Physics-Informed AI Methods for Deformable Object Manipulation

by

Omey M. Manyar

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(MECHANICAL ENGINEERING)

May 2025

To my parents, Mr. Mohan Manyar and Mrs. Ankita Manyar, my late Grandmother, Mrs. Ratnaprabha Desai, and my life partner, Dr. Jesal Shah.

Acknowledgements

I owe my deepest gratitude to my advisor, Dr. Satyandra K. Gupta, whose mentorship has shaped my research and my personal growth. His generous guidance through countless discussions, careful feedback, and unwavering support created an environment where I could pursue ambitious ideas with confidence. Working in the USC Center for Advanced Manufacturing under his leadership has been the highlight of my doctoral journey. I'd also like to thank my dissertation committee, Dr. Mitul Luhar, Dr. Quan Nguyen, and Dr. Stefanos Nikolaidis, for their expert insights and thoughtful critiques, which have significantly strengthened this work. I also appreciate the financial support of USC Viterbi (Graduate Fellowship) that helped me explore several interesting problems, the National Science Foundation, and Amazon Robotics, all of which made my research possible. A special note of appreciation to Shantanu Thakar, who guided me to be successful in the professional aspects of my career. His guidance helped me secure multiple internship experiences that were pivotal in my academic journey and that opened new avenues in my work. Early in my PhD, Ariyan Kabir, Brual Shah, and Aniruddha Shembekar offered crucial advice and encouragement that helped me learn new things quickly. I've been fortunate to share this journey with outstanding peers in my lab—Zachary McNulty, Rishi Malhan, Alec Kanyuck, Hantao Ye, Rutvik Patel, Jeon Ho Kang, Sarah Alhussaini, Yeo Jung Yoon, Rishabh Shukla, and Abhay Negi—whose collaboration and camaraderie made every challenge more rewarding. Apart from my peers, a lot of my friends who were also my collaborators helped me through this – Neel Dhanaraj, Jaineel Desai, Shahwaz Khan, Akshay Deshmukh, and many more. To my family: my father, Mohan, and my mother, Ankita, whose guidance and encouragement has been crucial in achieving this;

my late grandmother, Mrs. Ratnaprabha Desai, for her enduring love; and my partner, Dr. Jesal Shah, whose steadfast support and belief in me have carried me through every obstacle—thank you. Your faith in my work has been my greatest source of inspiration.

Table of Contents

List of Tables

List of Figures

Abstract

Deformable objects are an integral part of the world we inhabit, characterized by complex, nonlinear behavior under external forces and constraints. Nearly every object humans interact with exhibits some degree of deformability, and our ability to intuitively model and manipulate such objects has been a cornerstone of human dexterity across daily life and industrial tasks alike. Deformable materials play a crucial role in manufacturing, logistics, and assembly processes that power high-performance sectors such as aerospace, fulfillment, automotive, textiles, etc. Yet, despite their prevalence, many of these processes remain predominantly manual without large-scale integration of intelligent robots, primarily due to the inherent challenges of manipulating deformable objects with the precision, repeatability, and quality demanded in industrial environments.

Amidst a sharp decline in the availability of skilled labor willing to perform such physically demanding tasks, the industry faces an urgent need to deploy intelligent robotic systems that can handle deformable objects with the same resilience and adaptability as humans. This dissertation addresses this challenge by enabling robotic manipulation of complex deformable objects in semi-structured, high-variability environments, with a focus on large-scale, industrially relevant objects exhibiting intricate dynamics and high-speed interactions. Moving beyond simplified 1D and 2D cases, this work aims to tackle the nuanced realities of a new class of complex industrial deformable objects that demand precision, safety, and resilience—attributes essential for large-scale robot adoption in manufacturing, assembly, and logistics.

A fundamental advance proposed in this dissertation is the introduction of a physics-informed learning paradigm for robotic manipulation of deformable objects. The central

perspective is that purely data-driven approaches struggle to generalize in high-mix, low-volume (HMLV) industrial settings where exhaustive data collection is impractical. In contrast, integrating structured physics knowledge into learning pipelines significantly improves generalization, reduces data requirements, and enhances model interpretability and safety. To this end, the dissertation introduces physics-informed methods for: (i) learning simulation parameters for large deformable sheets, (ii) sequencing complex manufacturing tasks by capturing expert human preferences, (iii) planning manipulation actions using simulation-based grasp planning for precision draping, (iv) modeling the effect of tool compliance on manipulation under uncertainty, and (v) detecting failures in deformable object processes using simulation-augmented deep learning.

The framework is structured around four core pillars: (i) Simulation-based Learning, (ii) Advanced Learning Frameworks, (iii) Learning Physics-informed Constraints, and (iv) Compositional Learning. Together, these ingredients form a systematic strategy for endowing robots with the ability to reason about the physical consequences of their actions, enabling safe, adaptive, and reliable operation in high-performance industrial tasks involving deformable objects.

Through extensive real-world experiments—including composite sheet layup, precision screwdriving under uncertainty, online defect detection in manufacturing, and bimanual bin-packing of deformable packages- this dissertation demonstrates that physics-informed learning provides a critical pathway toward safer, more robust, and scalable robotic systems for manipulating deformable objects. By marrying the flexibility of learning with the structure of physics principles, this work takes a significant step toward the widespread deployment of intelligent robots capable of manipulating complex deformable objects in unstructured, real-world industrial environments.

Chapter 1

Introduction

"The behavior of soft materials resists simplification – not because they defy physics, but because they embody so much of it."

— Inspired by the work of Nobel Laureate Pierre-Gilles De Gennes

1.1 Background

For times immemorial, scientists and engineers have been captivated by the mechanics of deformable objects. Deformability, often associated with compliance or softness, is not a niche phenomenon but a defining characteristic of the physical world we inhabit. Most of the objects we interact with daily possess some degree of deformability. The impact of deformable objects on human evolution is so profound that many anthropologists and scientists credit the dexterity of our soft, compliant hands as a key enabler of our success as a species – allowing us to manipulate and craft complex tools and artifacts with remarkable precision.

Deformable objects are becoming increasingly prevalent in industrial settings—a shift driven by rising consumer demand for product customization and the widespread adoption of high-performance materials such as composites. These materials lie at the heart of several billion-dollar industries, including aerospace,

automotive, logistics, and advanced manufacturing. From forming critical structural components using large, flexible composite sheets to packaging and transporting goods in soft polybags, deformable objects play an essential role in enabling high-performance, scalable production workflows. Even the tools used in many of these processes exhibit compliance, significantly influencing the execution of tasks such as screwdriving, fastening, and precision assembly.

Today, as the field of robotics stands on the cusp of a new era, one defined by widespread deployment in unstructured, real-world environments, the ability to perceive, predict, and manipulate deformable objects has emerged as a critical frontier. Addressing this challenge is not merely academic—it is essential for automation in sectors that underpin global supply chains and modern infrastructure. To equip robots with manipulation capabilities approaching human versatility, we must develop methods that marry physical principles with intelligent, data-driven control, enabling robust and explainable interaction with these complex materials.

1.2 Motivation

(1) Composite Layup (2) Fulfillment (3) Maintenance

Figure 1.1: Industrial tasks involving manipulation of deformable objects.

Deformable object manipulation is a fundamentally challenging problem due to the complex, high-dimensional nature of soft materials. Unlike rigid bodies, which retain a consistent shape and can be represented with simple geometric

models, deformable objects exhibit continuous shape changes under external forces and constraints. Their behavior is influenced by a combination of geometry, material properties, and contact conditions, making it difficult to define coherent state representations or predict future states. Moreover, deformable objects often vary in size, shape, and material composition, which adds another layer of variability to the problem.

Humans, however, are remarkably adept at manipulating deformable objects. Through our multimodal sensing (vision, touch, proprioception), predictive modeling of physical dynamics, and rapid adaptation to unexpected changes, we are able to perform complex tasks involving soft materials with ease—be it folding laundry, assembling wiring harnesses, or packing irregular items. Our resilience to failure and capacity for reactive behavior are critical enablers of this proficiency. This human proficiency has historically been a cornerstone of skilled labor in manufacturing, enabling tasks that require dexterity, adaptability, and nuanced control. Our ability to manipulate deformable objects has played a critical role in daily life and also underpinned the progress of the Industrial Revolution itself.

However, the landscape of modern industry is rapidly evolving. The demand for greater customization, combined with the shrinking availability of skilled labor for physically demanding or tedious tasks, is driving a shift toward deploying robots in less structured, more variable environments. Industrial settings are increasingly characterized by high-mix, low-volume (HMLV) production, where the rigid automation strategies of the past no longer suffice. To complicate matters further, many of the tasks that remain for human workers involve manipulating deformable components such as compliant part assembly, deformable tool manipulation, fabrics, polybags, and composite materials (Refer Fig. 1.1). These objects are inherently variable and difficult to model, yet must be handled with high precision and care. Automating such tasks is

not only a matter of productivity but also of safety and sustainability in a workforce-constrained future. It is, therefore, imperative that robots gain the ability to understand, manipulate, and adapt to deformable objects with the same fluency as humans—this is the challenge that is directly addressed in this dissertation.

When it comes to industrial environments, the stakes are significantly higher. Standards for quality, precision, and safety are stringent, and any deployed robotic or automation solution must meet these requirements reliably. Although robots have proven their value in structured, repetitive tasks—especially in automotive and electronics assembly—the next frontier lies in enabling them to handle unstructured deformable object tasks with similar fluency. This transition demands more than just better models or more data—it calls for a new way of thinking about robot learning and reasoning.

This dissertation is motivated by the central question: How can we enable robots to manipulate deformable objects with the same robustness, adaptability, and efficiency as humans, especially in safety-critical, variable industrial environments? The answer explored here lies in combining physics-based modeling with data-driven learning to form a new class of intelligent, explainable, and resilient robotic systems.

1.3 Research Issues

Traditional industrial environments, such as automotive assembly lines, have long been considered structured settings. Standardization, along with significant investment in fixtures, conveyors, and jigs, was used to reduce variability and uncertainty. In such environments, robots could be deployed at scale without requiring intelligence in perception or decision-making. Instead, they executed pre-programmed motions with high speed and reliability. However,

this paradigm breaks down when we move toward the manipulation of deformable objects. Pre-scripted motion plans are insufficient for these tasks, which require dynamic adaptation based on the object's ever-changing state. To operate reliably, robots must sense using multiple modalities, reason about object behavior, predict deformations accurately, and recover from failures. These are capabilities that current automation pipelines are ill-equipped to handle.

The problem of deformable object manipulation is not new to robotics. Prior research has explored simplified instances involving 1D and 2D objects such as ropes, cloths, and elastic beams. These studies often make strong assumptions to reduce problem complexity—assumptions that break down in real-world, high-precision applications. Tasks demonstrated in domestic settings (e.g., folding laundry or placing garments in a basket) have limited demands on accuracy, speed, or safety. In contrast, many industrial tasks—such as manipulating compliant tools for assembly, packing polybag packages, or placing composite laminates—require millimeter-level precision, failure resilience, and compliance with strict safety standards. The manipulation of large or shell-like deformable objects, particularly those encountered in manufacturing and logistics, remains underexplored.

Recent progress in machine learning has enabled more capable robotic systems. Learning-based models can extract informative representations from high-dimensional sensor data and predict the nonlinear dynamics of deformable materials. However, these methods come with significant limitations. Their performance is tightly bound to the data they are trained on—the inductive biases encoded via datasets, architectures, and downstream tasks. As a result, they often fail to generalize to new object geometries, materials, or unseen task conditions. This is particularly problematic in industrial contexts, where

high-mix, low-volume (HMLV) production scenarios preclude large-scale data collection for every variation.

Even in high-volume production settings, collecting real-world data that adequately captures the full variability of deformable parts across shape, material properties, initial conditions, and environmental interactions—is both prohibitively expensive and time-consuming. The complexity and diversity of deformable object behavior make it nearly impossible to build comprehensive datasets that cover all relevant edge cases. As a result, purely data-driven models, which often rely on large-scale, labeled datasets to generalize effectively, struggle when deployed in the wild.

Compounding this issue is the nature of most deep-learning models themselves. Despite their impressive performance in controlled environments, these models typically operate as black boxes, offering little insight into why a prediction was made or how the system might behave under unfamiliar conditions. This lack of interpretability poses serious challenges in industrial contexts, where safety, traceability, and regulatory compliance are crucial. These concerns have created a significant barrier to the adoption of modern learning-based techniques beyond a lab setting in real-world industrial deformable object manipulation.

In contrast, humans are able to learn robust models of the world's physical behavior from limited data. We leverage a deep, intuitive understanding of physics to reason about object behavior and adapt our actions accordingly. Physics-informed machine learning (PIML) aims to bring this style of reasoning into modern AI. By embedding physical priors and governing constraints into data-driven models, PIML methods enhance generalization, reduce data requirements, and improve interpretability. This dissertation argues that physics-informed learning provides a path forward for enabling robust,

safe, and scalable deformable object manipulation in industrial environments. By integrating structured physics knowledge with the flexibility of learning, these methods offer the best of both worlds: improved efficiency in learning, better generalization to new conditions, and greater alignment with industrial safety and reliability standards. This hybrid approach forms the cornerstone of the contributions presented in this dissertation.

1.4 Objectives and Scope

The overarching objective of this dissertation is to advance the deployment of robotic systems capable of manipulating complex deformable objects in unstructured and high-stakes industrial environments. To this end, the dissertation proposes a unified, physics-informed framework that integrates the strengths of model-based reasoning with the adaptability of data-driven learning. This framework aims to bridge the gap between theoretical modeling and practical deployment by addressing five key research objectives:

- *Learning Physics-Informed Dynamics Models:* Develop methods to learn interpretable and generalizable models of deformable object dynamics. This includes building hybrid simulation pipelines that combine analytical physics with learned components and identifying material or interaction parameters that govern deformation behavior. The goal is to achieve data-efficient learning while maintaining physical plausibility and explainability.
- *Learning Physics-Informed Manipulation Strategies:* Utilize physics-based models to inform the development of manipulation policies that are aware of object deformation and task constraints. The objective is to enable robots to perform high-precision manipulation in scenarios where traditional rigid-body assumptions no longer hold, such as manipulating

deformable tools, large composite sheets, or handling soft packaging materials.

- *Conditioning Task Planning on Human Preferences:* Integrate expert decision-making into planning for deformable object manipulation. By learning from demonstrations or preference signals, the goal is to replicate human-like reasoning that accounts for nuanced trade-offs between efficiency, robustness, and safety, which are critical in high-mix, low-volume production scenarios.
- *Learning Physics-Based Defect Detection Models:* Investigate simulation-driven methods for detecting anomalies and failure modes during manipulation. By generating synthetic data using high-fidelity simulators and learning explainable models, the aim is to equip robotic systems with the ability to detect and respond to defects in real time, ensuring safe and resilient operation.
- *Studying Complex Industrial Deformable Objects:* Unlike much of the prior work focused on simple 1D or 2D deformable objects (e.g., ropes or small cloth patches), this dissertation focuses on complex, shell-like, and large-scale deformable objects commonly encountered in industrial applications. These include materials such as polybags, composite sheets, and screwdriving, each presenting unique challenges in terms of modeling, sensing, and manipulation.

By addressing these objectives, this work aims to push the boundaries of robotic manipulation, offering a roadmap for scalable, reliable, and explainable deployment of deformable object manipulation systems in real-world industrial settings.

Chapter 2

Foundations and Overview

2.1 Taxonomy of Deformable Objects

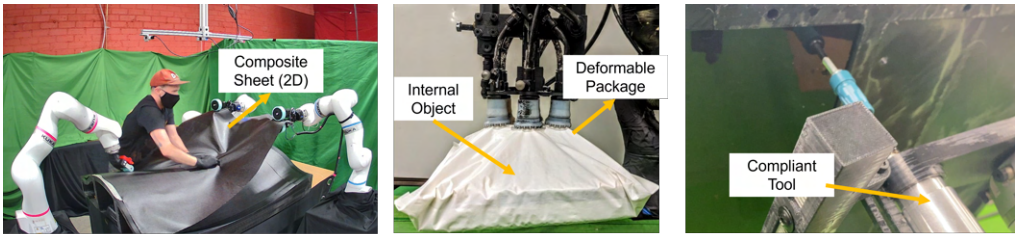


Figure 2.1: The three representative classes of deformable objects studied in this dissertation. Each class presents distinct challenges across modeling, task planning, manipulation, and failure detection, highlighting the need for tailored, physics-informed strategies to enable effective robotic manipulation.

A central contribution of this dissertation lies in redefining the types of deformable objects considered in robotic manipulation, especially in the context of real-world industrial applications. The complexity arises from the objects' continuously changing geometry, their sensitivity to contact and material properties, and the difficulty of modeling or predicting their dynamic behavior under manipulation. While Chapter 1, Section. 1.4 outlined the key technological components required to enable such manipulation—simulation and

modeling, task planning, manipulation planning, and failure detection—this chapter shifts focus toward another central theme of this dissertation: the classification and analysis of the deformable objects themselves. Unlike prior studies that focus on ropes or garments in relatively constrained settings, this work expands the scope to deformable objects encountered in high-precision industrial environments, where material scale, function, and failure tolerance demand a more rigorous and informed approach.

This dissertation investigates three distinct classes of deformable objects, each portraying a different axis of complexity and industrial relevance. As shown in Fig. 2.1, these object categories are representative of real-world challenges that push the boundaries of current robotic manipulation methods. First, we explore large deformable sheets, such as composite prepreg materials, which pose significant challenges due to their scale, anisotropic mechanical properties, and the high precision required in layup operations. Second, we focus on shell-like deformable objects, such as packages containing internal contents with independent dynamics. These 3D structures combine external elasticity with internal variability, creating a compounded modeling and planning challenge. Finally, we consider tool compliance, where the deformability of the end-effector itself—particularly in tasks like screwdriving—introduces nonlinear interaction dynamics that must be explicitly modeled to ensure accurate control and robust execution.

Across all three categories, this dissertation demonstrates how traditional methods—often designed for small-scale or low-variability deformable objects, fail to generalize to these more complex cases. In contrast, physics-informed learning emerges as a unifying framework that enables improved generalization, reliability, and safety by embedding physical structure into data-driven

models. This perspective guides the design of algorithms and systems throughout the dissertation, providing a robust foundation for manipulating complex deformable objects.

