal
Blue Tape	Required for cable detection	Stretch fabric
6-Axis Force-Torque Sensor	Used for precise force feedback during manipulation	Aidin Robotics (Model X)
Camera	Intel RealSense depth camera	-
Robotic Arm	Franka Emika Panda (FEP)	-
Assembly Tools	-	-

Table 3.1: List of Materials and Components Used in Experiments

Parameter	Description
Dimensions	400 mm x 400 mm x 20 mm
Material	Wood with a perforated matrix
Hole diameter	4 mm
Grid spacing	25 mm between holes, 2 mm from the edge
Hole grid	169 holes (13 x 13 matrix)
Surface finish	White spray paint
Aruco markers	4 markers, 7 cm x 7 cm, placed on the four corners
Feet supports	4 supports with 5 rubber anti-slip bearings each
Mounting compatibility	Compatible with Franka Emika Panda robotic arm
Configurations supported	Multiple configurations possible due to the hole matrix, with a few examples shown in Figure 3.1

Table 3.2: Technical Specifications of the Custom Board

#### Objects and Components for Manipulation

The robotic manipulation procedure was designed to address a series of challenges related to the handling of both rigid and deformable objects. These include a variety of geometric shapes used to interact with the only deformable object considered: the Ethernet cable.

The 3D models employed were inspired by examples presented by NIST, as they were already suited for approaches similar to the final project. These models were subsequently revised and customized to better fit the specific requirements of the robotic manipulation system and the experimental setup.

All objects were 3D printed using plastic material and were carefully adapted to the dimensions of the board and the cable used in the experiments. A detailed list of the 3D-printed tools, including their names, technical specifications, and representative images, is provided in Appendix A.

In addition to the objects for manipulation, the system utilizes additional components, such



as the Ethernet cable and, specifically, its connector, which was the subject of a detailed study to ensure it could be easily handled by the robotic arm. Two main modifications were made to facilitate the grasping and recognition of the connector:

- The use of colored tape to improve the visibility and localization of the connector.
- The addition of Aruco markers via a specific support to enhance tracking and precision during manipulation.

The goal of the manipulation procedure is to assess the robot's effectiveness in executing tasks such as grasping, moving, and assembling, while maintaining high precision and force control. These tasks are carried out with the assistance of force-torque sensors integrated directly into the robotic arm, which monitor real-time contact between the robot and the object, ensuring optimal and safe interaction.

### 3.3 Hardware Equipment

This section provides an overview of the key hardware components used in the robotic manipulation system.

#### 3.3.1 Robotic Arm (Franka Emika Panda)

The FEP robotic arm is a high-precision manipulator designed for complex robotic tasks. It has 7 degrees of freedom, which allow it to perform dexterous manipulations in a wide range of scenarios. The arm is equipped with integrated force-torque sensors in its joints, enabling it to interact delicately with objects while monitoring contact forces in real-time. This feature is particularly useful for tasks that require careful manipulation, such as handling deformable objects like the Ethernet cable. The robotic arm is responsible for executing all the manipulation tasks, including grasping, moving, and assembling objects on the custom board.

Parameter	Specification
Degrees of Freedom (DOF)	7
Payload	3 kg
Reach	855 mm
Repeatability	$\pm 0.1$ mm
Joint Torque Sensors	Integrated in all 7 axes
Force-Torque Sensing	Integrated in the end effector
Maximum Joint Velocity	2.15 rad/s
Control Interface	ROS-compatible, high-level APIs

Table 3.3: Technical Specifications of the Franka Emika Panda Robotic Arm

#### 3.3.2 Camera (Intel RealSense D435i)

The Intel RealSense D435i is a depth camera that provides critical visual feedback and depth perception, essential for accurate object tracking and manipulation. This camera is equipped with a wide field of view ( $86^\circ \times 57^\circ$ ) and an RGB sensor with a resolution of up to  $1920 \times 1080$  pixels. It also includes an IMU (Inertial Measurement Unit) that allows for tracking the orientation and movement of the camera, improving spatial awareness in dynamic environments.

In the context of this project, the D435i is used to monitor the 3D position and orientation of objects such as the Ethernet cable and its connector. The depth camera provides detailed information with a range of up to 10 meters, enabling precise alignment during manipulation tasks. The depth data also assist in detecting potential collisions and improving the accuracy of the robot's movements.

Parameter	Specification
Depth Field of View (FOV)	86° (H) x 57° (V)
Depth Resolution	Up to 1280x720 pixels
RGB Resolution	Up to 1920x1080 pixels
Maximum Range	10 meters
Frame Rate (Depth)	Up to 90 fps
Inertial Measurement Unit (IMU)	6 DOF (Gyroscope and Accelerometer)

Table 3.4: Technical Specifications of the Intel RealSense D435i Camera

## 3.4 Software Tools

This section describes the main software tools used for controlling the robotic manipulation system and supporting the experimental setup.

### 3.4.1 ROS (Robot Operating System)

ROS is an open-source middleware widely used in robotics to manage communication between different components of a robotic system. It provides a collection of libraries and tools that help developers build robotic applications.

### 3.4.2 Aruco Libraries

The Aruco libraries are a set of functions used to detect Aruco markers in the environment, providing robust and accurate tracking of objects and coordinates. These libraries are integrated into the system to detect and track the Aruco markers placed on the assembly board and the Ethernet cable connector.

### 3.4.3 Other Software Tools

Additional software tools were employed to support the manipulation experiments, including:

- **Python:** The main programming language used to develop control algorithms and interface with hardware components.
- **Gazebo:** A robotics simulator used to test algorithms in a virtual environment before applying them to the real robot.
- **Rviz:** A 3D visualization tool used in ROS to visualize sensor data, the robot's state, and the environment.
- **OpenCV:** A computer vision library used for image processing and object detection, providing support for tracking the Ethernet cable and its connector.



## Chapter 4

# System Design and Development

### 4.1 Introduction

The design and development of an autonomous robotic manipulation system involve a careful balance of hardware integration, software architecture, and system optimization. This chapter provides an in-depth exploration of the engineering decisions and implementation strategies adopted to enable the robotic system to handle both rigid and deformable objects in a dynamic environment.

Key elements of the system include the integration of advanced sensors and actuators, the use of a modular assembly board, and the implementation of a robust software architecture based on the Robot Operating System (ROS). The chapter begins by outlining the system requirements and design goals that guided the development process. Following this, it details the integration of hardware components, the synchronization of sensors and actuators, and the challenges encountered during the implementation phase.

Additionally, the software architecture is described, highlighting the communication strategies and motion planning techniques that ensure the system's efficiency and adaptability. Finally, the chapter concludes with a discussion of potential future improvements to further enhance the system's capabilities and scalability.

### 4.2 System Requirements and Design Goals

The primary objective of the robotic manipulation system is to enable the autonomous handling of both rigid and deformable objects in a dynamic and controlled environment. This requires a combination of advanced hardware integration, precise motion planning, and reliable software communication. This section outlines the functional requirements and design goals that guided the system's development.

#### 4.2.1 Functional Requirements

To achieve the desired level of autonomy and precision, the system must meet the following functional requirements:

- **Object Detection and Localization:** The system must accurately detect and track objects, including deformable components, using visual markers (e.g., Aruco markers), color and shape detection, and depth cameras.