2.2 Physics-Informed Learning Paradigm

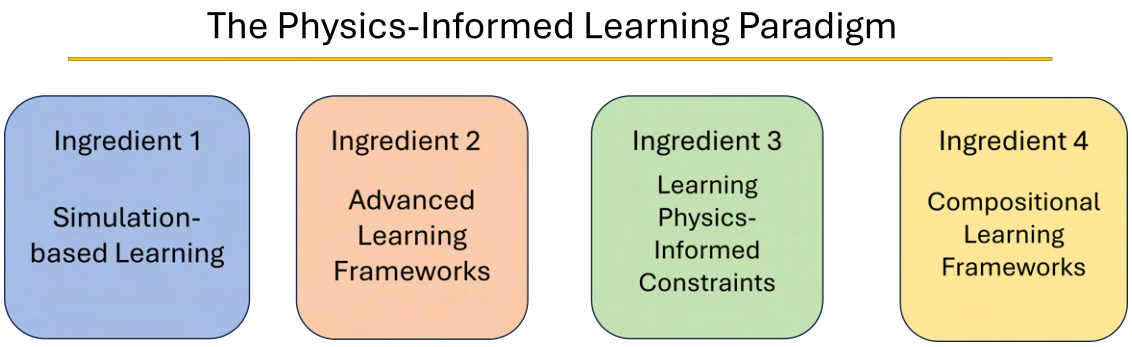


Figure 2.2: The four key physics-informed learning paradigms explored in this dissertation. These ingredients—simulation-based learning, advanced learning to enable structured dynamics modeling, physics-guided constraint learning for optimization, and compositional learning—can be applied individually or in combination to enable robots to effectively manipulate the complex deformable objects studied in this work.

The technological components introduced in Chapter 1, such as modeling, task planning, manipulation, and failure detection, have each been extensively studied in the context of deformable object manipulation. However, this dissertation presents a novel perspective on how to systematically embed physics-based knowledge into each of these components, particularly when dealing with the complex, industrially relevant deformable objects studied in this work.

There are several complementary ways in which physics can inform a robot's decision-making process. As illustrated in Fig. 2.2, we identify four key paradigms through which physical priors can be incorporated into the learning framework. These approaches are not mutually exclusive; instead, they can

be applied synergistically to improve learning efficiency, generalization, and interoperability.

1. *Simulation-based learning:* The first paradigm focuses on using simulation and physics-based models to aid in planning and generating high-quality synthetic data. This alleviates the need for costly and time-consuming real-world data collection, enabling the application of data-hungry learning techniques without sacrificing realism or relevance.

2. *Advanced Learning Frameworks:* In this paradigm, physical knowledge is embedded directly into the learning frameworks—either by learning structured representations that capture governing equations or by designing architectures that reflect known physical structure (e.g., the particle-based representation for shell-like packages in Chapter 4). These structured models help constrain the learning space and make the predictions more grounded in real-world physics.

3. *Learning Physics-based Constraints for Optimization via Loss Functions or Representations:* The third paradigm incorporates physical and task-specific constraints directly into the learning objective. By designing physics-informed loss functions or enforcing process constraints during optimization, we introduce inductive biases that guide the learning process toward feasible and safe behaviors.

4. *Compositional and Modular Learning Architectures:* Deformable object manipulation tasks are often too complex to be effectively learned end-to-end. Moreover, monolithic models can hinder interpretability and transferability. Instead, this dissertation advocates for modular architectures, where each component or sub-task is learned separately and infused with

relevant physics-based priors (e.g., the compositional manipulation strategy in Chapter 9). This structured approach improves both explainability and adaptability.

These four paradigms represent a practical yet flexible toolkit for integrating physics into learning. While not exhaustive, they offer a compelling foundation for enabling robust and scalable manipulation of complex deformable objects. In the chapters that follow, we explore each of these methods in detail, demonstrating how physics-informed learning can drive intelligent behavior in real-world robotic systems for manipulating deformable objects.

2.3 Overview and Structure

This dissertation introduces a unified framework for physics-informed learning to enable robotic manipulation of complex deformable objects in industrial environments. This chapter has laid the foundation by formally defining the three key classes of deformable objects investigated in this work and introducing the core principles of physics-informed learning that guide the methodology.

Figure 2.3 presents a conceptual matrix that organizes the contributions of this dissertation along three key axes: (1) Deformable object class (2D vs. 3D), (2) Technological component (e.g., simulation, planning, execution, anomaly detection), and (3) Type of physics-informed ingredient (e.g., simulation-based learning, structured models, loss/constraints, modular learning). This matrix illustrates how each chapter contributes to specific intersections within this conceptual space.

Guided by this structure, the dissertation is implicitly organized into four major parts, each centered on one of the technological components introduced in

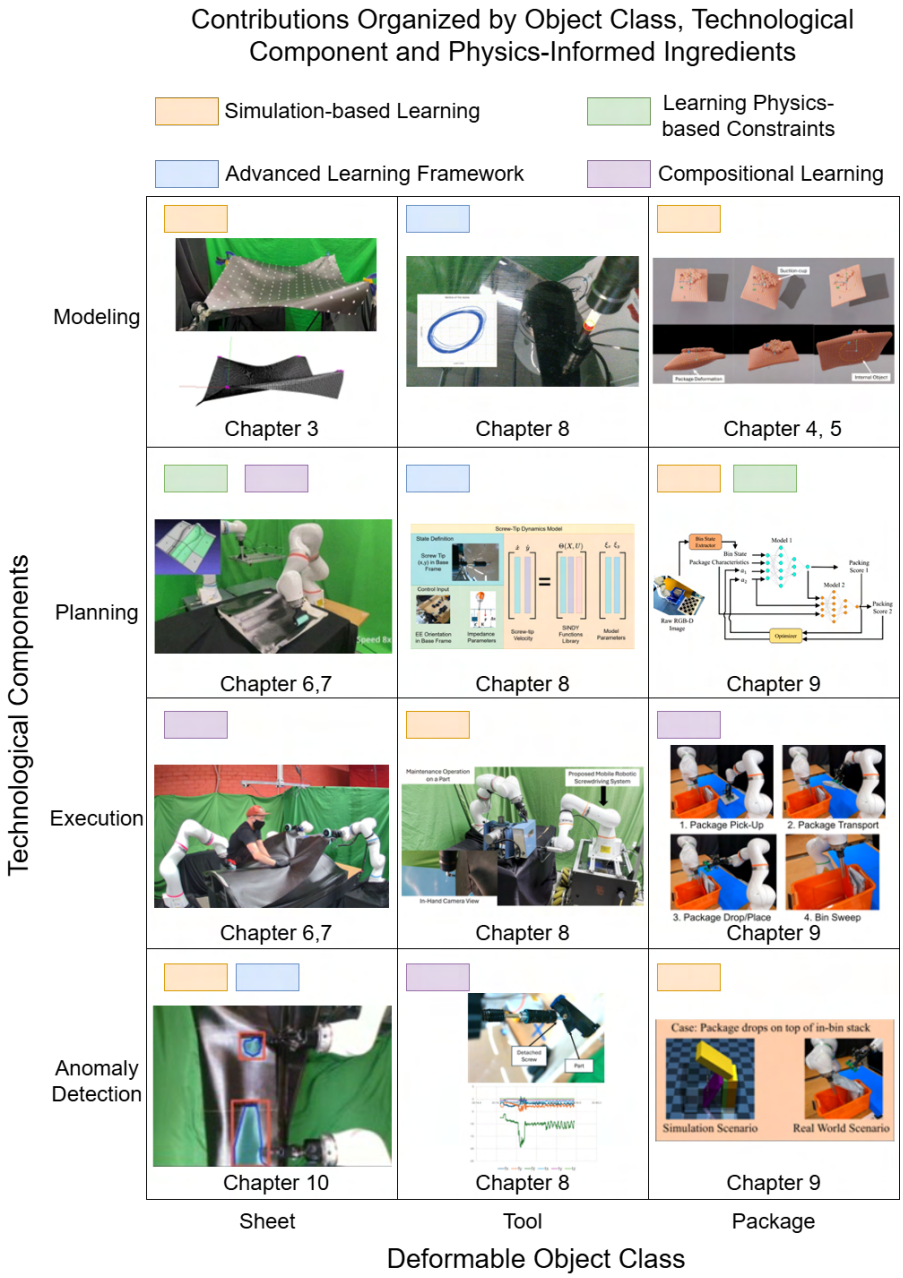


Figure 2.3: A structured overview of the contributions presented in this dissertation, organized across two axes: the class of deformable object (sheets, tools, packages) and the technological components (simulation, planning, execution, anomaly detection). Each marked cell represents a contribution where physics-informed strategies, such as simulation-based learning, advanced learning frameworks, constraint-aware planning, and compositional learning, were employed to address the challenges of manipulating complex deformable objects.

Section 2.1. Each part presents a set of contributions that address the unique modeling, planning, and learning challenges associated with the deformable object classes discussed in Section 2.1.

- *Part I: Simulation and Parameter Learning (Chapters 3, 4, 5, 8)*: This part focuses on structured representations and physical parameter estimation for deformable objects. We propose methods for learning the physical properties of large composite sheets (2D) and internal-object-containing packages (3D). In addition, we introduce a method to learn governing dynamics equations for complex objects, such as compliant tools in screwdriving tasks, resulting in explainable and task-relevant dynamic models.
- *Part II: Task Planning Conditioned on Expert Preferences (Chapter 6)*: Effective deformable object manipulation must be informed by real-world process knowledge. We show how expert demonstrations from industrial workflows can be used to learn task-level constraints and high-level decision-making strategies, enabling robots to perform context-aware and feasible actions.
- *Part III: Physics-guided Manipulation Planning (Chapters 7, 9)*: This part addresses manipulation planning using physics-informed simulation and structured models. We develop planning strategies for bi-manual manipulation of both 2D (composite sheets) and 3D (deformable packages) deformable objects, leveraging learned dynamics models to improve robustness and task success.
- *Part IV: Failure and Anomaly Detection (Chapters 8, 10)*: Robust operation requires the ability to detect and respond to process anomalies. We demonstrate how physics-informed models and synthetic data generation can be used to train interpretable failure detection systems. These include

detecting wrinkles in large sheets and modeling time-based anomalies in screwdriving tasks involving tool compliance.

Furthermore, each of these Chapters has been published or submitted to peer-reviewed forums. Chapter 3 has been published at the ASME Journal of Manufacturing Science and Engineering [1], Chapter 4 has been submitted to the ASME International Conference for Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Chapters 6 [2] and 7 [3] have been published at IEEE International Conference on Robotics and Automation (ICRA), The preliminary work for Chapter 8 has been published at IEEE International Conference on Intelligent Robots and Systems (IROS) [4], while an exhaustive work that was built up on this is submitted to the Robotics and Computer Integrated Manufacturing Journal. Chapter 9 has been published at IEEE IROS [5], while Chapter 10 has been published at the ASME Journal of Computing and Information Science in Engineering (JCISE) [6].

Together, these four parts demonstrate the versatility and effectiveness of physics-informed learning across a broad spectrum of deformable object classes and industrial use cases. By integrating simulation-based reasoning, data-efficient learning, and task-aware planning, this dissertation offers a unified perspective on how intelligent robotic systems can be designed to handle the complexities of deformable object manipulation. The proposed methods pave the way for more explainable, resilient, and generalizable robotic solutions for manipulating deformable objects, ultimately bringing us closer to safe and scalable deployment in real-world industrial environments.

Chapter 3

Learning Simulation Parameters for Large, Sheet-like Deformable Objects

3.1 Introduction

Sheet-like deformable objects are found across nearly every domain, from everyday household items to high-performance industrial materials. They vary widely in size, aspect ratio, and composition – from a small T-shirt to a large bedsheet, yet humans handle them with remarkable skill. Many of these tasks involve complex interactions between material properties, gravity, and contact forces. In aerospace, for instance, the layup of prepreg composite sheets requires carefully draping large, anisotropic materials over contoured molds while avoiding wrinkles and misalignment. Similarly, in the automotive domain, tasks like positioning seat covers or floor mats demand precise adjustments to achieve tight fits around complex geometries. In the textile industry, operations such as hemming require fine-grained control of soft fabrics with high positional accuracy. These examples highlight the intricate nature of sheet manipulation tasks and the dexterity, adaptability, and physical intuition that humans bring to them.

The primary obstacle to robotic adoption for tasks involving large, flexible sheets is their intricate, highly nonlinear dynamics. Industrial processes demand high-level precision, repeatability, and quality, often at production speeds that match or exceed human performance. Reliable robots, therefore, require predictive models that capture sheet behaviour under gravity, contact, and manipulation forces so that safe and efficient strategies can be planned. While rigid-body dynamics are well understood and widely exploited in robotics [7], those methods do not transfer directly to compliant materials. Accurate parameter identification – bending stiffness, membrane tension, friction, damping, and more is essential for building high-fidelity simulations of deformable sheets.

Prepreg composite layup exemplifies both the industrial importance and the modelling challenges. Composites are experiencing double-digit annual growth and underpin multi-billion-dollar sectors such as aerospace, wind energy, and advanced mobility [8]. In the prepreg process, resin-impregnated sheets are hand-placed onto tooling, then compacted ply by ply. Defects such as wrinkles, air gaps, and bridging arise when the sheet slides or buckles, jeopardising structural integrity [9, 10]. Existing automated solutions – Automated Fiber Placement (AFP) and Automated Tape Layup (ATL) handle only simple geometries; complex parts still rely on skilled labour, leading to variability, rework, and high cost. Automation will be viable only when a robot can predict sheet behaviour in advance and adjust its actions on-the-fly, because once a prepreg ply is draped, it is difficult to reposition without damage. Thus, if we want robots to handle such a complex task, a high-fidelity simulation of such multi-material sheets is important.

To address these challenges, this chapter presents a data-efficient, physics-informed framework for learning the simulation parameters of large, sheet-like deformable objects under fixed constraints. The proposed approach leverages

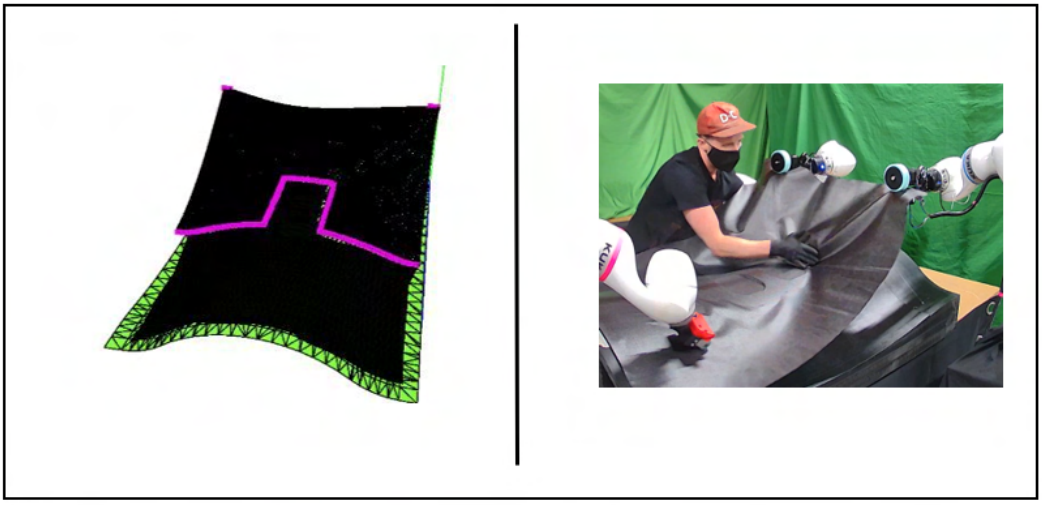


Figure 3.1: Left: the simulated prepreg under external forces and constraints. Right: the current robotic cell with two Kuka iiwa R7 robots and one Kuka iiwa R14 robot.

high-fidelity thin-shell simulation and real-world observations to estimate key material properties with minimal experimental effort. By incorporating prior deformation knowledge through a thin-sheet finite element formulation and using the VegaFEM [11] library for simulation, we construct models that are both accurate and physically interpretable.

The learned simulation parameters are subsequently used to construct a digital twin represented by a force, damping, and mass matrix that can emulate prepreg sheet behavior under external fixed constraints. The study then focuses on model evaluation and testing for different conditions. A detailed comparison of the parameter model predictions and experimental data is also presented. The proposed system enables real-time prediction of sheet dynamics and supports the development of autonomous robotic cells for tasks such as prepreg layup. As illustrated in Fig. 3.1, the learned parameters are integrated into a simulation loop that supports planning, validation, and feedback

control. We demonstrate how this approach generalizes across different materials, such as cotton, felt, and canvas, and discuss its potential in enabling safe, high-throughput manipulation of deformable objects.

3.2 Related Work

Mechanical simulation of composite prepreg sheets has been widely explored, particularly for predicting draping behavior over complex molds. These simulations typically represent the sheet using a mesh and aim to capture fiber alignment, shear, and bending characteristics. One of the simplest and computationally efficient approaches is kinematic simulation, which models the interaction based solely on mold geometry [12, 13]. While fast, such models often lack accuracy in predicting internal strain or out-of-plane deformation.

To achieve greater fidelity, elasticity-based simulations have been developed that compute internal strain distributions within the fabric [14]. Finite Element Analysis (FEA) models further extend this capability by explicitly simulating the deformation mechanics of cloth and composite sheets [15]. For instance, in [16], a robot was used to place flexible material on a doubly curved mold, where FEA models were employed to assess material conformity. Although FEA approaches offer high accuracy, they tend to be computationally expensive, limiting their use in iterative or real-time applications. In this chapter, a new variant of FEA developed in the computer graphics community is adopted, which offers a favorable trade-off between simulation speed, stability, and accuracy [17, 18]. This allows the proposed system to maintain a high level of fidelity while running at interactive rates, thus supporting fast design iteration and in-the-loop parameter optimization.

Comprehensive reviews of fabric simulation techniques are provided in [19], while [20] offers an overview of recent advances in automated composite draping. Several hybrid approaches—known as progressive drape models—combine elements of both kinematic and FEA-based methods to balance accuracy and computational efficiency [21, 22]. Another class of simulation methods is based on particle systems [23–26], which are computationally tractable but often lack the physical accuracy of continuum models due to their discrete nature.

For any of these simulation approaches, accurately tuning the model parameters to reflect real-world behavior is critical. For fabrics, material parameters can be extracted using Kawabata plots [27], which characterize properties like bending and shear under controlled conditions. However, prepreg composite sheets differ from fabrics in that they are significantly stiffer and require higher-precision force and displacement measurements, making techniques like Kawabata plots less practical. Instead, this chapter proposes a data-driven method that uses FEA simulation in conjunction with optical motion tracking to estimate material parameters. The proposed approach avoids direct measurement of internal elastic forces, enabling accurate parameter estimation for composite sheets under realistic constraints.

3.3 Simulation Model Description

This chapter employs a thin-shell finite element method (FEA) to simulate the behavior of viscoelastic prepreg composite materials. The simulation model is based on formulations developed in the computer graphics community [17, 18], and is summarized here for completeness. These models strike a balance between computational efficiency and physical accuracy and, to the best of the author's knowledge, have not been previously applied to real-world composite prepreg materials in industrial contexts.

While simulation fidelity is important, computational speed is also a critical factor, particularly for applications such as material parameter optimization and in-the-loop planning. The method described in this chapter is capable of interactive performance at approximately 10 frames per second, making it well-suited for rapid design iteration and parameter learning. Domain-specific constraints relevant to composite layup, such as surface adhesion and boundary pinning, are incorporated into the simulation framework, along with real-time sheet tracking and optimization routines for aligning simulation output with real-world deformation observations.

The prepreg sheet is modeled as a triangulated mesh, where the displacement of each vertex in 3D space (x, y, z) constitutes a degree of freedom in the simulation. The material's mechanical response is governed by internal forces resulting from bending, shear, and in-plane stretching. These forces are computed based on elasticity theory and serve as the basis for dynamic simulation. To improve computational efficiency, the simulation also includes the analytical computation of Jacobians for both bending and tensile-shear forces, enabling faster convergence during parameter optimization.

3.3.1 Tensile and Shear Forces

This subsection describes the formulation used to model tensile and shear forces within the sheet-like deformable object. The simulation operates on a mesh representation of the sheet, where each triangular element serves as the fundamental unit for computing internal elastic behavior. The surface of the sheet is parameterized using a 2D coordinate space, defined by (u, v) parameters. The corresponding 3D weft and warp vectors, denoted by $\vec{U}$ and $\vec{V}$ respectively, are derived from this parameterization. These vectors represent

local in-plane directions of the material and need not be orthonormal after deformation.

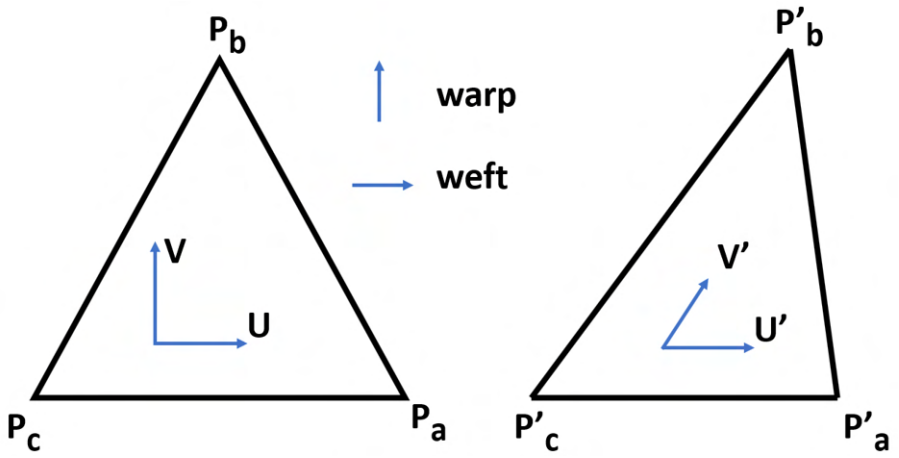


Figure 3.2: Undeformed(left) and Deformed(right) states of a triangle in the mesh representing the sheet. The warp and weft vectors $\vec{V}$ and $\vec{U}$ are used to compute the tensile and shear strains.

To illustrate the computation, consider a single triangle in the mesh in both its undeformed and deformed states, as shown in Fig. 3.2. The 3D vertex positions of the triangle, P_a, P_b, P_c, are mapped from their corresponding (u, v) parameter coordinates: (u_a, v_a), (u_b, v_b), and (u_c, v_c). The weft and warp vectors are represented by weighted sums of the three parametric vertices of the triangle. We can formulate a linear system of six equations: $\sum_i r_{ui} u_i = 1$, $\sum_i r_{ui} v_i = 0$, $\sum_i r_{ui} = 0$, $\sum_i r_{vi} u_i = 0$, $\sum_i r_{vi} v_i = 1$, and $\sum_i r_{vi} = 0$, where $i \in \{a, b, c\}$. The weights r_{ui} and r_{vi} can be precomputed using the equations: $r_{ua} = d^{-1}(v_b - v_c)$, $r_{va} = d^{-1}(u_c - u_b)$, $r_{ub} = d^{-1}(v_c - v_a)$, $r_{vb} = d^{-1}(u_a - u_c)$, $r_{uc} = d^{-1}(v_a - v_b)$, and $r_{vc} = d^{-1}(u_b - u_a)$, where $d = u_a(v_b - v_c) + u_b(v_c - v_a) + u_c(v_a - v_b)$. The system of six linear equations is solved to obtain the vectors $\vec{U}$ and $\vec{V}$ given by the equation (3.1). The viscosity of the material is given by the evolution rates or rate change of these vectors given by the equation (3.2). The vectors are then used to compute the Green-Lagrange

strain tensor, which consists of shear and tensile strains. The rate of change of these strains is then derived. Equations (3.3, 3.4, 3.5, and 3.6) gives the representations.

$$\vec{U} = \sum_{i \in \{a,b,c\}} r_{ui} P_i \quad \vec{V} = \sum_{i \in \{a,b,c\}} r_{vi} P_i \tag{3.1}$$

$$\vec{U'} = \sum_{i \in \{a,b,c\}} r_{ui} P'_i \quad \vec{V'} = \sum_{i \in \{a,b,c\}} r_{vi} P'_i \tag{3.2}$$

$$\epsilon_{uu} = \tfrac{1}{2}(\vec{U}^T \vec{U} - 1) \quad \epsilon'_{uu} = \tfrac{1}{2}(\vec{U}^T \vec{U'}) \tag{3.3}$$

$$\epsilon_{vv} = \tfrac{1}{2}(\vec{V}^T \vec{V} - 1) \quad \epsilon'_{uu} = \tfrac{1}{2}(\vec{V}^T \vec{V'}) \tag{3.4}$$

$$\epsilon_{uv} = \tfrac{1}{2}(\vec{U}^T \vec{V} - \vec{V}^T \vec{U}) \tag{3.5}$$

$$\epsilon'_{uv} = \tfrac{1}{2}(\vec{U}^T \vec{V'} + \vec{V}^T \vec{U'}) \tag{3.6}$$

Deriving the weft, warp, and shear components of total elastic energy of the triangle with respect to vertex position gives us the force applied at the j^{th} vertex of the triangle, given by the equation (3.7).

$$F_j = -\frac{|d|}{2}(\sigma_{uu}(r_{uj}\vec{U}) + \sigma_{vv}(r_{vj}\vec{V}) + \sigma_{uv}(r_{uj}\vec{V} + r_{vj}\vec{U})) \tag{3.7}$$

The stress tensor provides the values of the stresses σ_{ij} used in the equation (3.7). The relationship between stress and strain tensor is given by $\sigma = E\epsilon + E'\epsilon'$ where E and E' are the elastic and viscosity stiffness matrices of the material, and σ and ϵ are the 3D stress and strain vector. Subsequently, the force Jacobian necessary for the implementation and efficiency of numerical techniques are computed. The Jacobian for i^{th} and j^{th} vertex where both $i, j \in \{a, b, c\}$ is computed using the equation (3.8). If viscosity is also considered, then another contribution given by equation (3.9) must also be considered.

$$\frac{\partial F_j}{\partial P_i} = -\frac{|d|}{2}\left(\sum_{m,n\in\{uu,vv,uv\}} \frac{\partial \sigma_m}{\partial \epsilon_n}\left(\frac{\partial \epsilon_m^T}{\partial P_i}\frac{\partial \epsilon_n}{\partial P_j}\right) + \sum_{m,n\in\{uu,vv,uv\}} \sigma_m\left(\frac{\partial}{\partial P_i}\frac{\partial \epsilon_m^T}{\partial P_j}\right)\right) \tag{3.8}$$

$$\frac{\partial F_j}{\partial P_i'} = -\frac{|d|}{2}\left(\sum_{m,n\in\{uu,vv,uv\}} \frac{\partial \sigma_m}{\partial \epsilon_n'}\left(\frac{\partial \epsilon_m^T}{\partial P_i}\frac{\partial \epsilon_n}{\partial P_j}\right)\right) \tag{3.9}$$

The stiffness component governs how the stress-strain relationship affects the forces acting on the triangle's vertices. The new position of the vertices is then found by considering internal and external forces during the simulation. The force acting due to bending stress is also superimposed with the tensile and shear forces to improve simulation accuracy. The following subsection describes how bending forces are computed and incorporated to enhance simulation fidelity further.

3.3.2 Bending Forces

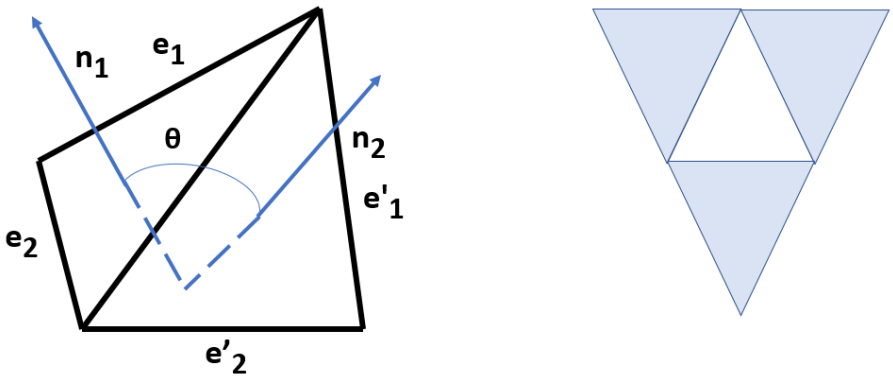


Figure 3.3: (Left) Two adjacent triangles in the mesh and the bending angle θ between them. (Right) Three neighboring triangles for a triangle under consideration are shown.

Bending forces are computed based on the hinge angle between adjacent triangles in the mesh. This chapter adopts the mathematical model introduced in [18], which provides a stable and efficient method for modeling thin-shell bending behavior. As shown in Fig. 3.3, two adjacent triangles in the mesh define a hinge, with their respective normals denoted by n_1 and n_2. The angle θ between these normals quantifies the local bending deformation at the hinge. Consider the total bending energy $E_b = \sum_i \psi(\theta_i)$ as a function of θ summed over all possible hinges i of the mesh. The function ψ is an application-specific function of the bending angle θ. We can obtain the bending force by differentiating the energy with respect to the vertex position x as $F(x) = -\sum_i \nabla\psi$ and the hessian can be obtained as $H(x) = \sum_i \psi' \text{Hess}(\theta_i) + \psi'' \nabla\theta_i{}^T \nabla\theta_i$, where $\text{Hess}(\theta_i)$ is the second-order derivative of θ_i with respect to x. In this work, the function $\psi(\theta)$ is given by the equation

$$\psi(\theta) = k(2tan(\frac{\theta}{2}) - 2tan(\frac{\bar{\theta}}{2}))^2, \theta \in (-\pi, \pi), \tag{3.10}$$

where k is a constant dependent on material properties and $\bar{\theta}$ is the angle at the rest configuration. Details on how to compute the gradient and the Hessian of the bending energy analytically are given in [18]. At each timestep, the bending forces and the Jacobian of the forces are updated based on the vertex positions.

3.4 Estimating Sheet Parameters

3.4.1 Overview

This section outlines the methodology used for estimating the physical parameters of various sheet-like deformable materials, including composite prepregs,

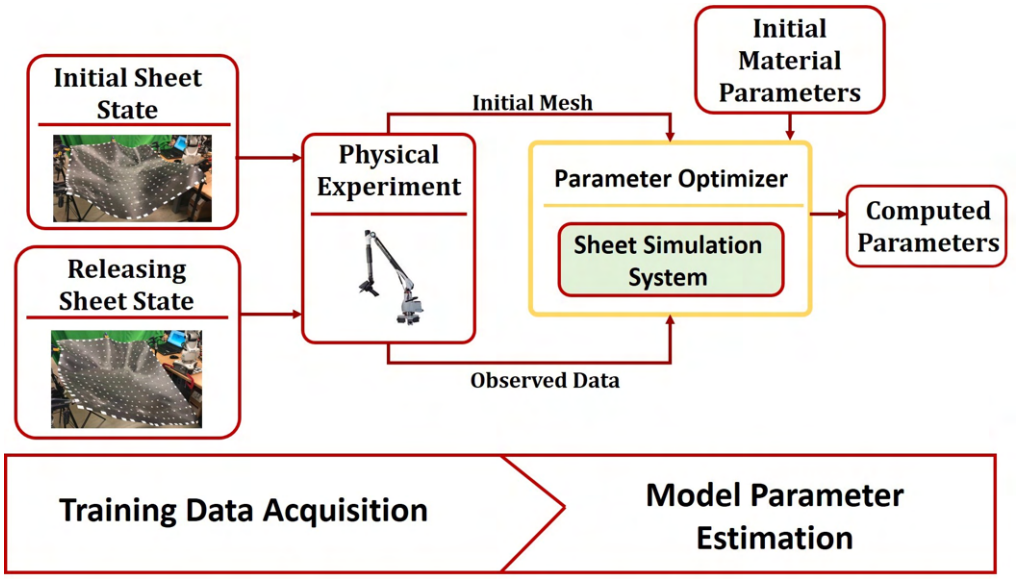


Figure 3.4: Process Overview: The initial state of the sheet is defined as the sheet configuration under initial boundary conditions. The releasing/released sheet state is defined as the sheet behavior after releasing one of the boundary conditions. After conducting physical experiments, initial mesh and observed data in the form of a mesh of the sheet are obtained from two sheet states, respectively. The data is further fed to the optimizer to acquire computed parameters.

cotton cloth, felt, and canvas. The parameter estimation process was divided into three main stages: (1) data acquisition, (2) data preprocessing and simulation-based optimization, and (3) parameter validation through testing.

Data acquisition was conducted in a physical environment using a guided manipulation procedure. During this phase, the sheet materials were subjected to a controlled release of boundary constraints, and their deformation was recorded using a 3D depth-sensing system to capture point cloud data. Two key configurations were recorded: the initial state, where the sheet was held under fixed boundary conditions, and the released state, observed after one constraint was removed. These configurations were converted into mesh representations suitable for simulation.

Subsequently, the mesh data was passed into an optimization framework operating entirely in a simulated environment. This framework iteratively adjusted the material parameters to minimize the discrepancy between simulated and observed deformations. The result was a set of optimized material parameters for each type of sheet.

Figure 3.4 illustrates the overall pipeline. The full details of the parameter optimization framework are presented in Section 3.4.3. This method was applied independently for all four material types, allowing a comparative evaluation of the model's ability to generalize across materials with varying mechanical properties.

3.4.2 Acquisition of Training and Testing Data

The data was collected using a Hexagon RS5 laser scanner and contact probe attached to the Romer Absolute Arm (87-Series). The class 2M laser scanner is hand-operated and generates point cloud data. Fig 3.5 shows all scanning equipment below. The sampling filter can be manually set to optimize the

Settings	Value
Max Capture Rate	752000 points/sec
Percentage	25 %
Exposure Time	200
Point Spacing	0.052 mm
Line Width	130 mm
Sampling Rate	51 Hz

Table 3.1: Settings used on the Hexagon RS5 laser scanner

percentage of points recorded and exposure time based on the light in the sampling environment and the color of the scanned component to reduce noise and capture the target locations. The settings can be seen in Table 3.1 below.

Figure 3.5: (A) Romer Absolute Arm, (B) Laser Scanner, (C) Contact Probe, and (D) Contact probe variety

The contact probe was calibrated using a TESA TKJ 3mm Ruby Ball Probe. The probe and laser were subject to accuracy specifications designated by the manufacturer in Table 3.2.

Equipment	Accuracy
Laser Scanner	0.028mm
Contact Probe	0.046mm

Table 3.2: Accuracy Specifications for Laser Scanner and Contact Probe

3.4.2.1 Sheet Preparation

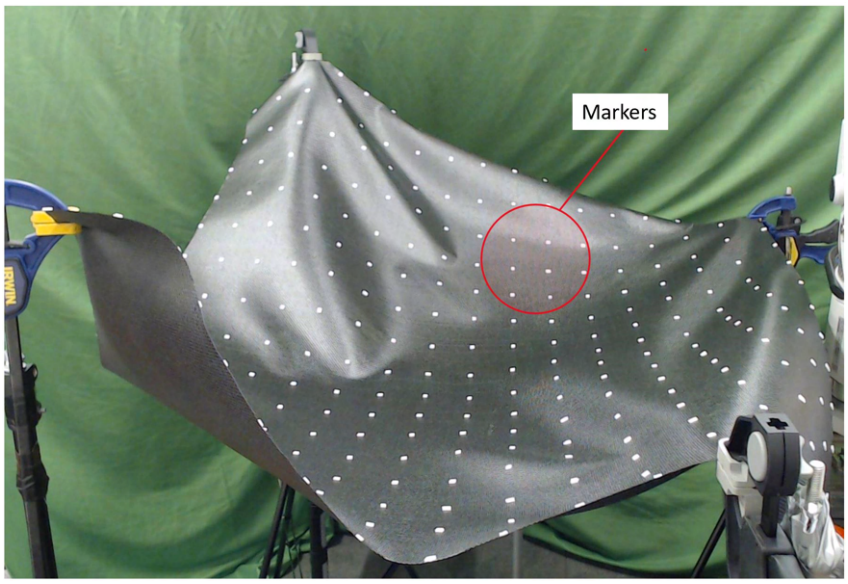


Figure 3.6: Sheets were prepared by attaching 289 quarter-inch markers to each sheet

To prepare each sheet for scanning, quarter-inch markers were placed at even intervals along the sheet in a 17x17 grid for a total of 289 markers. The markers used were white 3M double-sided foam tape squares cut to the correct size. The markers were raised from the surface, allowing the scanner to detect them easily. One of the sample materials used was white in color, and the markers had to be colored in black to be picked up by the scanner. Otherwise, the color difference was considered optimal for scanning with the previously mentioned settings. The four locations of the clamps obscured the markers and were recorded separately via the contact probe for use within the simulation as a fixed point.