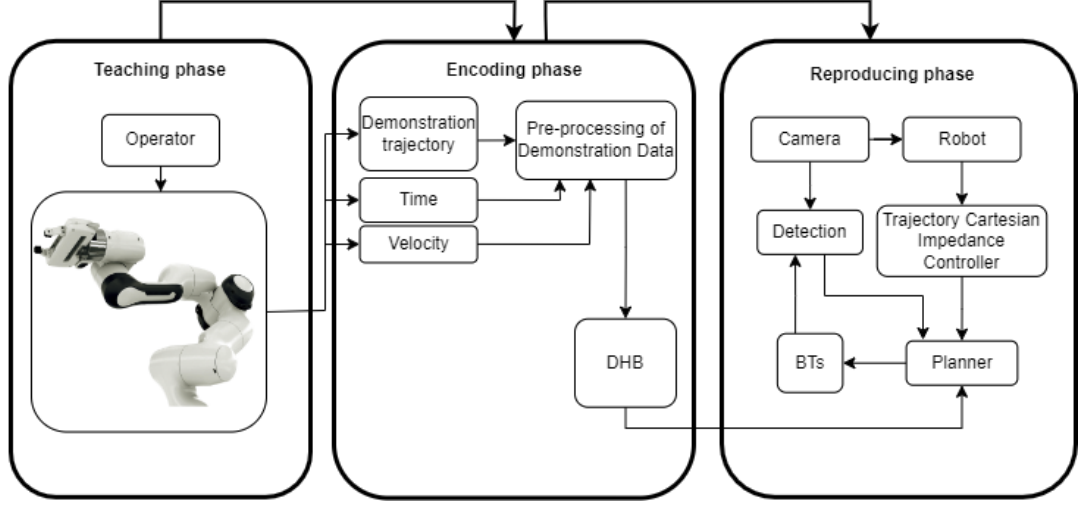


Figure 4.1: System design illustrating the teaching, encoding, and reproducing phases for trajectory learning and execution using invariant representations and Cartesian impedance control.

- **Precise Manipulation:** The robotic arm must be capable of performing fine manipulation tasks such as grasping, moving, and assembling objects while maintaining control over force and position. For this purpose, the inclusion of a pair of fingertips for the end effector (EE) is critical for accurate and functional manipulation.
- **Adaptability to Variability:** The system must handle variability in object positions, orientations, and environmental conditions, demonstrating flexibility in real-world scenarios.
- **Robust Data Flow:** Establish seamless communication between sensors, actuators, and the control system to ensure real-time responsiveness.

#### 4.2.2 Design Goals

The system was developed following these design principles:

- **Modularity:** Design a system architecture that allows for the easy integration of additional hardware or software components. Particular attention was given to the board, inspired by the NIST project [27], which enables countless configurations for arranging tools on it. Unlike the original design, which features a predefined configuration, this board provides significant flexibility. Additionally, the use of Behavior Trees (BTs) ensures a degree of hardware flexibility in completing tasks, allowing the system to adapt to different configurations and hardware setups without significant reprogramming.
- **Scalability:** The system is designed to support future upgrades, including handling more complex tasks or integrating additional robotic capabilities such as dual-arm manipulation or more sophisticated sensors.
- **Precision and Reliability:** Emphasize precision in manipulation tasks while ensuring consistent performance under variable conditions. For example, trajectory generalization

## 4.2. SYSTEM REQUIREMENTS AND DESIGN GOALS

Category	Subcategory	Description
Functional Requirements	Object Detection	Detect objects using Aruco markers, color and shape detection, and depth cameras.
	Precise Manipulation	Perform fine tasks using customized fingertips for the EE.
	Adaptability	Handle variability in object poses and environmental conditions.
	Data Flow	Ensure real-time communication between sensors and actuators.
Design Goals	Modularity	Enable integration of new hardware/software; flexible board inspired by NIST.
	Scalability	Support upgrades for more complex tasks and advanced sensors.
	Precision	Ensure consistent task execution and adaptability to object poses.
	Tool Integration	Seamlessly integrate with ROS, Aruco, and motion planning tools.
Constraints	Joint Limits	Avoid configurations exceeding robotic arm's joint mobility.
	Hardware	Constrained by sensor resolution and tool design/-materials.
	Environment	Stable lighting required for color detection; reduce sensor noise.

Table 4.1: Summary of System Requirements and Design Goals

based on the cable's detected pose on the board ensures the system can adapt dynamically to different initial conditions and execute tasks with high accuracy.

- **Integration with Existing Tools:** Ensure compatibility with the ROS framework and leverage existing tools, such as Aruco libraries and dynamic motion planning algorithms, to streamline development and reduce system complexity. This approach ensures seamless integration of vision-based detection and motion control.

### 4.2.3 Constraints

The system design is subject to the following constraints:

- **Joint Mobility Limitations of the Robotic Arm:** The robotic arm has specific joint limits that restrict certain rotations and configurations. This prevents the end effector (EE) from operating effectively when positioned too far from or too close to the robot's base frame. These limitations must be carefully addressed during task planning to avoid configurations leading to kinematic singularities or reduced precision. This includes analyzing workspace reachability and avoiding redundant postures that can lead to instability or inefficiency.
- **Hardware Specifications:** The system is constrained by the capabilities of its hardware components, such as the depth camera resolution and the design of the tools used on the board. The tools depend on the material used during 3D printing and their design and functionality. These factors can influence the system's ability to execute tasks efficiently and require careful consideration during development and testing.

- **Environmental Factors:** The system must function reliably under variable lighting conditions and potential sensor noise. In this project, the need for color detection was particularly sensitive to lighting, requiring consistent illumination throughout operations to ensure accuracy. Addressing these factors involved stabilizing the lighting conditions in the experimental setup and calibrating sensors to minimize noise and enhance robustness.

## 4.3 Integration of Hardware Components

This section describes the integration of the key hardware elements, focusing on their configuration and synchronization to ensure seamless operation.

### 4.3.1 Integration of the Robotic Arm and Sensors

The robotic arm, Franka Emika Panda (FEP), serves as the main manipulator in the system. The integration process included:

- **Camera Mounting and Calibration:** An Intel RealSense D435i depth camera was mounted on the robotic arm in an *Eye-on-Hand* configuration. This setup allows the camera to move with the end effector, providing real-time visual feedback for object detection and tracking. A detailed calibration process was conducted to align the camera's coordinate system with the base frame of the robotic arm. This ensures accurate spatial referencing for manipulation tasks and prevents misalignment errors during operation.
- **End Effector Configuration:** A customized end effector (EE) with interchangeable fingertips was designed and attached to the arm, providing enhanced grip and adaptability for handling both rigid and deformable objects. The fingertips were optimized for manipulating Ethernet cables and connectors with high precision.
- **Communication Interface:** The robotic arm and sensors communicate through a ROS-based interface, which allows real-time data exchange and ensures synchronized operation between sensing and actuation components.

### 4.3.2 RealSense and Aruco Synchronization

The Intel RealSense D435i depth camera plays a crucial role in providing real-time visual feedback for object detection and manipulation. To achieve precise spatial referencing, the camera's functionality was enhanced through the use of Aruco markers. The key aspects of synchronization include:

- **Aruco Marker Placement and Detection:** Aruco markers were strategically placed on the assembly board and the cable connectors to serve as reliable reference points. These markers were detected in real-time using ROS-compatible libraries, enabling the system to calculate their positions and orientations relative to the camera's coordinate frame.
- **Calibration for Coordinate Alignment:** A precise calibration process was performed to align the camera's coordinate frame with the base frame of the robotic arm. Using a hand-eye calibration method, the system established a unified spatial reference, ensuring that visual data could be accurately translated into the robot's workspace.

- **Depth Data Processing and Object Pose Estimation:** Depth data from the RealSense camera was combined with the marker detection to estimate the 3D poses of objects. This included calculating the position and orientation of deformable objects, such as cables, and dynamically updating these values to support real-time manipulation.
- **Real-Time Integration and Synchronization:** The integration between the camera and the robotic arm was implemented through a ROS-based framework, enabling synchronized updates between visual data and robotic motion. This synchronization ensured that changes in the workspace, such as object displacement, were promptly reflected in the robot's action plans.