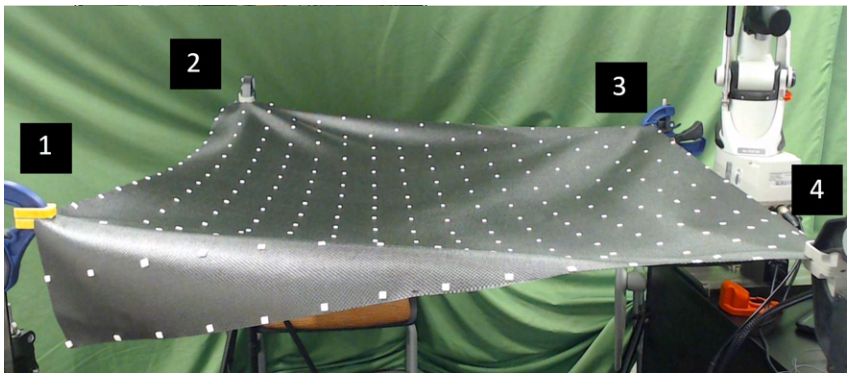


Figure 3.7: Clamps fixed the sheet configuration at four points. This is defined as the initial state.

3.4.2.2 Sheet Deformation Data Generation

The data was collected during two trials for each material tested. Each sheet edge was grasped using four vertically fixed clamps, allowing the sheet to rest suspended between them, as shown in Fig. 3.7. The clamping locations were chosen at differing and unique distances from adjacent corners. The collection period for each trial featured a five-stage setup, with each stage introducing a new modification to sheet positioning by moving the clamp with respect to the room while retaining the clamping position on the sheet. The translational movement of the clamp was chosen arbitrarily, but all movements had a change in distance of no more than 350 mm.

The first stage required placing the sheet under the initial boundary conditions of the first stage, resulting in the sheet being suspended in a relaxed, horizontal position. The next three stages involved isolated movement of only one grasping location to a new position, with a general increase in the z-axis, coinciding with x- and y-axis movement toward the center of the sheet. The last stage for every trial was the release of the clamp, allowing the material to settle into a hanging position. Within each stage, each clamp changed location only once and was allowed to settle into position until no visible movement

could be detected. After clamp relocation, the new clamp locations were collected via the contact probe at the beginning of each stage, and the sheet was scanned using the laser scanner. This procedure can be observed in Fig. 3.8.

The point cloud data collected from each stage was processed using PC-DMIS and exported as an XYZ file for post-processing. Each sheet contained thousands of data points from each stage, with each marker averaging 300 points. The simulator requires only one point from each marker, so each XYZ file was processed through Blender software using an original Python script to isolate the center point of each marker and export them as 3D points relative to the world coordinate system of the scanner base. A comparative image of the same sheet is shown above in Fig. 3.9, demonstrating the reduction from the point cloud marker clusters to single points.

On successful construction of the initial mesh, the boundary conditions of the sheet are changed by releasing one of the clamps. Note that the remaining clamps should not be moved to maintain consistency throughout the process. The sheet state after changing the boundary conditions is defined as the released state. Fig. 3.8 shows the difference between the initial state and the released state.

Since no mesh is required to generate from the released state, the point cloud data of such a state is clustered to represent each marker. The data set acquired in this process is further defined as the observed data. Fig. 3.9 shows the point cloud data clustering process.

3.4.3 Model Parameter Estimation

3.4.3.1 Sheet Simulation System

The crucial element of the parameter estimation process is the composite sheet simulation system. Fig. 3.10 illustrates the block diagram for the proposed

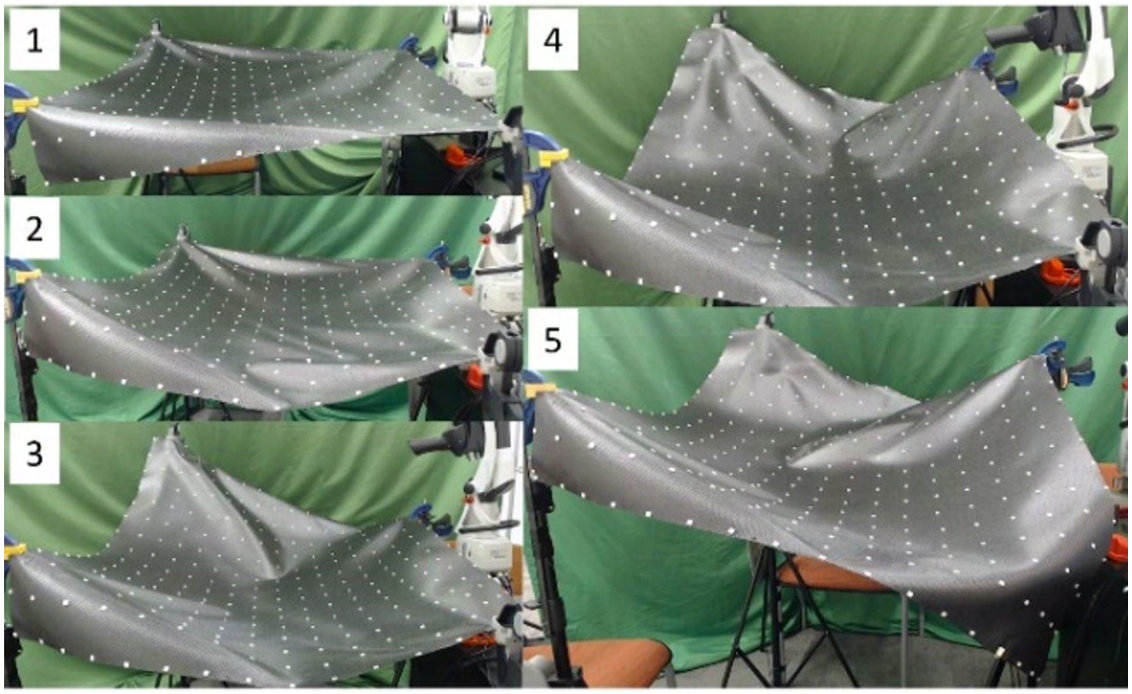


Figure 3.8: (A) Labeled sequence of stages depicting sheet position movements during one trial

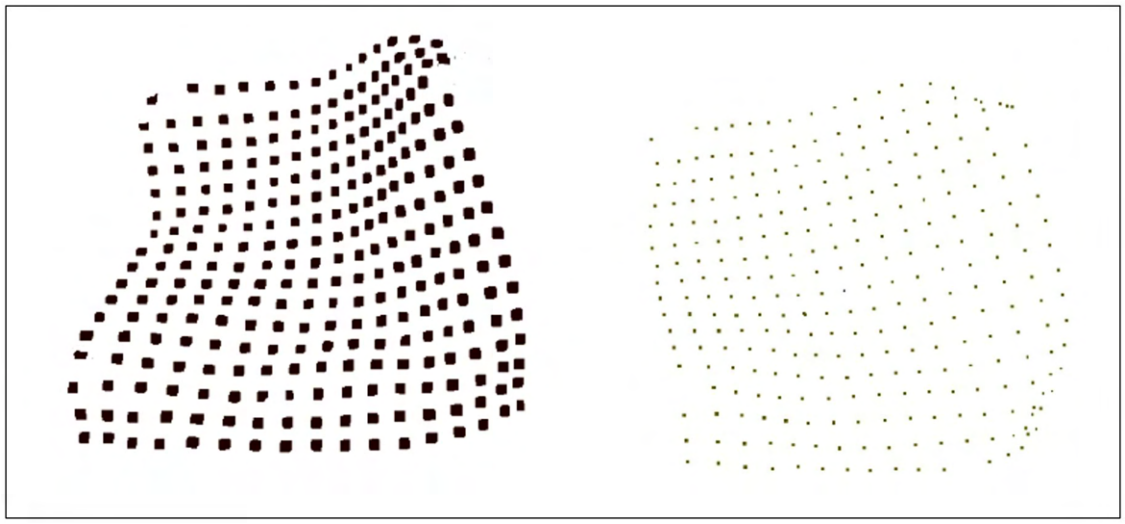


Figure 3.9: Point cloud clusters (left) vs single point vertices (right) on one sheet

simulation system. The simulator system utilizes the initial mesh as geometric information input and applies the model parameters to construct the composite sheet model.

The model parameters are categorized into two types: 1) Material parameters and 2) Integrator parameters. Material parameters consist of the surface density and the internal force parameters: tensile stiffness, shear stiffness, and bending stiffness. On the other hand, integrator parameters include damping stiffness and damping mass.

Once the sheet model is constructed [28], the numerical integrator applies the boundary conditions and external forces, such as gravity, to the sheet model and solves the deformation equation [29]. After the predicted mesh is generated, the prediction error is obtained by comparing the predicted mesh and the observed data. The prediction error, E, is a function of model parameters, P, initial mesh, M, and observed data, O. Algorithm (1) is used to calculate the prediction error function.

Algorithm 1: Prediction Error

Input:
$d \in O$
$M_{pre} \leftarrow Simulate(P, M)$
Output: $Error$
for d **do**
    $v \leftarrow FindClosest(d, M_{pre})$
    $disp \leftarrow |v - d|$
$Error \leftarrow Average(disp) + 0.5 * Maximum(disp)$;

Let d be a data point in the observed data, O. After getting the predicted mesh, M_{pre}, by applying P and M to the simulator, we query each d to find the closest vertex, v, on M_{pre}. The prediction error is then defined as the average displacement between v and d plus half of the maximum displacement across the entire mesh. The prediction error is used for parameter optimization, which will be discussed in Section 3.4.3.3.

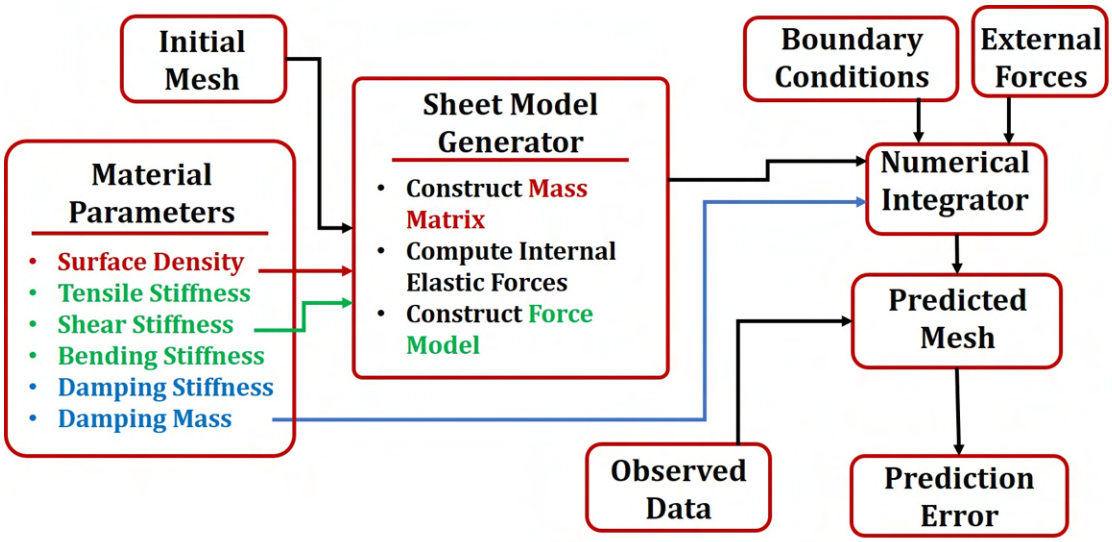


Figure 3.10: Simulation Process Overview: The simulator used the material parameters and mesh information to predict sheet behavior under specified conditions.

3.4.3.2 Parameter Boundary Selection

The model parameter identification uses a nonlinear optimizer to compute the optimal parameters for simulating the composite prepreg. However, the optimizer is not required to compute all model parameters. Some of these parameters can be measured directly. As mentioned in section-3.4.3.1, model parameters comprise material parameters and integrator parameters. Since integrator parameters are not related to model construction, the damping stiffness and damping mass are set to 1.0 and 0.0, respectively. The surface density can be measured directly by scaling the sheet and dividing it by the surface area. Therefore, the remaining parameters, tensile stiffness, shear stiffness, and bending stiffness, are the parameters that require optimization.

To ensure the effective performance of the optimizer, appropriate upper bounds, lower bounds, and initial values for the parameters are required. Fig. 3.11 shows the input-output diagram for parameter boundary selection. Tensile

stiffness and shear stiffness are sampled into three categories: 1) 500, 2) 5000, and 3) 50,000. For bending stiffness, the parameter is sampled into 1) 0.01, 2) 0.001, and 3) 0.0001.

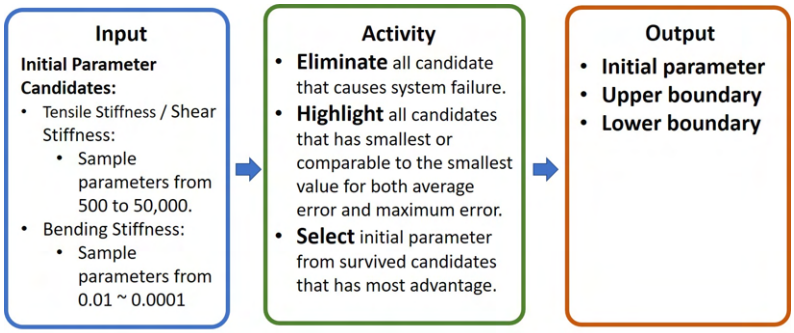


Figure 3.11: Input-Output Diagram for Parameter Boundary Selection.

After getting three samplers for each parameter, we shuffled them and got 27 candidates to test for parameter feasibility. All candidates that cause system failure are eliminated, and those with the smallest prediction error or comparable to the smallest one are highlighted. Candidate $\{5000, 5000, 0.001\}$ has the best overall performance among 27 candidates, and therefore, it is selected to be the initial value set for the parameter optimization. Then, the initial values are used as the median for the parameter boundary. Thus, the upper boundary for the {Tensile stiffness, Shear stiffness , Bending stiffness} is $\{9000, 9000, 0.0011\}$. The lower boundary for the parameter set is $\{1000, 1000, 0.0009\}$.

3.4.3.3 Optimization Algorithm

The optimization library used in this work is NLOPT[30], an open-source library for nonlinear optimization. The selected algorithm is ISRES, Improved Stochastic Ranking Evolution, a global-gradient-free optimization algorithm[31]. Fig. 3.12 shows the block diagram for the optimization process.

The parameter optimizer utilizes the training sets and initial parameters as inputs and calculates the prediction errors for each sampler. The goal of the optimizer is to find the parameters that minimize the sum of the errors from the training set.

$$\{m_1, \ldots, m_6\} \in M,$$
$$\{o_1, \ldots, o_6\} \in O$$
$$P_{opt} \leftarrow \arg\min_{P} \sum_{i=1}^{k} (E_i(P, m_i, o_i)) \tag{3.11}$$

The optimization problem can be expressed as equation (3.11). Let M be the initial meshes in the training set, which contains mesh data, m_i. O is the observed data set, which contains observed data o_i. Recall that the prediction error, E_i, is defined in the algorithm (1). Since five samples are used for model training, k is set to 5. The optimizer tries to find the parameters that reduce the average and maximum displacement differences between the predicted mesh and the observed data for each training set. This error value was set to (Average Error + 2*Max error). Once the error converges, the identified parameters, P_{opt}, simulate the composite sheet in real time.

3.5 Results

3.5.1 Experimental Specifics

Four materials are considered in this work: prepreg composite sheets, common cotton cloth, felt, and canvas. Boeing Inc. supplied the composite sheet, which came as 3ft x 4ft sheets, while all other sheets were purchased locally and were 2ft x 2ft in dimension, shown in Fig.3.13. As discussed previously, the prepreg

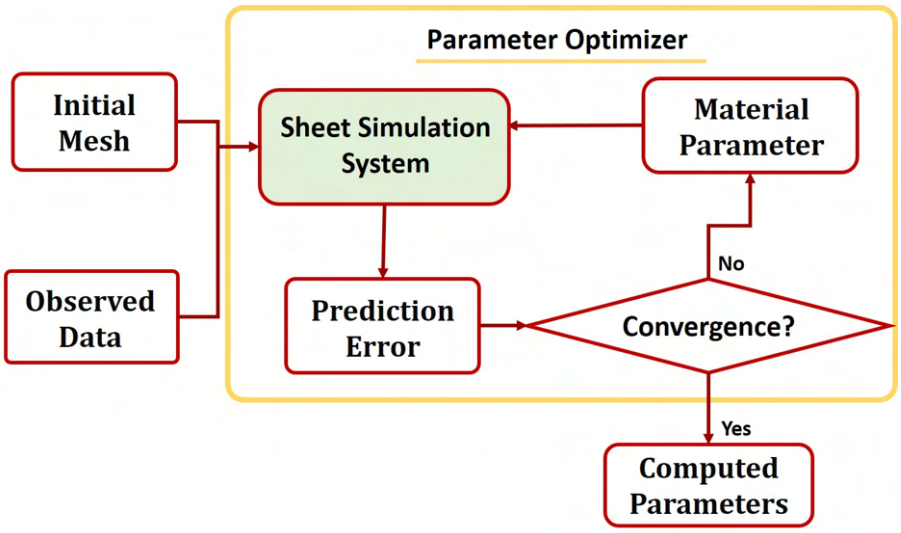


Figure 3.12: Parameter Optimization Process.

composite sheet contains viscoelastic properties due to the resin contents, while all other sheets contain no resin or viscous content. The elastic materials all have a standard/uniform weave. Details of the elastic materials are shown in Table. 3.3.

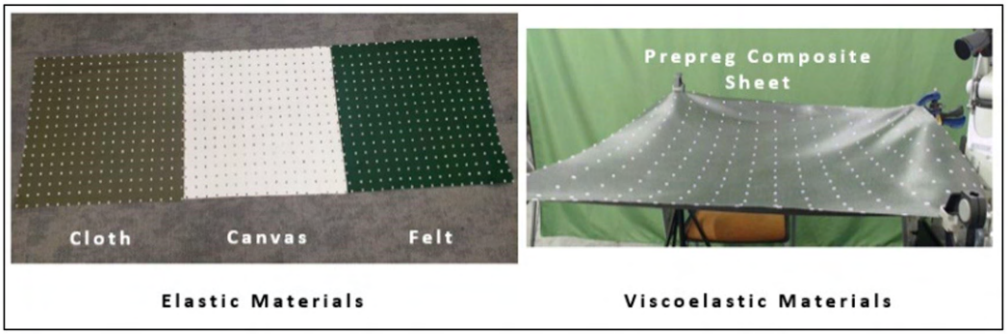


Figure 3.13: Sheets used for four material samples. Left: Elastic Fabric Materials. Right: Viscoelastic Prepreg Material.