#### 4.3.3 Custom Assembly Board and Object Manipulation

The custom assembly board was designed to provide a modular and adaptable workspace for robotic manipulation tasks. Its integration focused on maximizing flexibility and stability:

- **Modular Design:** The board features a perforated matrix that allows for multiple configurations, enabling the placement of objects and tools in various positions. This modularity supports testing a wide range of manipulation scenarios.
- **Aruco Markers for Spatial Alignment:** Four Aruco markers were affixed to the board's corners to define its position relative to the robot's workspace. This alignment facilitates consistent object placement and repeatability in experiments.
- **Stability and Anti-Slip Mechanism:** The board is supported by anti-slip rubber feet, ensuring stability during high-precision tasks. This prevents unintended movement that could affect the robot's performance.
- **Adaptability for Deformable Objects:** Specific modifications were made to support deformable object manipulation. For example, blue tape was added to the Ethernet cable to improve visibility for the camera, and additional supports were designed to stabilize the cable during manipulation.

#### 4.3.4 Challenges in Hardware Integration

Several challenges were encountered during the integration of hardware components, including:

- **Camera Calibration Accuracy:** Achieving precise alignment between the camera and the robotic arm required iterative calibration procedures and validation using test objects. This ensured consistency in pose estimation during experiments.
- **Workspace Optimization:** Configuring the assembly board to maximize accessibility while respecting the joint limits of the robotic arm required careful placement and testing to avoid configurations that could lead to reduced performance or collisions. Different positions and orientations of the board relative to the robot's base were tested to ensure the robot could operate freely without restrictions.
- **Design Challenges for End Effector Fingertips:** Designing customized fingertips for the end effector (EE) posed several challenges due to the need to interact with both rigid and flexible components:



- **Dual Interaction Requirements:** The fingertips needed to handle rigid objects, such as connectors, and flexible components like Ethernet cables. The design had to accommodate these diverse requirements while ensuring precise manipulation.
- **Grip Stability for Flexible Components:** Ensuring a secure grip on the flexible cable was particularly challenging. The fingertips were designed with grooves and a textured surface to prevent slippage and allow for precise control without damaging the cable.
- **Size and Compatibility Constraints:** The fingertips had to be compact enough to handle small components while maintaining sufficient contact area for stability. Multiple iterations were conducted to optimize the dimensions and functionality.
- **Integration with the End Effector:** The fingertips were designed to attach seamlessly to the EE, ensuring quick replacements and modularity for future adjustments or tasks.

## 4.4 Software Architecture and Communication

The software architecture of the robotic system was designed to ensure robust communication and seamless integration of hardware components. The system leverages the Robot Operating System (ROS) as middleware, providing modularity, scalability, and real-time communication between sensors, actuators, and control algorithms. This section describes the main components of the software architecture and their role in enabling autonomous manipulation.

### 4.4.1 ROS Node Management

The implemented ROS architecture of the robotic manipulation system is organized into multiple nodes, each responsible for specific tasks. These nodes interact through ROS topics, services, and actions to ensure seamless integration between sensing, control, and motion execution. Figure B.1 in the Appendix B illustrates the entire ROS node graph used in this project.

The main nodes utilized in the system are detailed below:

1. **gazebo\_gui:** This node is part of the Gazebo simulator and provides a graphical interface for monitoring and interacting with the simulated environment. It is essential for testing the system in a virtual setup before deployment on real hardware.
2. **fr3\_gripper\_spawner:** This node handles the initialization and management of the robotic gripper. It ensures proper synchronization between the robot's end effector and the gripper, allowing for precise control during manipulation tasks.
3. **motion\_AS:** This Action Server node manages the execution of planned trajectories based on the DHB model. It communicates with Behavior Tree (BT) nodes to execute high-level tasks such as grasping, placing, and manipulating objects.
4. **camera\_link\_broadcaster:** This node broadcasts the transformations (TFs) between the camera's reference frame and the robot's base frame. These transformations ensure spatial consistency for visual data and facilitate accurate object localization and trajectory planning.
5. **arucos\_detection:** Dedicated to detecting Aruco markers, this node identifies the position and orientation of objects and tools on the modular assembly board. The detected data is used for object tracking and manipulation.

6. **trajectory\_cartesian\_impedance\_controller/follow\_cartesian\_trajectory:** This node implements Cartesian impedance control to execute planned trajectories while maintaining compliance and safety during interactions with the environment.
7. **camera/realsense2\_camera\_manager:** This node manages the Intel RealSense depth camera, handling the acquisition of RGB and depth images. The visual data is used for object detection, pose estimation, and workspace mapping.
8. **joint\_state\_publisher:** This node publishes the states of the robot's joints, including angles, velocities, and effort, providing real-time feedback for both simulation and real hardware.
9. **robot\_state\_publisher:** This node publishes the robot's state, including the forward kinematics information, based on the joint states. It ensures that the system has an updated representation of the robot's pose in the workspace.
10. **franka\_state\_controller:** This node handles the Franka Emika Panda's state management, publishing information such as the robot's pose, joint states, and measured forces. It acts as a critical interface between the hardware and the software system.
11. **trajectory\_cartesian\_impedance\_controller/cartesian\_trajectory\_goal:** A client node responsible for sending Cartesian trajectory goals to the impedance controller, ensuring smooth and precise motion execution.
12. **tf\_static:** This node publishes static transformations between the various reference frames in the system, ensuring a consistent spatial reference during operation.

**Interaction and Data Flow:** Each node interacts through predefined ROS topics, services, or actions, as visualized in the node graph. For example:

- `/camera/realsense2_camera_manager` publish data on topics like `/camera/depth/image_color`, which are subscribed to by `/arucos_detection` for object detection and localization, as the cable connector.
- The `/motion_AS` node communicates with Behavior Tree nodes to execute high-level manipulation tasks.
- Transformation nodes like `/camera_link_broadcaster` ensure consistency across reference frames, enabling seamless integration between vision, planning, and execution layers.

**Visual Representation:** To better understand the interactions between these nodes, the full ROS node graph is included in Appendix B. The graph provides a comprehensive overview of all nodes, topics, and their interconnections, showcasing the complexity and modularity of the system.

By structuring the ROS architecture in this manner, the system achieves robust communication, scalability, and adaptability, ensuring its effectiveness in real-world applications.



## Chapter 5

# Experimental Method

### 5.1 Introduction

This chapter describes the comprehensive approach adopted to evaluate the capabilities of the proposed robotic manipulation system in diverse scenarios. This involves a structured methodology encompassing the setup, execution, and analysis phases to systematically validate the system's performance.

The experiments focus on assessing the system's ability to autonomously manipulate both rigid and deformable objects, specifically targeting assembly and disassembly tasks. Key aspects of this evaluation include the calibration of the workspace, recording and processing of demonstration trajectories, and execution of predefined tasks using an adaptive robotic framework.

In this chapter, the experimental setup is detailed, highlighting the physical configuration of the workspace and test scenarios designed to challenge the robot's adaptability and precision. Following this, the procedure for trajectory recording and processing is outlined, utilizing kinesthetic demonstrations and invariant transformations to ensure robust trajectory planning and execution. Finally, the execution phase is elaborated, including task-specific strategies, system initialization, and error handling, with an emphasis on collecting and analyzing key performance metrics.

### 5.2 Experimental Setup

The experimental setup was designed to evaluate the manipulation capability of the robotic system and the feasibility of successful assembly and disassembly tasks. This section describes the physical configuration of the workspace and the different test scenarios developed for the evaluation phase.

#### 5.2.1 Configuration of the Environment

This section describes the physical setup and configuration of the environment in which the robotic manipulation experiments were conducted. The organization and use of the hardware components and supplementary tools, previously listed in Table 3.1, are detailed below.