The composite prepreg material provided by Boeing came with manufacturer-prescribed density and thickness specifications (refer to Table 3.4). The elastic

Table 3.3: Elastic Material Measurements

Measurement	Cloth	Canvas	Felt
Side1	611 mm	613 mm	615 mm
Side 2	611 mm	613 mm	612 mm
Thickness	0.1 mm	0.65mm	1.8mm
Mass	42.1 g	168.9 g	66.9 g
Surface Density	0.1126 kg/m^2	0.4509 kg/m^2	0.0018 kg/m^2

materials came with no specifications and thus required density calculations and thickness measurements, shown below in the Table. 3.3

Table 3.4: Viscoelastic Material Specifications

Measurement	Composite Sheet
Side1	1.17 m
Side 2	0.975 m
Thickness	0.3 mm
Poisson's ratio	0.3
Surface Density	0.3 kg/m^2

The physical experiment process described previously was repeated twice for each material type for a total of eight trials. In the case of the composite sheet, two different sheets were used, as it was determined that the combination of experimental movements and exposure to air may degrade the sheet, possibly providing poor results. The elastic materials were all used on the same sheet twice. Of the two trials for each material, the data collected from one trial was used for training purposes, while the other was used for testing purposes.

3.5.2 Sheet Parameter Estimation

Estimation of sheet parameters was accomplished through the simulator, as previously discussed. The internal parameters were considered in one state throughout the simulation procedure. Still, research has shown that linear elastic woven fabrics vary in many of their internal parameters due to the anisotropic behaviors of the fabric [17]. As such, manual manipulation of

the parameters from the initial internal parameters was needed to find an ideal starting range for the optimization. The initial parameters were found experimentally and were then run through the optimizer, providing the lowest error results.

The following two tables highlight the training and testing performance of the simulator in terms of the average error and maximum error for the training and testing data sets. Table 3.5 gives the results for the training of fabric materials, and Table 3.6 gives the results for the testing of fabric materials.

Table 3.5: Fabric Material Training Data

Fabric Material Training		
Material	Avg Error [cm]	Maximum Error[cm]
Felt	2.3	7.8
Cloth	1.1	5.5
Canvas	1.4	6.0

Table 3.6: Material Testing Data

Fabric Material Testing		
Material	Avg Error [cm]	Maximum Error[cm]
Felt	2.6	8.2
Cloth	1.6	5.6
Canvas	1.5	6.2

Figs. 3.14, 3.15, and 3.16 show the configuration of stage 5 for the felt, cloth, and canvas sheets, respectively. In each figure, the upper image is the observed position photograph, the middle image is the mesh generated from the scanned point cloud data, and the lower image is an image of the simulation mesh at the end of the stage.

The following figures depict the initial and final positions for the composite sheet trials. Within the initial stage shown in Fig. 3.17, consistency can be seen between the image, observed data, and simulation data.

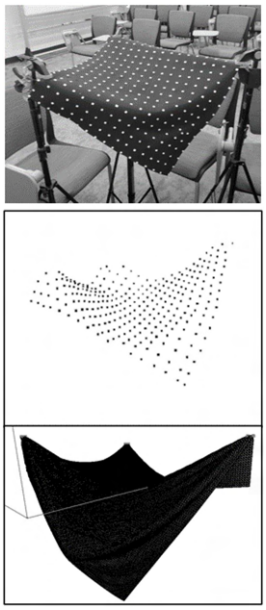


Figure 3.14: Trial 1 Stage 5 Results of Felt Sheet: (Upper) Observed Position Image, (Middle) Generated Mesh, and (Lower) Simulated Mesh.

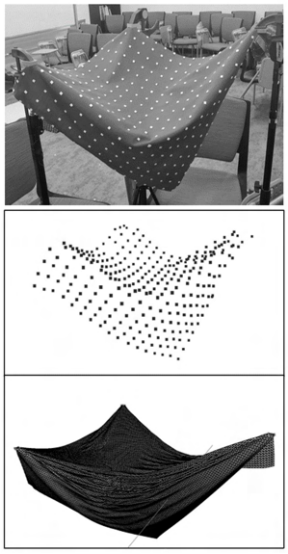


Figure 3.15: Trial 1 Stage 5 Results of Cloth Sheet: (Upper) Observed Position Image, (Middle) Generated Mesh, and (Lower) Simulated Mesh

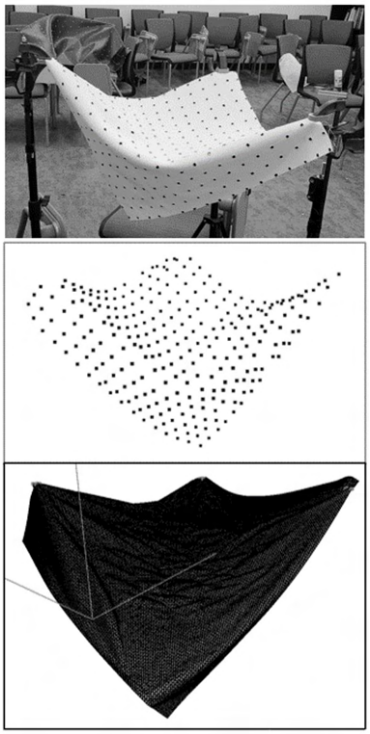


Figure 3.16: Trial 1 Stage 5 Results of Canvas Sheet: (Upper) Observed Position Image, (Middle) Generated Mesh, and (Lower) Simulated Mesh

Table 3.7: Composite Training and Testing Data.

Composite Sheet Material		
Sheet	Avg Error [cm]	Maximum Error [cm]
Training 1	1.6	10.2
Testing 1	1.4	13.9
Testing 2	2.2	13.6

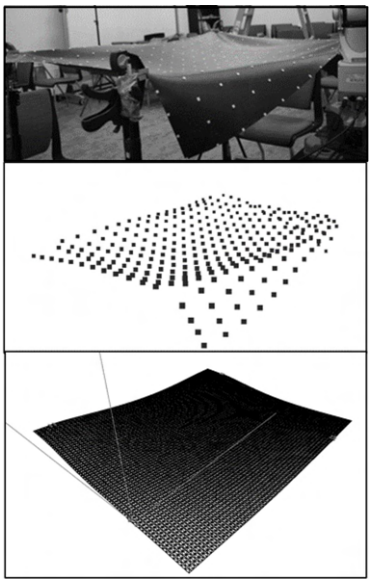


Figure 3.17: Trial 1 Stage 1 Results of Composite Sheet: (Upper) Observed Position Image, (Middle) Generated Mesh, and (Lower) Simulated Mesh.

Within the final stage, there was consistent error across all material types, however this error was higher than all other stages indicating that this position was the most difficult for the simulator to emulate. Fig. 3.18 depicts this position and displays results from the observed data and simulated data.

The error from the training data is shown in Table. 3.7. The optimal parameters were a bending stiffness of 4.77682e7 N/m^2 and a shear stiffness of 2942.93 N/m. The optimization process began with initial values based on the data sheet supplied by Boeing, as shown in the Table below. 3.8.

Table 3.8: Initial Optimizer Parameters

Parameter	**Initial Value**
Sheet thickness	0.0003 [m]
Poisson's ratio	0.3
Sheet density	0.3 $[kg/m^2]$
Shear Stiffness	3.3e2 $[N/m]$
Bending Stiffness	1.1e8 $[N/m^2]$

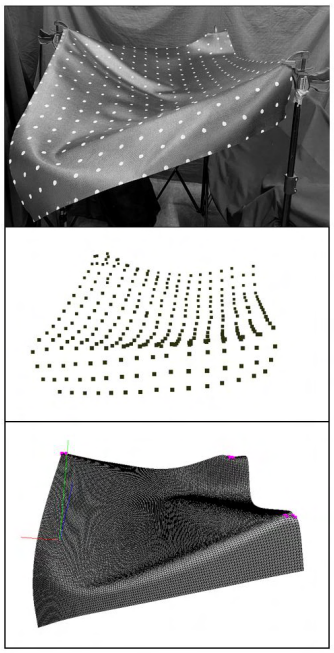


Figure 3.18: Trial 1 Stage 5 Results of Composite Sheet: (Upper) Observed Position Image, (Middle) Generated Mesh, and (Lower) Simulated Mesh.

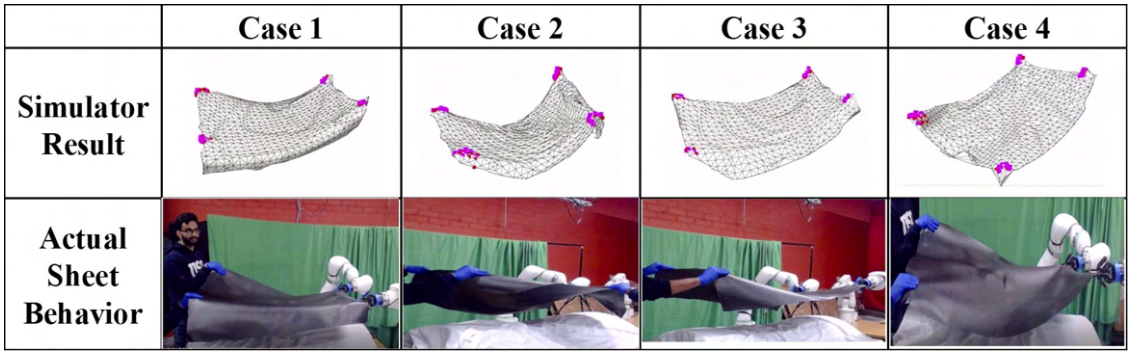


Figure 3.19: Real-time sheet simulation result: The first row shows the predicted mesh from the simulator. The second row shows the actual behavior of the composite sheet.

3.5.3 Sheet Simulation

The results obtained from the material parameter estimation are then utilized to develop a simulation system that can predict the material's behavior under varied constraints. This section focuses on one of the potential applications of such a high-fidelity simulator. In Composite Prepreg sheet layups, as discussed earlier, the sheet needs to be held at appropriate locations to avoid any potential defects during the layup. A sheet simulation system that can predict the appropriate locations for grasping the sheet can be instrumental in process planning. The key elements of such a simulation system would be (1) A Sheet Simulation Model generated using the estimated material parameters and (2) A Real-Time Sheet Tracking System that can generate a mesh of the sheet in real-time at high frame rates and that can then be used for comparison with the simulated result. To demonstrate the feasibility of such a system, an experiment was designed using the carbon fiber-reinforced epoxy sheet provided by Boeing Inc.

The methodology for material parameter estimation proposed earlier was used to generate an appropriate model for the Boeing Composite Sheet. The simulation system was built using the VegaFEM library [11]. Fig. 3.20 shows the block diagram of the proposed real-time sheet tracking and simulation process.

The dimensions of the composite sheet used in this experiment were 4ft x 3ft, which required a multi-camera system to track the entire sheet. The real-time sheet tracking system proposed in this study consists of three RealSense D415 cameras. The entire sheet is captured by fusing the RGB-D feed from all three cameras. Color-based filtering techniques are employed to filter the prepreg sheet from the rest of the scene. Resampling is performed on the resulting points to obtain a uniform distribution of the filtered points. The face normals are then recomputed using these points. Surface reconstruction

is performed in scale space by implementing Advancing Front Surface Reconstruction [32]. A scale-space describes the point set at a dynamic scale, and this additional dimension allows us to control the degree of smoothness required for the reconstruction [33]. After post-processing, a surface mesh with around 6,000 triangles was obtained.

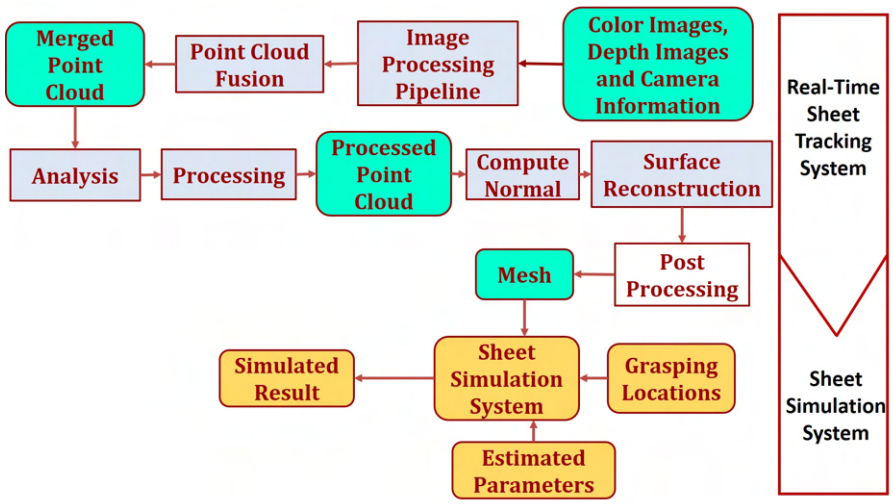


Figure 3.20: Real-time sheet simulation process.

The composite sheet was constrained in four different states to test the system's performance. Initially, the sheet is constrained at four grasping locations. A human and two 7 DOF robotic manipulators KUKA iiwa R7, are used to apply the fixed constraints. The Real-Time Sheet Tracking System captures the mesh and grasping locations for the corresponding state. In the next step, one of the grasping locations is released. The sheet is allowed to settle to its minimum energy state, and the real-time sheet tracking system captures the sheet in its respective state. This exercise is repeated for three other states by changing the grasping locations and capturing the sheet behavior. This data is then compared to the predictions generated by the simulation system. Fig. 3.19 contrasts the four states where the composite sheet was suspended and the corresponding simulator predictions.

The proposed simulation and real-time sheet tracking system can play a pivotal role in predicting optimal grasping locations for the draping of composite sheets. These grasping locations can then be used to deploy a human-robot collaborative cell where the robots can aid the human in the draping process by holding the sheet appropriately [3]. Furthermore, the sheet tracking data can act as a rectifying input for the grasping location in case of sub-optimal simulator predictions. Such a conjunctive system can aid in streamlining the prepreg layup process and achieve an overall higher degree of automation.

3.6 Summary

This chapter presented a framework for estimating material parameters for sheet-like deformable objects and constructing high-fidelity simulation models that capture their dynamic behavior under external constraints. The focus was on modeling large, compliant sheets – including prepreg composites, cotton cloth, felt, and canvas– using a data-driven approach grounded in elasticity theory and implemented through an efficient thin-shell finite element formulation.

A thin-shell simulation environment was developed using the VegaFEM library, enabling rapid evaluation and optimization of material parameters. A multi-stage methodology was introduced, encompassing physical data acquisition, parameter estimation through simulation-based optimization, and validation across multiple materials. The modeling approach incorporated domain-relevant constraints such as fixed supports and gravity, and demonstrated the ability to simulate sheet deformation under realistic conditions with high accuracy.

The parameter identification pipeline enables the generation of digital twins that can serve as predictive tools in downstream robotic applications. Specifically, this chapter's contributions lay the groundwork for planning and executing robotic manipulation strategies that depend on understanding the deformation dynamics of flexible sheets. This capability is especially critical for automating prepreg composite layup–a process where accurate prediction of material behavior is essential for avoiding defects and ensuring high-quality results.

The findings in this chapter support several broader goals of the dissertation:

- First, they demonstrate how physical priors, when embedded into simulation and learning pipelines, improve both accuracy and efficiency.
- Second, they provide a validated modeling foundation for subsequent chapters focused on task and manipulation planning (e.g., grasp planning and sequencing for tasks involving deformable sheets).
- Finally, they highlight the importance of combining real-world sensing with simulation to build generalizable, explainable models of soft, deformable sheet-like objects.

The subsequent chapters build directly on this work by leveraging the learned simulation model for robotic manipulation planning, where simulation is used as a surrogate model to generate feasible and task-aware robotic actions. Furthermore, it's also used to generate physics-informed synthetic data for defect detection in sheet-like objects.

Chapter 4

High-Fidelity Simulation of Shell-Like Deformable Objects Using FEM

4.1 Introduction

Deformable objects have long been studied in the robotics and simulation communities, particularly focusing on cases where the deformations are primarily surface-level, such as 1D ropes, or 2D sheet-like structures like fabrics and composite layers. However, the landscape of deformable object manipulation is rapidly evolving. Driven by the explosive growth of online retail and consumer demand for small-volume shipments, a new class of deformable objects has become increasingly prevalent across logistics, warehouse automation, and fulfillment industries: deformable packages.

These packages, such as polybags and padded mailers (Refer Fig. 4.1), present a unique structure: they can be conceptually viewed as two flexible surfaces (top and bottom sheets) fused together, containing an internal object that moves independently within the package. This layered system introduces new challenges as external deformations are compounded by internal dynamics, as the mass and shape of the contained object shift unpredictably during manipulation. Unlike simple sheets, these shell-like deformable objects require

specialized modeling and planning approaches that account for both surface compliance and internal mass movement, making them fundamentally different from traditional deformable systems studied in robotics.