The experiments were conducted in a controlled environment to minimise external variables, such as lighting, and ensure repeatability.

The physical configuration of the experimental setup consists of two main subsystems: the "assembly board system" and the "robotic arm system". These are composed of various elements.

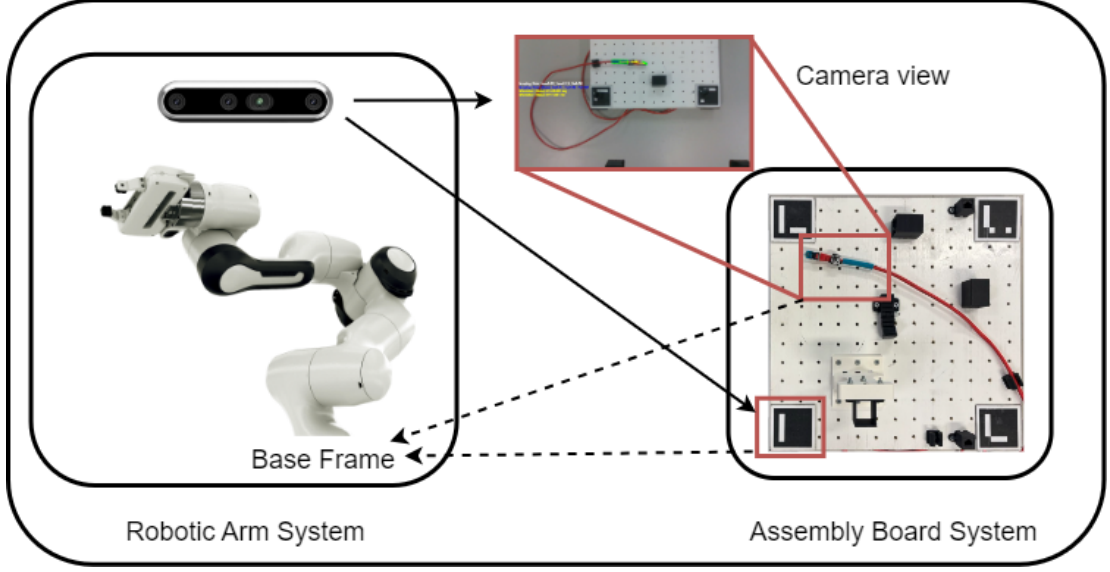


Figure 5.1: Experimental setup showing the robotic arm system (left), camera view (top center), and the assembly board system with cable manipulation configuration (right).

**Assembly Board System** The assembly board is positioned at the center of the workspace, mounted on four supports equipped with anti-slip rubber bearings to ensure stability, and fitted with four Aruco markers to define its position in space relative to the base frame of the robotic arm.

On top of the board, 3D-printed elements and the Ethernet cable, with which the robotic arm will interact, are placed according to a specific configuration.

### Robotic Arm System

The FEP is positioned next to the board, with the ability to reach all areas to perform manipulation tasks. The camera is mounted, using a dedicated support, directly above the EE frame, in an Eye-on-Hand configuration. This setup provides precise visual feedback of the first subsystem.

The environment is calibrated to align the coordinate systems of the camera, the board, the connector, and the robotic arm, creating a unified system in which spatial coordinates are consistent across all components.

### 5.2.2 Test Scenarios

During the experimental phase, two main test scenarios were developed and analyzed to evaluate the capabilities of the robotic system. These scenarios focus on specific tasks and the flexibility of the experimental setup.

#### 1. Clip Task:

- *Objective:* Validate the manipulation procedure for one of the tools on the board, referred to as the "clip," which required a complex sequence of operations, including opening and closing the tool and inserting the Ethernet cable into it.

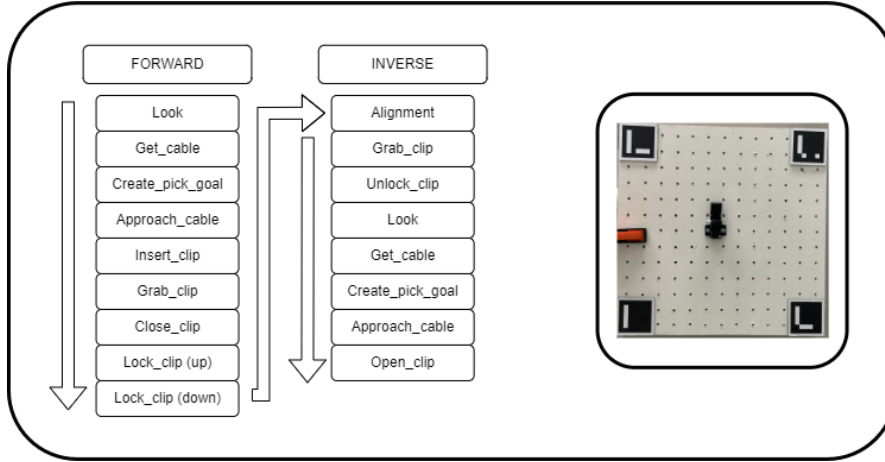


Figure 5.2: Top view of the board with the configuration and all the steps used for the Clip Task.

- *Setup:* The robot had to precisely execute the following steps:
  - (a) Perform an initial trajectory called "Look" to obtain a complete view of the board.
  - (b) Detect the clip on the board using Aruco markers placed on the board.
  - (c) Detect the position and orientation of the cable relative to the robot's base frame through color and shape detection.
  - (d) Approach and grasp the connector.
  - (e) Insert the Ethernet cable into the open clip.
  - (f) Close the clip.
  - (g) Start the clip-opening procedure using a series of trajectories.
  - (h) Perform the "Look" trajectory again to verify the correct pose of the connector once the tool was unlocked, completing the operation.

This scenario was chosen to test the system's ability to perform delicate and precise manipulations on objects with movable parts.

- *Adopted Configuration:* Figure 5.2 shows a top view of the board with the configuration used for this scenario.

## 2. Complete Path Based on NIST Setup:

- *Objective:* Attempt to complete a full path inspired by the setups proposed by NIST, which included the use of multiple tools on the board.
- *Setup:* The path consisted of sequential tasks simulating a complete route on one of the NIST boards, starting from an initial point and guiding the cable to the endpoint through the tools positioned along the path. Although the test was not completed, the initial operations demonstrated the system's adaptability in handling different tools.
- *Adopted Configuration:* Figure 5.3 shows a top view of the board with the configuration used for the NIST-inspired path.

## 3. Flexibility Test of the Board and Object Placement:

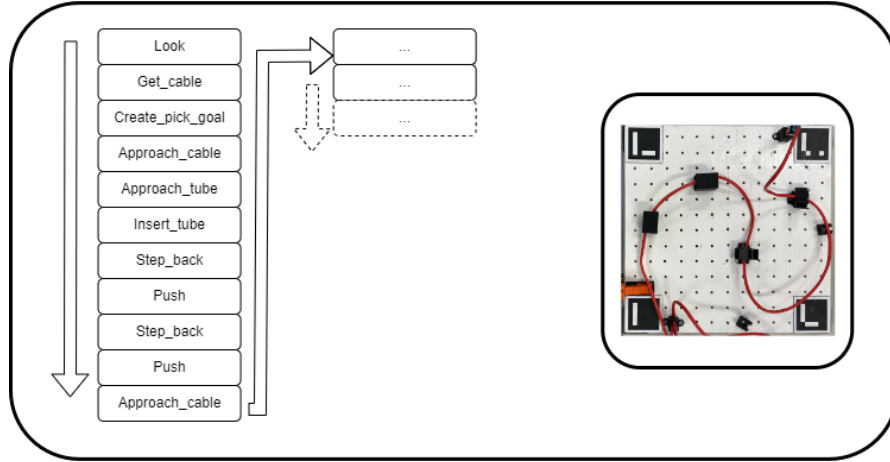


Figure 5.3: Top view of the board with the configuration and all the steps used for the first part of the Complete Path based on the NIST Setup.