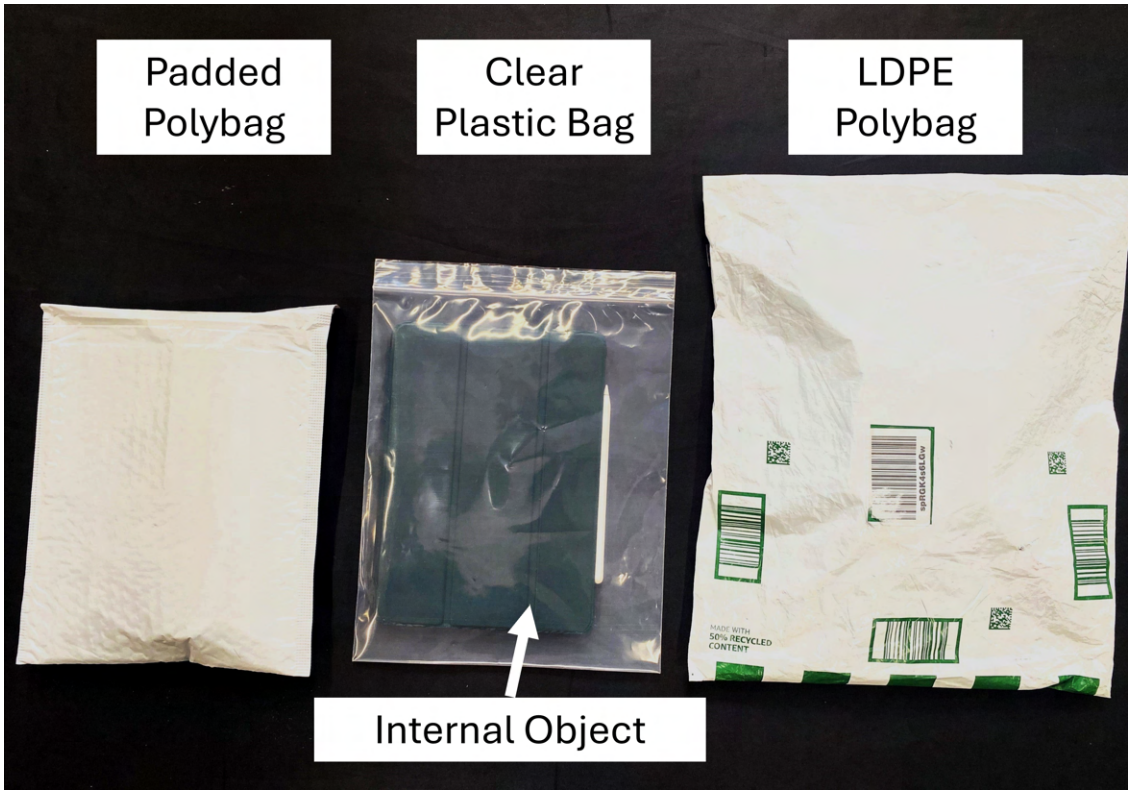


Figure 4.1: Examples of deformable packages made from various materials used in the fulfillment industry. The internal object, a tablet in this case, is enclosed within the package, demonstrating the challenge of handling such deformable structures.

Despite their potential, robotic systems face significant challenges in handling deformable packages due to their complex nature. In large-scale fulfillment centers, these packages are primarily handled using suction-based end effectors, as vacuum gripping is well-suited for their smooth, non-porous surfaces (Refer Fig. 4.2). The primary task in these environments involves picking up a package from a moving conveyor belt and placing it into a designated bin for

sorting, packaging, or shipping. However, unlike rigid boxes, deformable packages undergo complex shape changes when manipulated, making traditional rigid-body suction-based grasping strategies ineffective.

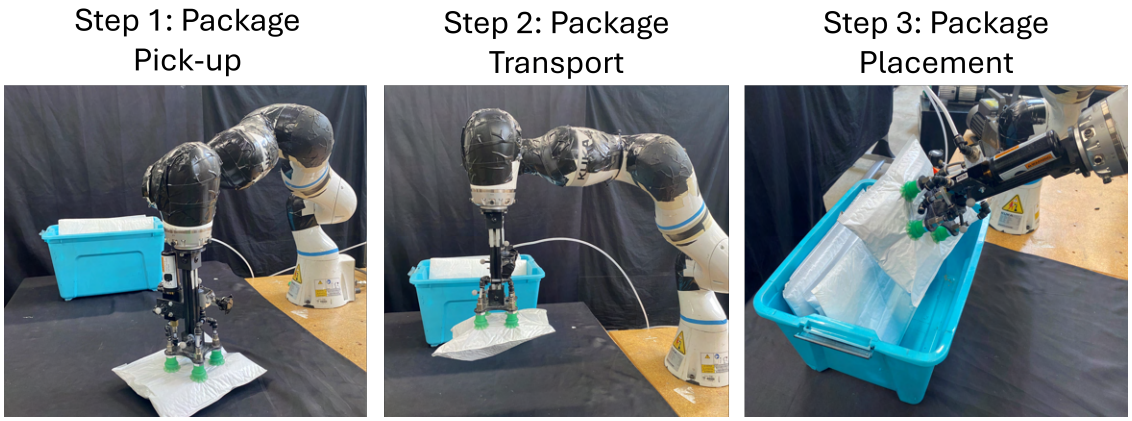


Figure 4.2: Example of a robotic cell using a suction-based tool to pick and place a deformable package into a bin, where failures can occur at any stage of the process.

Further complicating the problem, these packages often contain loosely packed internal objects that shift dynamically based on the robot's motion. This internal movement alters the package's center of mass and deformation characteristics, making it difficult to anticipate failure modes and ensure stable manipulation. As a result, robotic systems must account for the package's external deformability as well as the complex interplay between the internal object and the surrounding flexible material.

Although suction-based manipulation is effective in handling such complexities, several failure modes can still disrupt the process, leading to inefficiencies in the form of slower trajectories and the risk of potential package damage if trajectories are too fast. The three primary failure modes (Refer Fig. 4.3) encountered when handling deformable packages include: (1) Loading Failures: The suction cup fails to establish a secure grip, often due to insufficient suction or improper seal due to excessive surface irregularities; (2) Peeling

failure: Wrinkles or folds in the package can cause premature loss of suction, resulting in the package detaching from the end effector and (3) Shear failure: Rapid movements or improper trajectory planning can lead to excessive lateral forces, breaking the suction seal and causing package drop. These failures primarily stem from the way the package deforms during manipulation and how the internal object motion further influences these deformations. The interaction between the shifting internal mass and the flexible packaging material introduces instability, making it challenging to predict and prevent failure modes.

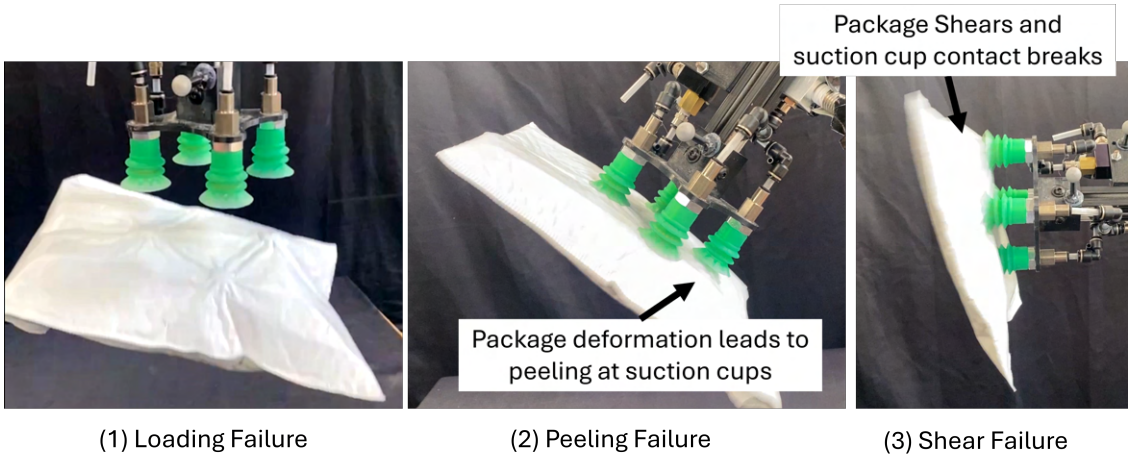


Figure 4.3: Illustration of three failure modes in package handling, primarily caused by wrinkles from package deformation and the dynamic motion of the internal object

Human operators effortlessly adapt to such variability in package shape, weight distribution, and internal motion, building an intuitive model of the deformable package-object system to manipulate it effectively. Similarly, enabling robots to autonomously handle such packages requires developing a precise model of this multi-object deformable system, a problem that remains largely unexplored in robotics. While challenging, creating such a simulation model could accelerate the large-scale adoption of robotic automation for these tasks. A well-designed simulation can predict package deformations and internal object

dynamics, allowing for the automated computation of optimal manipulation trajectories. These trajectories/policies must strike a balance between safety, minimizing failure risks leading to package drop, and efficiency by executing trajectories at maximum feasible speeds.

Recent advancements in learning-based manipulation have shown promise in equipping robots to handle the complexities of deformable package manipulation. These methods are particularly appealing for warehouse automation, where robots must adapt to diverse package conditions while ensuring safety and reliability. However, a major challenge of learning-based approaches is their reliance on large, high-quality datasets to provide informative inductive biases for robust policy training. Data collection in real-world warehouse environments is expensive and time-consuming due to constantly changing workspace layouts and package attributes. Even minor variations in package properties can necessitate extensive fine-tuning, further increasing the data burden. A common strategy to mitigate this challenge is training robotic policies in simulation, where data generation is scalable, controlled, and reproducible. However, to the best of our knowledge, no existing simulation framework captures both deformable package behavior and internal object dynamics, creating a significant gap in transferring learned policies to real-world settings.

Physics-based simulators are widely used for rigid-body manipulation, but modeling deformable packages, especially those with internal objects and coupled dynamics, remains challenging due to complex material properties, contact-rich interactions, and high computational costs. Soft-body simulation [34, 35] offers a promising solution, enabling high-fidelity physics modeling for efficient policy learning. Unlike conventional simulation methods, these simulators leverage structured physics priors, improving data efficiency, reducing sample complexity, and enhancing generalization. Building on these advancements

and addressing the challenges of deformable package manipulation, we develop a physics-based simulation framework for multi-object deformable packages. This simulation is designed to compute optimal trajectories for safe and efficient robotic manipulation, ensuring seamless transfer to real-world conditions.

This chapter introduces a physics-based simulation framework for modeling deformable packages with internal dynamics. The proposed framework uses a hyperelastic model that captures both package deformation and internal object dynamics. The model represents the package as a hyperelastic triangular mesh with bending and elastic properties, while the internal object exhibits interdependent motion within the deformable structure. Suction-based constraints are incorporated to simulate interactions with a suction gripper, ensuring realistic manipulation dynamics. To bridge the sim-to-real gap, real-world deformation data is collected using a motion capture system for system identification, refining simulation parameters to match physical behavior. Additionally, a parallelized simulation environment is developed for scalable, parallelized training of manipulation policies and trajectory optimization, enabling robots to adapt to diverse package dynamics.

The contributions of this chapter are summarized as follows:

1. A high-fidelity simulation model of a deformable package with internal object dynamics

2. A simulation parameter learning framework that employs a structured loss to capture the nuances of the deformable package system accurately

3. An environment for parallelized training of robotic manipulation policies in simulation

The proposed approach is validated through real-world package transport experiments, demonstrating that the proposed simulation model accurately captures package deformation and object dynamics. The framework is computationally efficient and enables robots to adapt quickly to deformable packages in dynamic logistics settings. By integrating real-world deformation data, we enhance the development of safer and more efficient manipulation strategies. Additionally, in Section 4.8, we explore how this framework optimizes package handling efficiency and generates safer, more reliable trajectories for deformable object manipulation.

4.2 Related Works

4.2.1 Deformable Object Simulation

Researchers have successfully applied simulation across various deformable materials: 1D (rope) [36], 2D (fabric) [37, 38], 3D (elasto-plastic objects) [39, 40], and combinations of liquid, fabric, and elastoplastic objects [41–43]. While these works demonstrate a single-object system in simulation for various deformable materials, the specific challenges for simulating multi-object systems with constrained deformations - particularly packages with internal objects - remain underexplored. Our framework addresses this gap by developing a hyperelastic model that accurately captures the intricate dynamics of package-object interactions while maintaining computational efficiency.

Various numerical methods have been applied to the modeling of deformable objects, each with distinct advantages for particular material behaviors. The Material Point Method (MPM) has been employed for materials undergoing large deformations and topology changes [39, 44], while Position-Based Dynamics (PBD) offers computational efficiency and unconditional stability

for real-time applications [36, 45]. The Finite Element Method (FEM) remains the gold standard for accurately modeling hyperelastic materials [40, 46], with implicit integration schemes providing superior stability for stiff materials. Our work employs implicit Euler integration with FEM to capture the complex interactions between deformable packages and internal objects while maintaining numerical stability.

The development of specialized software frameworks has been crucial for making simulation accessible and computationally efficient, enabling researchers to implement complex physical models without rebuilding systems from scratch. Several notable frameworks [47–49] have emerged recently to address this need. For multi-physics simulation, GradSim [46] combines differentiable physics with differentiable rendering to jointly model scene evolutions and appearance. NVIDIA Warp [34], which we employ in our work, incorporates the computing engine from GradSim, and offers a GPU-accelerated Python framework supporting both rigid and soft body simulation, with particular strengths in parallel computation and handling complex material models. We selected Warp for our implementation due to its efficient GPU utilization, flexible material modeling capabilities, and robust contact handling, which are essential for accurately simulating the interactions between deformable packages and internal objects at interactive rates.

4.2.2 System Identification and Policy Learning

System identification helps calibrate simulation parameters in mathematical models based on measured data. Differentiable simulation has been applied to estimate cloth stiffness and elasticity [37, 38, 46], though most studies validated parameter accuracy in simulated environments rather than real-world settings. Notable exceptions include DiffCloud [50], which utilized DiffSim

[47] to perform parameter estimation for cloth in quasi-static states using real point clouds. In [51], the authors extended estimation to include cloth dynamics using point cloud data. Our work not only aims to calibrate package material parameters but also to align internal object positions. To capture more comprehensive information, we employed motion capture systems to track the dynamics of both the package and its internal object.

Traditional simulation-based approaches for manipulation policy learning [3, 6, 52, 53] have primarily relied on sampling-based data collection, with limited focus on multi-object systems. More recent work has explored the manipulation of elastoplastic objects [54–56] and cloth [38, 57, 58], advancing simulation techniques for deformable materials. While studies have begun addressing deformable package simulation [5] and manipulation constraints [59], we extend these efforts by developing a high-fidelity multi-object simulation framework that captures real-world package interactions and enables policy learning through our high-performance simulation pipeline.

4.3 Problem Formulation

Consider an agent $\mathcal{A}$ manipulating a deformable thin-shell package $\mathcal{P}$. Let $\Psi_{package}$ be the set of parameters governing $\mathcal{P}$'s deformation and dynamic behavior. The package $\mathcal{P}$ contains an object $\mathcal{O}$, which is constrained to move within $\mathcal{P}$. The dynamics and motion of $\mathcal{O}$ are determined by its corresponding parameter set $\Psi_{\mathcal{O}}$. Thus, we define a multi-object system consisting of $\mathcal{P}$ and $\mathcal{O}$ manipulated by $\mathcal{A}$ and represented by the parameter sets $\{\Psi_{\mathcal{P}}, \Psi_{\mathcal{O}}\}$. This system as a whole is subjected to external forces F_{ext} and constraints C, which govern its evolution over time.

$\mathcal{P}$ is grasped by $\mathcal{A}$ via an interface $\mathcal{I}$, characterized by parameters ${}^{\mathcal{A}}\mu_{interface}$. This interface imposes constraints (C) on $\mathcal{P}$. In our case, $\mathcal{I}$ represents a suction

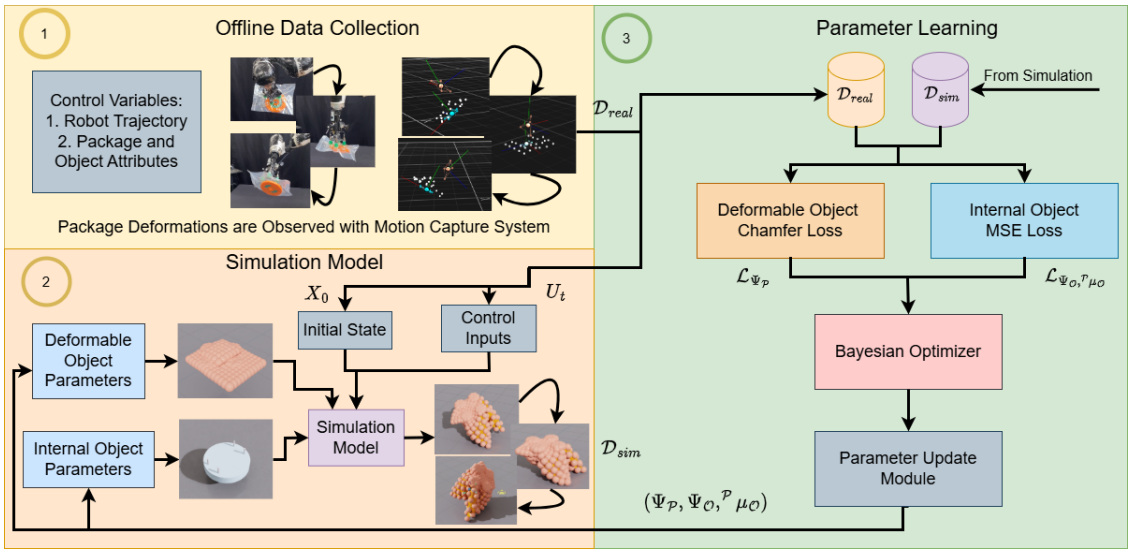


Figure 4.4: Overview of our proposed simulation parameter identification framework. As depicted, we have a real-world data collection phase, where we collect a set of robot trajectories and observe package deformation and object dynamics. Then we learn the corresponding parameters for our simulation to generate the model of our system.

cup interface between the robot and the package, but our formulation remains generalizable to non-suction-based interfaces. Similarly, we define ${}^{\mathcal{P}}\mu_{\mathcal{O}}$ to describe the interface parameters governing interactions between $\mathcal{O}$ and $\mathcal{P}$. These interface parameters, i.e., μ, determine how the objects interact with each other (see Section 4.4.3 for details). Additionally, the agent $\mathcal{A}$ excites the package-object system by applying external control inputs U_{ext}, primarily through end-effector trajectories τ executed to manipulate $\mathcal{P}$ for a given task. Additionally, we have external forces F_{ext} that include gravitational forces acting on the system.

At a given time instance t, we define $X_t^{\mathcal{P}}$ and $X_t^{\mathcal{O}}$ as the states of the package and the object, respectively. Our objective is to learn the parameter sets Ψ and μ of a function Φ that models the behavior of the package-object system. Given an external control inputs U, forces F_{ext} and constraints C, this function

should accurately predict $X_t^{\mathcal{P}}$ and $X_t^{\mathcal{O}}$. Thus, we formulate the following problem:

$$\Phi_{\Psi,\mu}(F_{ext}^t, U^t) \mapsto X_t^{\mathcal{P}}, X_t^{\mathcal{O}} \tag{4.1}$$

Where Φ represents the simulation model, U^t represents the control input from the commanded robot trajectory τ at the time t, and F_{ext} denotes the corresponding external forces acting on the system at time t due to gravity and the robot's acceleration.