- *Objective:* Test the flexibility of the manipulation procedure by repositioning objects on the board and adopting different configurations, including changing the position and orientation of the board relative to the robot's base frame. This latter modification was also useful for finding a configuration that avoided limitations caused by the robot arm's joint constraints.
- *Setup:* The objects and tools on the board were repositioned multiple times to explore:
  - The robot's ability to adapt to different configurations.
  - The possibility of avoiding limitations caused by the robot's joint constraints, ensuring smooth and collision-free movements.

This scenario allowed for the evaluation of the effectiveness of calculated and generalized trajectories and the robustness of the system calibration.

## 5.3 Experimental Procedure

The experimental procedure was designed to systematically evaluate the performance of the robotic system in the defined test scenarios. This section describes the approach followed during the experiments to ensure consistency and repeatability, starting with the trajectory recording phase and ending with the execution of the target task.

### 5.3.1 Preparation Phase

The first phase consists of preparatory steps to ensure the system was calibrated, remained consistent during the subsequent phases, and ready for operation:

- **Workspace Preparation:** The board and all tools were arranged according to the test scenario, with the positioning of objects and tools adjusted to align with the system's field of view and reachability, following the pattern to be tested.
- **Calibration:**

1. The camera coordinate system was aligned with the robot's base frame using hand-eye calibration techniques.
  2. **Aruco markers were detected and employed to define the reference frame of the board, ensuring consistent spatial alignment with the robot base. However, these markers were not utilized in the final tests; instead, rigid objects, such as the 3D-printed components on the board, were directly defined through their coordinates in the code.**
- **Test Environment Control:** Lighting conditions and external variables were stabilized to minimize noise in visual data. Additionally, the board was secured to prevent movement during interactions with the robot or the user.

### 5.3.2 Trajectory Recording and Processing Phase

The trajectory recording phase employed the Kinesthetic Demonstration technique, described in Chapter 2, *Theoretical Foundations*, subsection 2.2.1. In this approach, the robot was physically guided by a human operator to follow the desired trajectory. This method allowed the necessary data to be directly acquired from the robot's system, recording the sequence in a ROS bag file.

This phase was divided into the following sub-phases:

- **Trajectory Acquisition:**
  - The robot was manually guided to execute the desired movements using the *Kinesthetic Demonstration* technique.
  - During this procedure, data related to the pose, linear velocity, angular velocity, force, moment, and time of the end effector (EE) were recorded in ROS bag files.
  - The recording included key topics related to the robot's state (`/franka_state_controller/0.T_EE` and `/franka_state_controller/0.T_EE.vel`), ensuring all necessary information for subsequent processing was captured.
- **Filtering and Pre-Processing:**
  - Acquired data for linear and angular velocities were filtered using a moving average window to reduce noise and enhance stability.
  - Temporal data were normalized to ensure uniform time alignment, starting from the defined starting point (*start point*).
  - The filtered trajectories were saved in **npz** format, enabling more efficient and flexible management during subsequent processing stages.
- **Transformation into Invariant Space:**
  - The trajectories were processed using the Denavit-Hartenberg-inspired bidirectional model (DHB), transforming them into an invariant space.
  - This transformation allowed the trajectories to be represented independently of rotations, translations, and temporal variations, providing a robust and generalizable representation.
  - The invariant variables ( $I$ ) and initial parameters ( $H_v0$  and  $H_w0$ ) were computed using the `computeDHB` function.
- **Trajectory Generalization and Reconstruction:**



- Trajectories were generalized to adapt to new task requirements, such as varying start and goal positions. Using the `trajectory_generalization` function, the recorded trajectory was scaled and rotated to align with new conditions, maintaining the overall shape and structure of the original trajectory.
- This step proved particularly useful during cable detection scenarios, where the position and orientation of the cable varied. The trajectory was adapted dynamically, ensuring precise alignment of the robot’s movements with the cable’s detected pose.
- For orientation generalization, quaternions were converted into Euler angles to apply scaling factors. The generalized orientations were then reconstructed into quaternions to maintain compatibility with robotic motion planners.
- Reconstructed trajectories were verified to ensure consistency with the generalized goal and the original trajectory data.

### Data Visualization and Analysis

During the processing phase, visualizations were generated to analyze the trajectories and compare the original data with the generalized and reconstructed results:

- Filtered linear and angular velocities were compared to their reconstructed counterparts to evaluate the accuracy of the DHB model and generalization process.
- Calculated invariant variables were graphed to observe their stability and behavior during the trajectories.
- Generalized positions and orientations were visualized to confirm alignment with the new start and goal conditions.

### Saving Results

The final results, including invariant variables, initial parameters ( $H_v0, H_w0$ ), positions, orientations, and generalized trajectories, were saved in `npz` format for future use, ensuring quick access during the execution phase.

### 5.3.3 Execution Phase

Each experiment followed a structured approach to ensure systematic evaluation. The entire process involved simultaneously running two main files: a *launch file* to initiate the ROS nodes and Action Server, and a Python script to execute the Behavior Tree (BT) specific to the task.

1. **System Initialization:** The system was configured and started using a *launch file*, which performed the following tasks:
  - Configured the robot for use in simulation or on real hardware, based on the `use_sim` parameter.
  - Launched the *Action Server* (`dhb.action_server.py`), responsible for executing the trajectories requested by the BT.
  - Activated the detection pipeline through the `detection_pipeline.launch` file to identify objects and markers in the scene.
  - (Optional) Recorded experimental data in bag files if the `record` parameter was set to `true`.

2. **Launching the Behavior Tree:** In parallel, the BT specific to the task was launched (e.g., `Clip_seq` for executing the "clip" task, as introduced in subsection 5.2.2). This BT orchestrated the sequence of actions needed to complete the task, using ROS nodes to interact with the robot and gripper.
3. **Trajectory Initialization:** Before executing the task, the robot performed a preliminary trajectory called *Look* to scan the board and collect initial data. This trajectory was a prerequisite for every test. During this phase, the system:
  - Identified the position and orientation of tools using Aruco markers.
  - Located the Ethernet cable using color and shape detection.
4. **Task Execution:** The BT orchestrated the sequence of actions to complete the task, interacting with the Action Server to execute the trajectories. During this phase, the operator monitored the robot (in case of real hardware execution) and the scene viewed from the EE-mounted camera, as issues or errors in trajectory generation could arise.
5. **Data Collection:** During execution, key metrics were recorded, such as:
  - Execution time for each task.
  - Positional accuracy during manipulations.
  - Success rate of operations (e.g., correct clip closure), noted manually by testing the task multiple times.
6. **Error Management:** In case of errors, such as missed detections or collisions, the system or the user:
  - Stopped the current execution.
  - Provided feedback to adjust parameters or the setup.
  - Restarted the task to repeat the operation.

#### 5.3.4 Post-Experiment Analysis

Once the experiments were completed, the data collected in each scenario were analyzed to evaluate the system's performance:

- **Trajectory Review:** Recorded trajectories were analyzed for smoothness, stability, and adaptability to changes.
- **Error Analysis:** Failure points were examined to identify possible improvements in object detection, trajectory planning, or execution.