4.4 Methodology

4.4.1 Overview

To model the function Φ, our proposed methodology consists of four key components (Refer Fig. 4.4): (1) A Real-World Data Collection Stage, (2) Package-Object Representation, (3) The Simulation Framework, and (4) The Real-to-Sim Parameter Learning Framework. Our data collection process is detailed in Section 4.6, where we describe how data is collected in alignment with our system representation, as outlined in Section 4.4.2. We adopt a particle-based representation [37] for the deformable package $\mathcal{P}$, a widely used approach for deformable objects due to its ability to capture intricate deformations. Moreover, modeling a closed package containing another object introduces challenges that require careful considerations during initialization and simulation, as discussed in Section 4.4.2.

Our proposed framework has the potential to serve as a valuable tool for the robotic logistics industry and other applications involving deformable objects, where learning safe and efficient policies is important. Therefore, the choice

of the simulation environment is crucial, as both simulation fidelity and computational efficiency directly impact performance. Recent advancements in soft-body simulators have made them strong candidates for learning complex manipulation policies [42]. These simulators provide access to high-fidelity deformation information, enabling end-to-end learning of manipulation policies. As described in Section 4.4.3, we employ NVIDIA Warp as our simulation environment due to its GPU acceleration capabilities, which enhance computation efficiency in optimization while maintaining high fidelity through advanced representations of deformable objects.

A key component of our system is the Bayesian optimization-based parameter learning module, which, given real-world data, minimizes the real-to-sim gap by learning the desired parameter set Ψ of the function Φ. This is achieved by reducing the discrepancy between observed system states in the real world and their corresponding behavior in the simulation environment (Refer to Section 4.4.4). Through this approach, we construct a simulation framework that facilitates robotic manipulation of deformable packages containing dynamic objects. Additionally, we demonstrate how this simulation model can be leveraged to learn manipulation policies for package transport in Section 4.5.

4.4.2 System Representation

The package $\mathcal{P}$ is modeled as a two-layer hyperelastic shell structure with thickness t, discretized in a particle grid $M \times N \times 2$. This dual-layer approach effectively captures the physical structure of packaging materials with finite thickness, enabling the model to represent realistic deformation behavior of both the upper and lower surfaces during manipulation. The surface is tessellated into triangular elements, with each 2×2 particle cell subdivided into two triangles. As illustrated in Figure 4.5a, the first triangle is formed

by connecting the bottom left particle with its right and upper neighbors, while the second triangle connects the top right particle with its left and lower neighbors. This tessellation approach ensures a balanced distribution of deformation forces across the surface. The three particles group is modeled as a triangular finite-element unit characterized by elastic stiffness $k_{tri,e}$, damping $k_{tri,d}$, and adhesion $k_{tri,a}$, which collectively form part of the parameter $\Psi_{\mathcal{P}}$ governing $\mathcal{P}$'s deformation. To capture out-of-plane deformation behavior, bending elements are introduced between adjacent triangular elements. The calculation of bending energy follows the formulation proposed by [60], which models the angular displacement between neighboring triangles, and the bending component is modeled as an edge and measured using elastic stiffness $k_{edge,e}$ and damping $k_{edge,d}$, which are remaining parts of the parameter set $\Psi_{\mathcal{P}}$. The edges connecting two layers also share the same structure of triangles and edges.

The interface $\mathcal{I}$ between agent $\mathcal{A}$ and package $\mathcal{P}$ consists of n suction cup units arranged in a circular configuration, where each suction cup is positioned at an angle θ_i, where $i \in \{1, 2, \ldots, n\}$, along a circle with a radius r_{cup} from the center. Each suction cup unit is modeled as a particle grid.

The particle grid with dimensions $N_\theta^{cup} \times N_r^{cup}$ representing circumferential and radial discretization of the cup contact surface is shown in Figure 4.5b. This grid is represented as a hyperelastic shell structure parameterized by elastic stiffness $k_{cup,e}$, damping $k_{cup,d}$, and adhesion $k_{cup,a}$. For the cylindrical portion, we use a hollow cylinder-shaped particle grid discretized as $N_\theta^{cyl} \times N_r^{cyl} \times N_z^{cyl}$. This hyperelastic mesh is constructed using tetrahedral FEM elements with a Neo-Hookean energy density following [61]. The cylindrical component is parameterized by Lamé parameters μ_{cyl} and λ_{cyl}, and damping $k_{cyl,d}$. This comprehensive representation enhances the fidelity of deformation behavior under vacuum-induced stress.

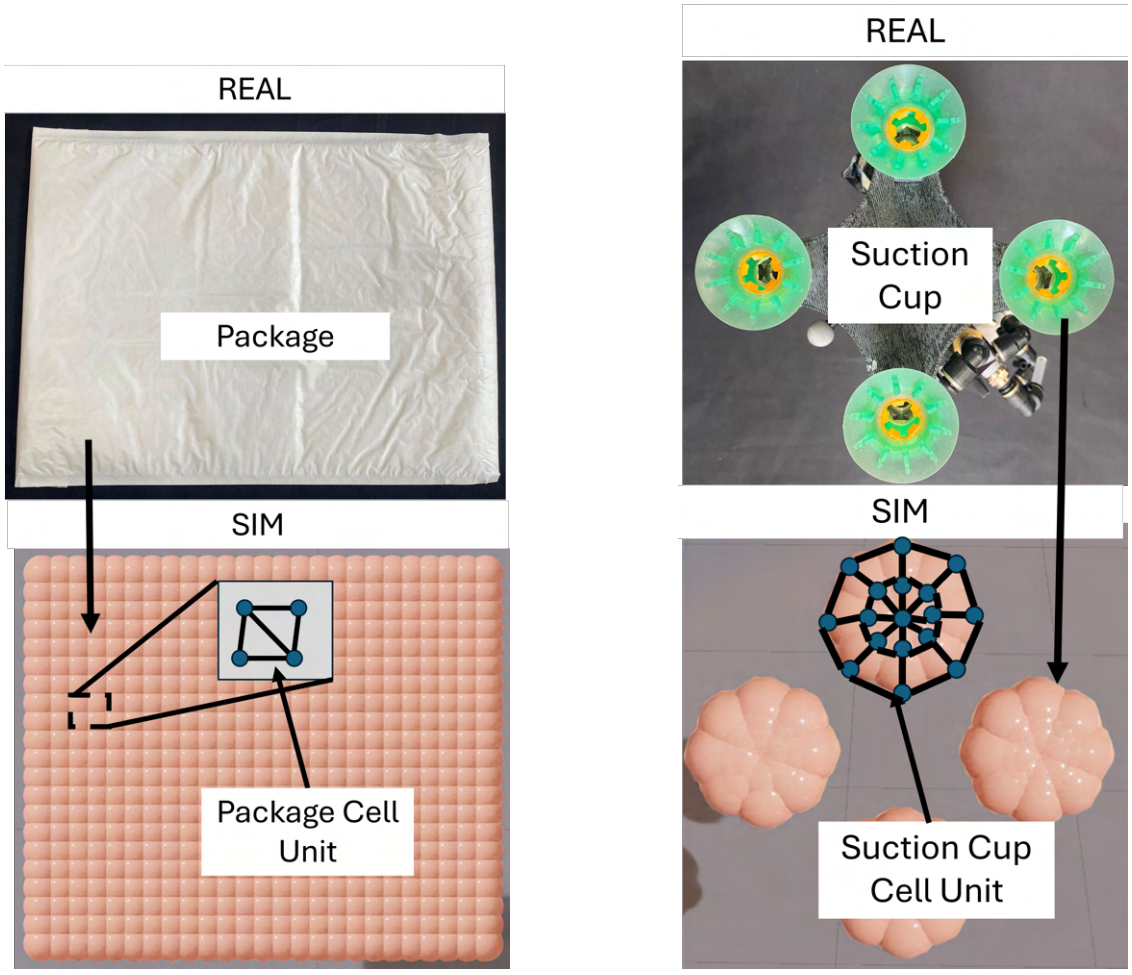
The vacuum-based adhesion phenomenon is represented by the interface parameters ${}^{\mathcal{A}}\mu_{interface}$ characterizing the interaction between the suction cups and package through a system of radially distributed springs connecting the cup's ring-shaped particles to the closest particles on the upper layer of the package. Each connection is characterized by a spring stiffness $k_{spr,e}$ and damping coefficient $k_{spr,d}$ as part of ${}^{\mathcal{A}}\mu_{interface}$. While this simplification does not explicitly model vacuum forces, it captures the essential mechanical coupling between the suction cup and package surface during manipulation.

The internal object $\mathcal{O}$ is represented at the center between the two deformable layers of $\mathcal{P}$. The interface parameters ${}^{\mathcal{P}}\mu_O$ describe the contact between $\mathcal{O}$ and the package layers, defined as soft contact characterized by contact stiffness $k_{con,e}$, damping $k_{con,d}$, and friction coefficient μ_{con}. This representation accounts for the challenges of modeling a closed package containing internal objects, particularly how these contents influence deformation behavior during manipulation.

This system representation forms the foundation for our simulation framework described in Section 4.4.3, enabling the implementation of the function $\Phi_{\Psi,\mu}$ that maps external forces F_{ext}^t and control inputs U^t to the states of the package $X_t^{\mathcal{P}}$ and object $X_t^{\mathcal{O}}$

4.4.3 Simulation Framework

Simulating a complex multi-object system with deformable components, as in this work, presents unique challenges. The model must accurately capture the nuanced deformations of a two-layered package under manipulation constraints, external forces, and gravity while also handling contact dynamics and suction-based coupling. In logistics and warehouse automation, where



(a) Package particle cell unit

(b) Suction cup unit

Figure 4.5: System Representation Overview: Each element of the package and suction cup is modeled using particles connected by edges to simulate elasticity and bending. Additionally, particle radius accounts for adhesion and compression in the deformable components.

speed and efficiency are critical, high-speed simulation is essential for seamless deployment. This demands a high-fidelity framework that supports coupled multi-object interactions with diverse material properties. High-fidelity physics information enables safe trajectory optimization and policy learning, ensuring fast, safe, and reliable package handling. Additionally, the simulation must balance fidelity and computational efficiency to enable precise system identification and execution in practical robotics applications.

Soft-body simulators offer a crucial advantage over traditional physics engines by addressing key challenges in modeling complex deformable interactions. They achieve this through two main capabilities: (1) they enable efficient optimization by providing structured representations of system dynamics, allowing for more precise policy learning, trajectory optimization, and system identification compared to sampling-based and finite-difference approaches. (2) They leverage parallel computing architectures to achieve high-speed simulations, sometimes running significantly faster than real-time for simpler systems, even in scenarios involving intricate contact dynamics and deformable objects.

When evaluating potential simulation frameworks for our application, we considered several critical factors as mentioned above: high-fidelity physics, material model flexibility, computational efficiency, and extensibility for implementing custom physics for specialized components like suction cups. After comparing several options including DiffTaichi[48], MJX[62], and Nvidia Warp[34]. We selected Warp as our simulation framework due to its comprehensive fulfillment of these requirements.

While DiffTaichi offers flexibility for modeling dynamical systems, it lacks essential features such as URDF import and kinematic chain generation, requiring extensive custom implementation compared to general-purpose physics

engines. Similarly, MJX struggles with handling large numbers of contacts efficiently, as it precomputes all potential contacts rather than resolving them dynamically. In contrast, NVIDIA Warp leverages parallel computation to accelerate simulation while providing built-in physical models and integrators that streamline development. Its support for FEM techniques makes it well-suited for our package-object system, and its interoperability with PyTorch facilitates seamless integration with learning-based frameworks. By distributing particle updates, constraint solving, and contact resolution across GPU cores, Warp enables high-speed simulation, achieving rates exceeding 60 Hz even for complex systems with over 1,000 particles.

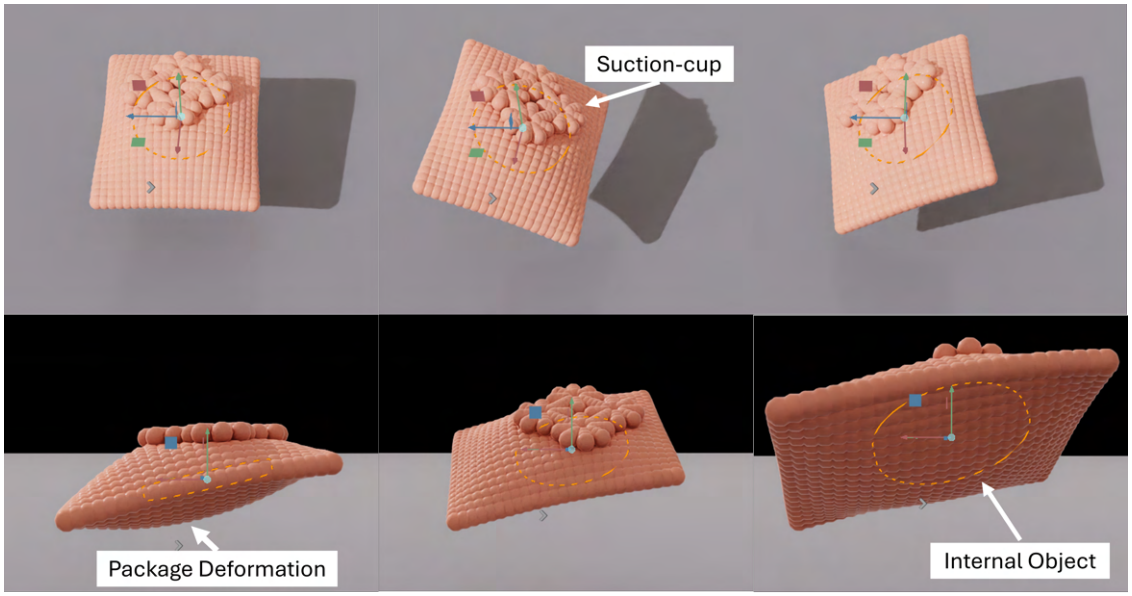


Figure 4.6: The simulation of the deformable package, suction cup, and internal object in our proposed simulation environment

Our implementation (Refer Fig. 4.6) leverages several key components of the Warp framework to realize the system representation described in Section 4.4.2. We use Warp's particle system to represent the vertices of both the package's two-layer structure and the suction cup array, constructing triangular meshes according to our earlier definitions. The force computation for

hyperelastic materials and rigid bodies is handled by Warp's built-in semi-implicit Euler integration scheme. For actuation, trajectory control inputs are converted to timestamped velocity commands for the suction cups, which we implement through a customized kernel function.

Warp's GPU-accelerated computation, efficient contact handling, and soft-body physical modeling capabilities make it an ideal framework for simulating and optimizing our complex package-object system. Its ability to efficiently model large-scale deformable interactions enables accurate system identification and rapid policy learning, overcoming the computational bottlenecks of traditional simulation methods. This approach directly addresses the challenges of robotic manipulation in logistics, ensuring both high-fidelity simulation and scalable deployment.

4.4.4 Real-to-Sim Parameter Identification

The package ($\mathcal{P}$), Object ($\mathcal{O}$), and the suction cup system are characterized by a set of parameters, (Ψ, μ), as described in Section 4.4.2, that governs specific aspects of the system's response to external forces and constraints. For instance, the parameters $k_{trie,e}$ and $k_{tri,d}$ primarily influence the deformation behavior of the package, whereas $k_{con,e}$ predominantly affects the motion of the object $\mathcal{O}$ inside the package. Recognizing these distinctions is crucial, as it informs the design of our learning framework for parameter identification. Leveraging this understanding, we develop a method to infer these parameters from real-world data (see Section 4.6).

Now, given that our simulation model represents the deformable package $\mathcal{P}$ as a system of particles with grid size $M \times N \times 2$, it is crucial to collect real-world data that provides congruent information for learning the simulation parameters. Additionally, tracking the position of the internal object $\mathcal{O}$ is essential.

Section 4.6 details our experimental setup and data-collection strategy to obtain this information. Specifically, we employ a motion capture system with reflective markers placed on the package, the object, and the suction tool to record timestamped trajectory data. Consequently, our observation set $\mathbf{O}$ consists of timestamped position data of the package particles, the object, and the robot end-effector, i.e., the suction tool's position and orientation. Notably, these markers are required only during the training phase; during execution, our model can infer this information seamlessly. Thus, no markers or motion capture system are needed for actual robot deployment (see Section 4.6).

From our data collection framework, we receive access to a dataset of timestamped trajectories $\mathcal{D}_{real} = \{\tau_1^{real}, \tau_2^{real}, \ldots, \tau_n^{real}\}$. Where,

$$\tau_n^{real} = \{(X_0^{real}, U_0^{real}), (X_1^{real}, U_1^{real}), \ldots, (X_T^{real}, U_T^{real})\}$$

is a timestamped trajectory consisting of information of system state X_t and the corresponding control input U_t at a given time instance t. Here $X_t = \{X_t^{\mathcal{P}}, X_t^{\mathcal{O}}\}$ is the state of package-object system. The control inputs U_t are the robot's end-effector position, orientation, velocity, and acceleration parameters at a given time t. Now, given this dataset $\mathcal{D}_{real}$, we use our simulation framework described in Sections. 4.4.2 and 4.4.3, where we initialize the $\mathcal{P}$ and $\mathcal{O}$ with the corresponding initial states and constraints. During initialization, we can only control U_o, i.e., the control input, and the constraints, i.e., the contact between the suction cup and the package. Under the external forces, the package-object system assumes a state X_0^{sim}. We then simulate the entire trajectory τ_i^{real} in our simulation environment and observe the corresponding states at the timestamps $t = 0 : T$. This process constructs a corresponding dataset of simulated trajectories, $\mathcal{D}_{sim}$, which we use to learn the simulation parameters Ψ. Our parameter estimation framework is detailed

in Algorithm 2, where we employ the Bayesian optimization [63] framework to optimize Ψ.