## 5.4 Results and Observations

### 5.4.1 Validation of Individual Modules

#### Perception

The perception module, while not the primary focus of this thesis, played a fundamental role in enabling the detection and localization of the cable connector for subsequent manipulation tasks. Following a careful analysis of the problem and the various options available in the literature,

a custom implementation was developed, combining color and shape detection, and validated through iterative testing. While this approach does not introduce innovative methodologies compared to state-of-the-art solutions discussed in Chapter 1, it was tailored to meet the practical requirements of the project, offering a compromise between simplicity, effectiveness, and robustness.

**Results with Color and Shape Detection** The implemented perception pipeline, based on HSV color segmentation and shape analysis, demonstrated sufficient stability and reliability across all test scenarios. The key observations from the iterative tests are summarized as follows:

- **Detection Stability:** The system consistently detected the connector’s position and orientation with high stability, even in scenarios involving partial occlusions (up to 30%-40% of the connector’s area, provided the end of the connector remained visible) or when the camera approached the connector at very close distances (less than 30 cm).
- **Real-Time Performance:** The detection algorithm operated in real-time, providing continuous feedback without significant delays, which was essential for ensuring smooth integration with the robotic manipulation framework. It should be noted that the connector was not moved during the execution phase, as the project’s goal did not involve handling highly dynamic conditions where objects on the board are moved in real-time.
- **Variability Handling:** The pipeline effectively managed variability in the connector’s position, orientation, and partial occlusions, making it suitable for the dynamic nature of the experimental environment.
- **Practical Effectiveness:** Although not innovative, the approach proved sufficient to reliably detect the connector under controlled conditions, meeting the specific requirements of the project without the need for additional hardware or complex computational resources.

A key strength of the approach was its resilience to common error modes in perception tasks. For example:

- Detection was unaffected by slight variations in lighting conditions, as the color thresholds were iteratively calibrated to ensure robustness.
- Proximity effects, such as when the camera was too close to the connector, did not result in positional loss or instability, as the system was programmed to retain the last detected position of the connector before losing detection.

**Comparison and Discussion** While the implemented solution lacks the complexity of advanced methods in the literature, such as deep learning-based object detection or hybrid approaches combining color, shape, and feature tracking, it aligns well with the project objectives. Specifically:

- Advanced methods, while offering greater generalization and accuracy, often require extensive training data, high computational resources, and prolonged setup times, which were not feasible given the project constraints.
- The chosen implementation provided a lightweight, efficient, and sufficient solution, tailored for detecting a single object (the Ethernet cable connector) in a controlled experimental environment.

### DHB-Based Goal Estimation

The DHB-based framework is at the core of this project, as the approach presented in the paper "A Bidirectional Invariant Representation of Motion for Gesture Recognition and Reproduction" [20][21], has been fundamental in ensuring accurate and robust goal estimation for various recorded and reconstructed trajectories required for task execution. By transforming the trajectories into an invariant space, the model delivered consistent performance across varying initial conditions and spatial configurations. This section presents the results of the DHB-based goal estimation process, highlighting its precision, generalization capability, and robustness in different scenarios.

**Positional and Orientational Accuracy** The DHB model achieved high positional and orientational accuracy in estimating target poses. The main findings include:

- **Positional Accuracy:** The deviation between estimated and actual goal positions consistently remained within  $\pm 2$  mm, demonstrating reliable precision for cable manipulation tasks.
- **Orientational Accuracy:** Orientation errors, measured in Euler angles, remained below  $\pm 1^\circ$ , ensuring the connector was correctly aligned for grasping and insertion operations.

It should be noted that these uncertainties arise during the trajectory generalization phase, not from the application of the DHB approach. After reconstructing trajectories from the invariant space, the DHB model exhibited negligible error.

**Comparison Between Original and Generalized Trajectories** Trajectory generalization occurs after reconstructing the original trajectories from the invariant space using the DHB model. This process allows trajectories to adapt to new start and goal configurations while preserving the overall motion structure. The adaptation is achieved through simple scaling or extension of the original trajectory, ensuring the shape remains consistent with the original kinematic and dynamic characteristics.

- **Shape Preservation:** The generalization, based on scaling and rotations of the original trajectory, preserved the overall shape of the movement. This result highlights the DHB model's ability to adapt trajectories to new contexts without altering the fundamental motion characteristics.
- **Adaptability to Variations:** The algorithm demonstrated high flexibility, successfully adapting trajectories to various start and goal configurations. Specifically, it managed cases involving system rotations, scaling variations in trajectory length, and significant changes in initial and final positions.

A crucial aspect of the process was maintaining stability and robustness even with significant adaptations. Generalization ensured the resulting trajectory conformed to the new geometric constraints without introducing instability or unwanted alterations. However, in situations close to the robot's kinematic singularities, slight oscillations were observed in the generalized trajectory, suggesting a need for further refinement of the transformation parameters. For this reason, the board's position was occasionally adjusted, as it was either too close to or too far from the robot's base.

### 5.4.2 Framework Performance in Task Execution

The framework evaluation was conducted by examining the system’s ability to complete various tasks inspired by the benchmarks proposed by NIST. This section analyzes the successes achieved, the limitations encountered, and possible solutions to overcome the obstacles identified during testing.

**Clip Task: Complete Success** This task involved inserting the cable into the clip, closing it, opening it, and subsequently removing the cable. It was successfully completed in all stages:

- The system accurately handled cable insertion into the clip, maintaining stability even during contact.
- The clip’s closure was achieved without errors, thanks to well-defined trajectories and proper calibration of the perception system.
- The inverse process, i.e., opening the clip and removing the cable, was also completed without issues, demonstrating the framework’s robustness for relatively simple and well-structured tasks.

**Complete Path: Task Incomplete** This task required passing the cable through a complex path involving multiple tools. It could not be completed due to several limitations:

- **Challenges with Cable Nature:** The cable’s deformability introduced instability during positioning, with unpredictable tensions making precise control difficult.
- **Kinematic Limits of the Robotic Arm:** Despite attempts to reposition the board and objects, the required configurations often caused the arm to reach its mechanical limits, preventing smooth execution of certain trajectories.
- **Complexity of Required Movements:** The need to perform complex movements, such as inserting and removing the cable from multiple clips with varying orientations, exceeded the current system’s capabilities.

#### Improvement Proposals

**Configuration with Two Robotic Arms** A configuration with two robotic arms could significantly enhance the system’s capabilities:

- **Coordinated Manipulation:** Two robotic arms would enable coordinated cable manipulation, offering better tension management and greater stability during tasks.
- **Increased Flexibility:** The use of two arms would reduce the need for reaching kinematically unfavorable configurations.

**Integration of an Advanced Perception System** The adoption of an advanced perception system could improve the understanding of the cable’s body:

- **3D Models of the Cable’s Body:** Using real-time 3D reconstruction techniques would allow monitoring the entire cable configuration.
- **Additional Sensors:** Integrating tactile sensors along the robotic arm’s body could provide more detailed feedback during manipulation.

**Final Observations**

While the Clip Task demonstrated the framework’s effectiveness in simpler tasks, the failure of the Complete Path highlighted the need for further improvements. Specifically:

- The complexity of tasks based on NIST benchmarks requires a combination of hardware solutions (e.g., multi-arm configurations) and software solutions (e.g., advanced perception algorithms).
- The developed framework provides a solid foundation for further developments, delivering promising results in robotic manipulation tasks with less complex requirements.



# Conclusions

This research focused on designing and validating a framework for robotic manipulation capable of addressing complex assembly tasks involving both rigid and deformable objects. At the core of the project was the DHB-based invariant representation model, which enabled the transformation of motion trajectories into a generalizable space, ensuring robustness and consistency even under varying conditions. This methodology allowed for the planning of precise and flexible trajectories, adaptable to different configurations without compromising the integrity of the original movements.

A critical component was the perception module, developed to identify and localize the cable connector used in the tests. Despite being based on a relatively simple approach that combined color and shape detection, the system proved reliable enough to operate in controlled conditions. While it did not introduce innovations compared to advanced methods in the literature, the adopted solution fully met the practical needs of the project, ensuring stability even in challenging scenarios such as partial occlusions or close-up views of the connector.

The experiments highlighted the potential of the developed framework while also revealing some intrinsic limitations. The system demonstrated reliability and robustness in handling structured tasks with relatively simple requirements, confirming the effectiveness of the chosen approach. However, significant challenges arose in scenarios demanding greater flexibility, such as managing deformable objects or performing particularly complex movements. These challenges emphasized the need to explore alternative solutions, such as more advanced hardware configurations or sophisticated perception systems, to successfully address more intricate and dynamic scenarios.

The DHB model proved to be a promising foundation for robotic manipulation applications, thanks to its ability to adapt trajectories to new conditions without the need for complex recalibrations. Nonetheless, improvements are required to handle more dynamic and demanding scenarios. Potential enhancements include integrating tactile sensors or employing real-time 3D reconstruction techniques, which could significantly improve the system's ability to manage deformable objects and complex movements.

In conclusion, this work represents a significant step toward developing robotic manipulation systems that are modular, adaptable, and capable of balancing simplicity and efficiency. While some limitations emerged, the results provide a solid foundation for further developments and refinements. This research not only broadens the technical possibilities of robotic systems but also underscores the value of practical and well-considered solutions to address real-world challenges. It is a forward-looking contribution, laying the groundwork for a new generation of robots that are more versatile and aware of the complexities of the environments in which they operate.





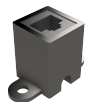

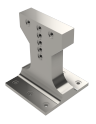
## CONCLUSIONS

---



## Appendix A

# 3D-Printed Tools for the Custom Assembly Board

Tool Name	Description and Technical Specifications	Image
Aruco Base	A square plate designed to hold Aruco markers, positioned at the corners of the board.	-
Cable Holder	A cylindrical block with a V-shaped opening to secure the cable in place, facilitating easy insertion.	
Clip	A flexible tool with a hinge mechanism that allows the cable to be inserted and securely fixed inside.	
Ethernet Housing	A socket designed to hold the Ethernet connector securely during manipulation.	
Tube	A hollow parallelepiped structure through which the cable is inserted on one side and exits on the other.	
T-Bracket	A complex tower-like structure with a vertical tube mounted on top, allowing the cable to be inserted from above and exit downward.	

## Appendix B

# ROS Node Graph

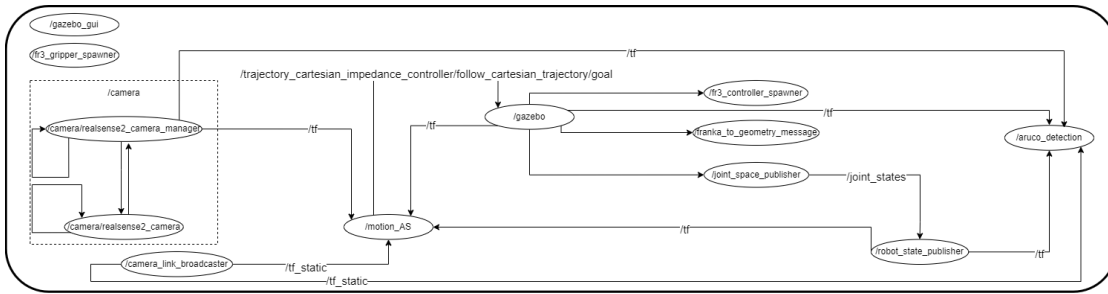


Figure B.1: Complete ROS node graph used in the project. The graph illustrates the interactions and data flow between perception, control, and communication nodes.



# Bibliography

- [1] Fares Abu-Dakka et al. “Dynamic Movement Primitives in Robotics: A Tutorial Survey”. In: *arXiv preprint arXiv:2102.03861* (2021). URL: <https://arxiv.org/abs/2102.03861>.
- [2] Paul Bakker and Yasuo Kuniyoshi. “Robot See, Robot Do : An Overview of Robot Imitation”. In: *AISB96 Workshop on Learning in Robots and Animals* (May 1996).
- [3] Aude Billard and Danica Kragic. “Trends and Challenges in Robot Manipulation”. In: *Science* 364.6446 (2019), eaat8414. DOI: [10.1126/science.aat8414](https://doi.org/10.1126/science.aat8414).
- [4] Júlia Borràs et al. “The KIT Swiss Knife Gripper for Disassembly Tasks: A Multi-Functional Gripper for Bimanual Manipulation with a Single Arm”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 4590–4597. DOI: [10.1109/IROS.2018.8593567](https://doi.org/10.1109/IROS.2018.8593567).
- [5] Roger Bostelman and Joseph Falco. *Survey of Industrial Manipulation Technologies for Autonomous Assembly Applications*. en. 2012-03-16 2012. DOI: <https://doi.org/10.6028/NIST.IR.7844>.
- [6] Alessio Caporali, Kevin Galassi, and Gianluca Palli. “Deformable Linear Objects 3D Shape Estimation and Tracking From Multiple 2D Views”. In: *IEEE Robotics and Automation Letters* 8.6 (2023), pp. 3852–3859. DOI: [10.1109/LRA.2023.3273518](https://doi.org/10.1109/LRA.2023.3273518).
- [7] Alessio Caporali et al. “A Weakly Supervised Semi-Automatic Image Labeling Approach for Deformable Linear Objects”. In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 1013–1020. DOI: [10.1109/LRA.2023.3234799](https://doi.org/10.1109/LRA.2023.3234799).
- [8] Alessio Caporali et al. “Deformable Linear Objects Manipulation With Online Model Parameters Estimation”. In: *IEEE Robotics and Automation Letters* 9.3 (2024), pp. 2598–2605. DOI: [10.1109/LRA.2024.3357310](https://doi.org/10.1109/LRA.2024.3357310).
- [9] Alessio Caporali et al. “RT-DLO: Real-Time Deformable Linear Objects Instance Segmentation”. In: *IEEE Transactions on Industrial Informatics* 19.11 (2023), pp. 11333–11342. DOI: [10.1109/TII.2023.3245641](https://doi.org/10.1109/TII.2023.3245641).
- [10] Peng Chang and Taskin Padi. *Model-Based Manipulation of Linear Flexible Objects with Visual Curvature Feedback*. 2020. arXiv: [2007.08083](https://arxiv.org/abs/2007.08083) [cs.R0]. URL: <https://arxiv.org/abs/2007.08083>.
- [11] K Chen, W Zhang, et al. “Robotic peg-in-hole assembly based on reversible dynamic movement primitives and trajectory optimization”. In: *IEEE Robotics and Automation Letters* 5 (2020), pp. 500–507. DOI: [10.1109/LRA.2020.2975767](https://doi.org/10.1109/LRA.2020.2975767).
- [12] Andrew Choi et al. “mBEST: Realtime Deformable Linear Object Detection Through Minimal Bending Energy Skeleton Pixel Traversals”. In: *IEEE Robotics and Automation Letters* 8.8 (Aug. 2023), pp. 4863–4870. ISSN: 2377-3774. DOI: [10.1109/lra.2023.3290419](https://doi.org/10.1109/lra.2023.3290419). URL: <http://dx.doi.org/10.1109/LRA.2023.3290419>.

- [13] Michele Colledanchise and Petter Ögren. “Behavior Trees in Robotics: A Survey”. In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1230–1240. DOI: [10.1109/TR0.2018.2853726](https://doi.org/10.1109/TR0.2018.2853726).
- [14] Michele Colledanchise and Petter Ögren. “Behavior Trees in Robotics: A Survey”. In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1230–1240. DOI: [10.1109/TR0.2018.2853726](https://doi.org/10.1109/TR0.2018.2853726).
- [15] Ravinder Dahiya and Maurizio Valle. “Tactile Sensing for Robotic Applications”. In: *Sensors* 13.7 (2013), pp. 9182–9206. DOI: [10.3390/s130709182](https://doi.org/10.3390/s130709182).
- [16] Kevin Galassi, Alessio Caporali, and Gianluca Palli. “Cable Detection and Manipulation for DLO-in-Hole Assembly Tasks”. In: *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*. 2022, pp. 01–06. DOI: [10.1109/ICPS51978.2022.9817006](https://doi.org/10.1109/ICPS51978.2022.9817006).
- [17] Seth Hutchinson, Greg Hager, and Peter Corke. “A Tutorial on Visual Servo Control”. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 651–670. DOI: [10.1109/70.538972](https://doi.org/10.1109/70.538972).
- [18] Shiyu Jin et al. “Robotic Cable Routing with Spatial Representation”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5687–5694. DOI: [10.1109/LRA.2022.3158377](https://doi.org/10.1109/LRA.2022.3158377).
- [19] Azarakhsh Keipour et al. *Detection and Physical Interaction with Deformable Linear Objects*. 2023. arXiv: [2205.08041](https://arxiv.org/abs/2205.08041) [[cs.R0](https://arxiv.org/abs/2205.08041)].
- [20] Dongheui Lee, Raffaele Soloperto, and Matteo Saveriano. “Bidirectional Invariant Representation of Rigid Body Motions and its Application to Gesture Recognition and Reproduction”. In: *Autonomous Robots* 42 (Jan. 2018), pp. 1–21. DOI: [10.1007/s10514-017-9645-x](https://doi.org/10.1007/s10514-017-9645-x).
- [21] Dongheui Lee, Raffaele Soloperto, and Matteo Saveriano. “Bidirectional Invariant Representation of Rigid Body Motions and its Application to Gesture Recognition and Reproduction”. In: *Autonomous Robots* 42 (Jan. 2018), pp. 1–21. DOI: [10.1007/s10514-017-9645-x](https://doi.org/10.1007/s10514-017-9645-x).
- [22] Y. Li, X. Huang, et al. “Towards Reversible Dynamic Movement Primitives”. In: *Journal of Advanced Robotics* (2021), pp. 237–246. DOI: [10.1109/JAR.2021.2976765](https://doi.org/10.1109/JAR.2021.2976765).
- [23] Naijing Lv, Jianhua Liu, and Yunyi Jia. “Dynamic Modeling and Control of Deformable Linear Objects for Single-Arm and Dual-Arm Robot Manipulations”. In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2341–2353. DOI: [10.1109/TR0.2021.3139838](https://doi.org/10.1109/TR0.2021.3139838).
- [24] Julian Michels, Michele Colledanchise, and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CreateSpace Independent Publishing Platform, 2018.
- [25] Julian Michels et al. *Behavior Trees in Robotics and AI: An Introduction*. CreateSpace Independent Publishing Platform, 2018.
- [26] Manfred Morari and Jay H. Lee. “Model Predictive Control: Past, Present and Future”. In: *Computers & Chemical Engineering* 23.4–5 (1999), pp. 667–682. DOI: [10.1016/S0098-1354\(98\)00301-9](https://doi.org/10.1016/S0098-1354(98)00301-9).
- [27] National Institute of Standards and Technology. *Robotic Grasping and Manipulation Assembly - Assembly Tasks Board*. <https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly>. Accessed: 2024-10-03. 2023.
- [28] Antonis Paraschos et al. “A Reversible Dynamic Movement Primitive Formulation”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014), pp. 1507–1515.

- 
- [29] Lawrence R. Rabiner. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. DOI: [10.1109/5.18626](https://doi.org/10.1109/5.18626).
- [30] Cen Rao, Mubarak Shah, and Tanveer Syeda-Mahmood. “Action Recognition based on View Invariant Spatio-temporal Analysis”. In: *Computer Vision Lab, University of Central Florida* (2003).
- [31] Cen Rao, Alper Yilmaz, and Mubarak Shah. “View-Invariant Representation and Recognition of Actions”. In: *International Journal of Computer Vision* 50.2 (2002), pp. 203–226.
- [32] Harish chaandar Ravichandar et al. “Recent Advances in Robot Learning from Demonstration”. In: *Annu. Rev. Control. Robotics Auton. Syst.* 3 (2020), pp. 297–330. URL: <https://api.semanticscholar.org/CorpusID:208958394>.
- [33] Stefan Schaal, Auke Jan Ijspeert, and Aude Billard. “Dynamical movement primitives: Learning attractor models for motor behaviors”. In: *Neural computation* 25.2 (2003), pp. 328–373.
- [34] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005. ISBN: 978-0262201629.
- [35] Maxim Vochten, Tinne De Laet, and Joris De Schutter. “Robust Optimization-Based Calculation of Invariant Trajectory Representations for Point and Rigid-body Motion”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 5598–5605. DOI: [10.1109/IROS.2018.8593540](https://doi.org/10.1109/IROS.2018.8593540).
- [36] Isaac Weiss. *Geometric Invariants and Object Recognition*. Tech. rep. CS-TR-2942. Sponsored by Office of Naval Research, Defense Advanced Research Projects Agency. Center for Automation Research, University of Maryland, 1992.
- [37] Yuxuan Yang, Johannes A. Stork, and Todor Stoyanov. “Learning differentiable dynamics models for shape control of deformable linear objects”. In: *Robotics and Autonomous Systems* 158 (2022), p. 104258. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2022.104258>. URL: <https://www.sciencedirect.com/science/article/pii/S0921889022001518>.
- [38] Mingrui Yu et al. “Global Model Learning for Large Deformation Control of Elastic Deformable Linear Objects: An Efficient and Adaptive Approach”. In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 417–436. DOI: [10.1109/TR0.2022.3200546](https://doi.org/10.1109/TR0.2022.3200546).
- [39] Zuyuan Zhu and Huosheng Hu. “Robot Learning from Demonstration in Robotic Assembly: A Survey”. In: *Robotics* 7.2 (2018). ISSN: 2218-6581. DOI: [10.3390/robotics7020017](https://doi.org/10.3390/robotics7020017). URL: <https://www.mdpi.com/2218-6581/7/2/17>.



## BIBLIOGRAPHY

---

# List of Figures

2.1	.....	10
2.2	The three main phases in imitation .....	10
3.1	Modular assembly board configurations: base setup (left), various tool arrangements (center), and cable manipulation scenario (right). .....	30
4.1	System design illustrating the teaching, encoding, and reproducing phases for trajectory learning and execution using invariant representations and Cartesian impedance control. ....	36
5.1	Experimental setup showing the robotic arm system (left), camera view (top center), and the assembly board system with cable manipulation configuration (right). ..	44
5.2	Top view of the board with the configuration and all the steps used for the Clip Task. ....	45
5.3	Top view of the board with the configuration and all the steps used for the fist part of the Complete Path based on the NIST Setup. ....	46
B.1	Complete ROS node graph used in the project. The graph illustrates the interactions and data flow between perception, control, and communication nodes. ....	59



# List of Tables

2.1	Comparison of learning-from-demonstration methods . . . . .	11
2.2	Behavior Tree node types and their respective outcomes. . . . .	27
3.1	List of Materials and Components Used in Experiments . . . . .	31
3.2	Technical Specifications of the Custom Board . . . . .	31
3.3	Technical Specifications of the Franka Emika Panda Robotic Arm . . . . .	32
3.4	Technical Specifications of the Intel RealSense D435i Camera . . . . .	33
4.1	Summary of System Requirements and Design Goals . . . . .	37