Algorithm 2: Learning Simulation Parameters via Bayesian Optimization

Data: Real-world Trajectory Dataset: $\mathcal{D}_{real}$
Input: N : Number of Optimization Iterations,
Φ: Deformable Simulation Model,
F_{ext}: External Force (Gravity),
$\mathcal{G}$: Gaussian Process Surrogate Model
Result: $\Psi^*_{\mathcal{P}}, \Psi^*_{\mathcal{O}}, {}^{\mathcal{P}}\mu^*_{\mathcal{O}}$: Optimized Simulation Parameters
Initialize: Initialize search space for $\Psi_{\mathcal{P}}$, $\Psi_{\mathcal{O}}$, ${}^{\mathcal{P}}\mu_{\mathcal{O}}$
Fit initial Gaussian Process $\mathcal{G}$ using random samples
$i = 0$
while $i \leq N$ **do**
  for $\tau^{real}_{batch} \in \mathcal{D}_{real}$ **do**
    Sample candidate parameters $\Psi_{\mathcal{P}}$, $\Psi_{\mathcal{O}}$, ${}^{\mathcal{P}}\mu_{\mathcal{O}}$ from acquisition function
    $X^{sim}_{batch} = \Phi(\Psi_{\mathcal{P}}, \Psi_{\mathcal{O}}, {}^{\mathcal{P}}\mu_{\mathcal{O}}, U^{real}_{batch})$
    $\mathcal{L}^{batch}_{\Psi_{\mathcal{P}}} = CD(X^{real}_{\mathcal{P}_{batch}} - X^{sim}_{\mathcal{P}_{batch}})$
    $\mathcal{L}^{batch}_{\Psi_{\mathcal{O}}, {}^{\mathcal{P}}\mu_{\mathcal{O}}} = ||X^{real}_{\mathcal{O}_{batch}} - X^{sim}_{\mathcal{O}_{batch}}||^2_2$
    $\mathcal{L}_{batch} = \mathcal{L}^{batch}_{\Psi_{\mathcal{P}}} + \mathcal{L}^{batch}_{\Psi_{\mathcal{O}}, {}^{\mathcal{P}}\mu_{\mathcal{O}}}$
    Update $\mathcal{G}$ with new observation $(\Psi_{\mathcal{P}}, \Psi_{\mathcal{O}}, {}^{\mathcal{P}}\mu_{\mathcal{O}}, \mathcal{L}_{batch})$
  Select the next parameters by optimizing the acquisition function over $\mathcal{G}$
  if $\mathcal{L}_{batch} < \mathcal{L}_{min}$ **then**
    $\mathcal{L}_{min} = \mathcal{L}_{batch}$
    $\Psi^*_{\mathcal{P}} \leftarrow \Psi_{\mathcal{P}}$
    $\Psi^*_{\mathcal{O}} \leftarrow \Psi_{\mathcal{O}}$
    ${}^{\mathcal{P}}\mu^*_{\mathcal{O}} \leftarrow {}^{\mathcal{P}}\mu_{\mathcal{O}}$
  i++

As discussed in Sections 4.4.1 and 4.4.2, each parameter in the full parameter set Ψ contributes uniquely to the system's dynamics. To ensure our optimization framework captures this structured influence, we design distinct loss functions that appropriately guide the optimization process. Specifically, for optimizing the package parameters, we employ the Chamfer loss, defined as follows:

$$\mathcal{L}_{\Psi_{\mathcal{P}}} = \frac{1}{2|\mathcal{X}_{\mathcal{P}}^{real}|} \sum_{x_i^{real} \in \mathcal{X}_{\mathcal{P}}^{real}} min||x_i^{real} - \mathcal{X}_{\mathcal{P}}^{sim}||_2 + \frac{1}{2|\mathcal{X}_{\mathcal{P}}^{sim}|} \sum_{x_j^{sim} \in \mathcal{X}_{\mathcal{P}}^{sim}} min||x_j^{sim} - \mathcal{X}_{\mathcal{P}}^{real}||_2 \quad (4.2)$$

Chamfer loss is well-suited for training because it directly minimizes the discrepancy between simulated and real particle positions, ensuring that the simulated package deformation closely matches real-world behavior. By penalizing the distance between corresponding particle sets in a trajectory pair $(\tau_i^{sim}, \tau_i^{real})$, Chamfer loss enforces fine-grained spatial alignment without requiring explicit point correspondences. This makes it particularly effective for capturing complex deformations, as it naturally handles variations in particle distribution while maintaining robustness to minor noise in the dataset.

Similarly, for optimizing the object parameters $\Psi_{\mathcal{O}}$ and the interface parameters ${}^{\mathcal{P}}\mu_{\mathcal{O}}$, we utilize the Mean Square Error (MSE) loss function to effectively guide the optimization of these parameters:

$$\mathcal{L}_{\Psi_{\mathcal{O}}, {}^{\mathcal{P}}\mu_{\mathcal{O}}} = ||X_{\mathcal{O}}^{real} - X_{\mathcal{O}}^{sim}||_2^2 \quad (4.3)$$

We specifically rely on $\mathcal{O}$'s state observations for optimizing the interface parameters, as our experiments indicate that $\mathcal{O}$'s position is primarily influenced by these parameters, which govern contact forces and friction. Thus, our optimization process jointly refines the simulation parameters to enhance accuracy. The overall loss function, $\mathcal{L}_{\mathcal{D}}$, as defined in the Algorithm. 2 is the weighted sum of the individual loss terms in Eqs. 4.2 and 4.3, ensuring that each set of parameters is optimized accordingly. This structured approach allows us to learn simulation parameters that accurately reproduce the dynamics of our

multi-object system. Furthermore, as outlined in Algorithm 2, we optimize the parameters using the Bayesian optimization framework over N epochs. Bayesian optimization excels in efficiently exploring the parameter space, especially when the objective function is complex, noisy, or expensive to evaluate. A key component of this framework is the acquisition function, which guides the search for optimal parameters by balancing exploration and exploitation. The acquisition function evaluates the utility of sampling a particular set of parameters based on the current probabilistic model of the objective function. By selecting the parameters that maximize the acquisition function, we ensure that each iteration improves our understanding of the parameter space and helps identify regions with high performance.

In our approach, the acquisition function helps us prioritize promising regions for parameter updates, ensuring that we avoid redundant evaluations and focus computational resources on the most informative areas. This leads to stable convergence with fewer iterations, making the optimization process computationally efficient while maintaining high-quality results. The combination of the probabilistic model and the acquisition function makes Bayesian optimization particularly powerful for parameter estimation tasks, enabling robust and reliable learning even with limited data.

4.5 Manipulation Policy Learning

The proposed simulation model is designed to support both online planning and offline learning of safe and efficient manipulation policies. Prior work [64] has demonstrated the importance of simulators in enhancing the learning of manipulation policies, particularly for complex robotic tasks, by providing valuable inductive biases. Our framework leverages these biases to enable both data-driven policy learning and model-based trajectory optimization. The

ability to capture high-frame-rate deformation information further enhances the model's applicability to online planning, offering a powerful solution for tackling dynamic manipulation tasks with precision and efficiency.

One of the standout features of our approach is the use of NVIDIA Warp, a CUDA-based framework optimized for GPU computation. Massive parallelization is essential for reinforcement learning (RL) applications, as training requires extensive exploration across diverse scenarios. By leveraging GPU acceleration, we simulate multiple training instances simultaneously, drastically reducing policy optimization time. This level of parallelism is crucial for large-scale learning, where sample efficiency and computational speed are fundamental constraints.

In high-mix, high-volume environments such as logistics and warehouse automation, where speed and efficiency are crucial, this parallelism marks a significant advancement in policy learning. The ability to train and refine policies at scale enables the rapid deployment of learned behaviors in real-world applications. While the primary focus of this work is on our simulation model and parameter identification framework, we also highlight the practical applicability of our differentiable simulation system by implementing an RL Gym [65] environment. This open platform enables the robotics community to train their own RL-based manipulation policies efficiently and in a scalable manner. Additionally, our simulator facilitates model-predictive control (MPC), enabling the optimization of robot trajectories for safe and efficient motion generation. This capability is crucial for handling deformable objects in real-world applications, ensuring that the resulting motions adhere to safety constraints while maximizing efficiency.

As a demonstration, we train a Proximal Policy Optimization (PPO)-based policy for package transport. The primary objective of this policy is to minimize the internal object movement within the package and reduce its deformation curvature near the suction cup to ensure stable grasping. This example highlights how our simulator enables large-scale policy learning while maintaining physically meaningful constraints that are crucial for real-world deployment.

By integrating high-fidelity physics with reinforcement learning and model-based control, our approach bridges the gap between soft-body simulation and practical robotic policy deployment. The ability to train and optimize policies at scale makes it a valuable tool for advancing deformable object manipulation in industrial settings.

4.6 Experiments and Data Collection

4.6.1 Experimental Design

Our experimental setup consists of a 7-DOF KUKA LBR iiwa robot, chosen for its impedance control capabilities, which facilitate the handling of deformable packages during trials. The robot is equipped with a four-suction cup tool connected to a 9 CFM vacuum pump with a 1.5-micron ultimate pressure via a manifold (see Fig. 4.7). Suction control is handled by a relay operated through an Arduino microcontroller. To systematically study the effects of robot motion on both the package and its internal object, we designed a trajectory that captures a range of dynamic movements relevant to real-world logistical settings. The trajectory simulates a pick-and-place operation, where a package is lifted from a conveyor belt and placed into a bin. The final placement orientation varies from horizontal (0^o) to near-vertical, with

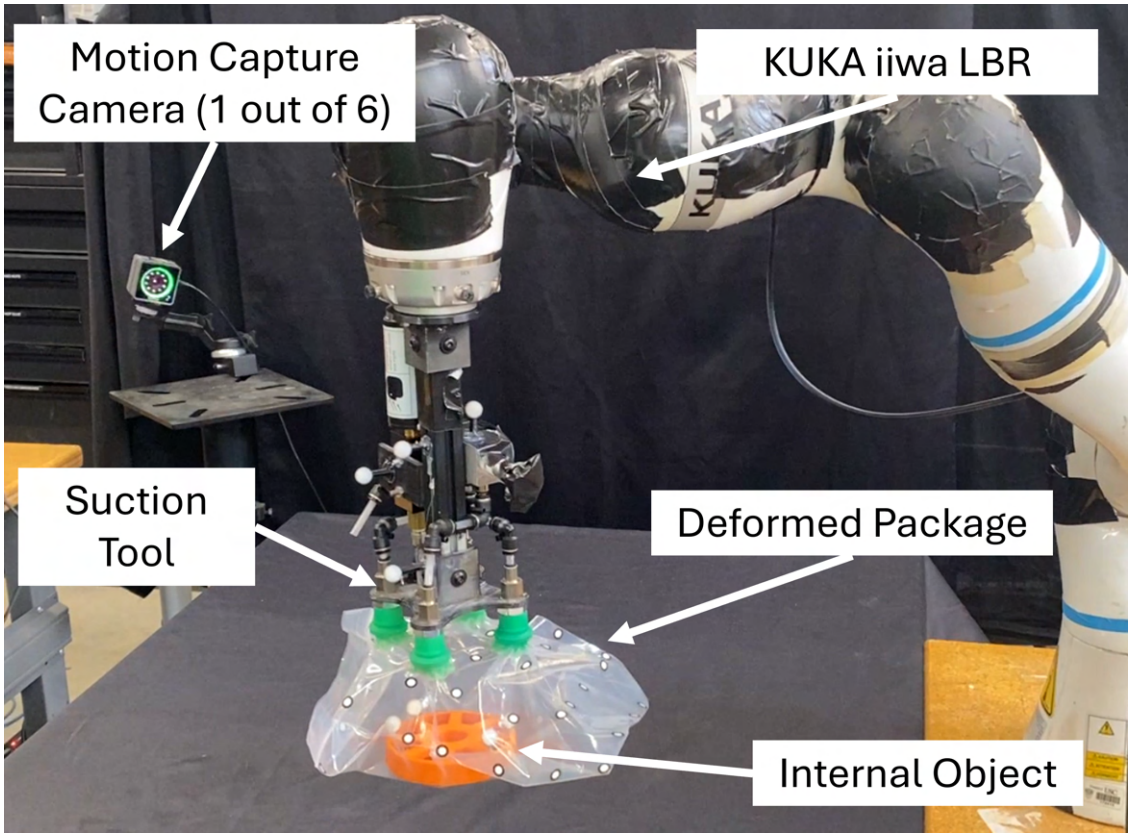


Figure 4.7: The experimental setup with a 7 DOF kuka LBR iiwa Robot. A suction tool is attached to the robot that is connected to a 9 CFM vacuum pump. The suction cups selected in this task are rated to handle such objects specifically

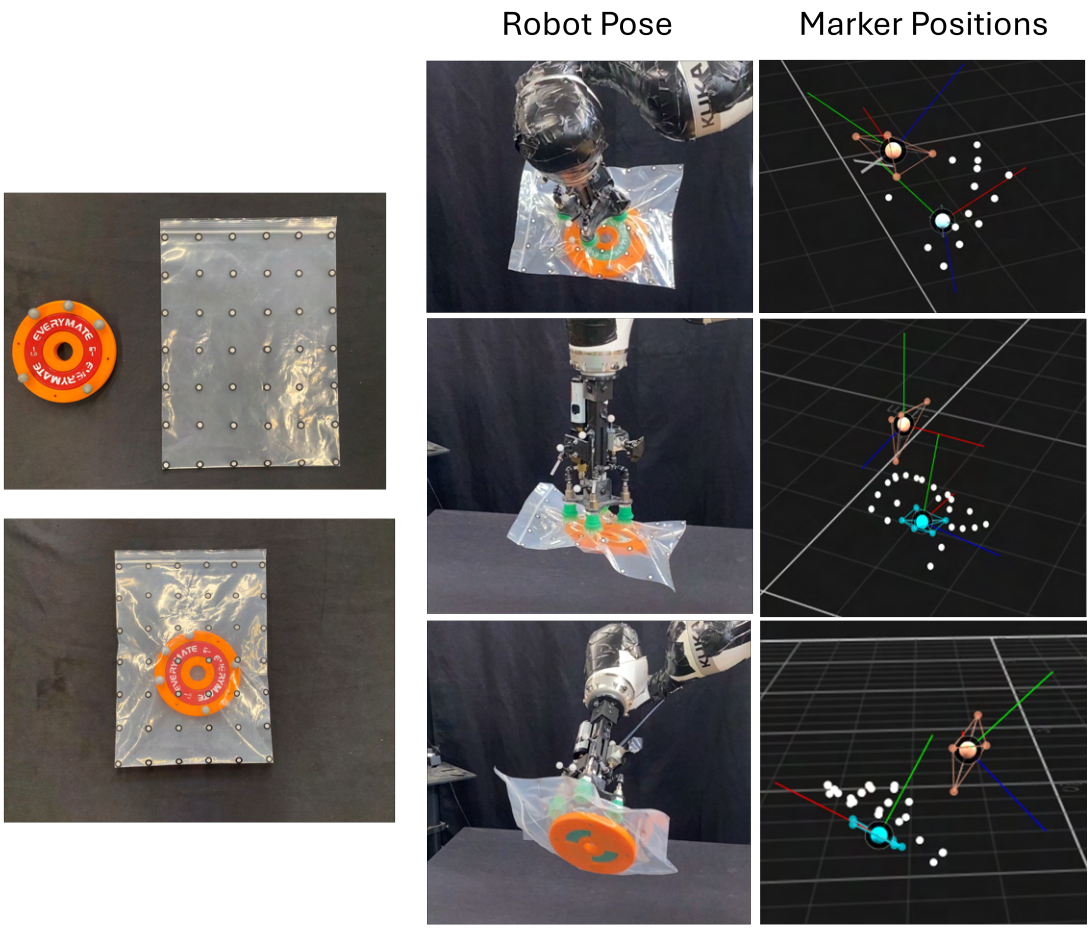
three distinct angles: 30^o (low), 45^o (medium), and 60^o (high). The trajectory consists of three primary motions: a linear motion, followed by a circular motion, and an orientation change at the final placement stage. To further analyze motion effects, we vary joint velocity and acceleration, categorizing each into three levels: low, medium, and high. Each of these trajectory variations reflects the changes in U_{real}. The test package measures 10×12 inches and contains a 3D-printed internal disc of 160 mm diameter. The internal mass is also varied across three levels: 0.5 lbs (low), 0.75 lbs (medium), and 1 lbs (high). We collect a total of 27 distinct timestamped trajectory data points corresponding to 1 package, 3 placement orientations, 3 mass variations, and 3 velocity conditions. Each trajectory lasts an average of 6 seconds, and we neglected the failure scenarios since the focus was mainly on the dynamics. Our data acquisition framework operates at 250 Hz, providing high-resolution motion data that captures detailed package deformation over time. Despite the limited number of 24 trajectories after filtering the failure scenarios, the high temporal resolution of the data ensures sufficient detail for learning the parameters effectively.

A key component of our data collection framework is a 6-camera OptiTrack motion capture system, which is primarily used to track the deformation of the package. OptiTrack's Motive 2 software streams data from these cameras, allowing us to track the positions of individual reflective markers or defined rigid bodies and compute their corresponding $\mathbb{SE}(3)$ poses with a submillimeter accuracy in tracking. As discussed in Section 4.4.2, our simulation represents the package $\mathcal{P}$ as a particle-based system. To emulate this in the real world, we affix N 2D reflective markers to the package, effectively creating a particle-based representation. The (x, y, z) positions of these markers are tracked in Motive 2, providing high-fidelity deformation data. Thus, the state of the package at a time instance t is represented by the position of these markers

$X_{\mathcal{P}} = [x_i, y_i, z_i]_{i=1}^{N}$. To ensure accurate tracking of both the package and its internal object, we make a design choice to use transparent packages. This allows us to track the internal object without additional engineering efforts for time synchronization. We acknowledge that real-world logistics settings typically involve non-transparent packages. However, our primary objective in this work is to establish a proof of concept demonstrating that our simulation framework can handle such systems. For non-transparent packages, an alternative approach involves using an external force-torque sensor to estimate the center of mass (CoM) of the internal object. This would require additional engineering, particularly for time synchronization and ensuring reliable CoM estimation. In this study, we opt for transparent packages to simplify data collection while maintaining the visibility of the internal object. For tracking the internal object, we use spherical reflective markers and define it as a rigid body in Motive 2 (see Fig. 4.8). This setup allows us to capture both position and orientation relative to the motion capture system's base frame, ensuring accurate and consistent tracking. For simplicity, we only consider the Cartesian position of the object, and thus $X_{\mathcal{O}} = [x_{\mathcal{O}}, y_{\mathcal{O}}, z_{\mathcal{O}}]$.

4.6.2 Processing of Motion Capture Data

The timestamped data collected from the motion capture system is inherently noisy, primarily due to marker occlusions and the presence of spurious reflections from other reflective surfaces in the camera's field of view. These spurious reflections can result in erroneous marker detections, which we mitigate through precise calibration. Specifically, during data collection, we execute predefined trajectories and manually mask out any spurious markers that appear. This ensures that reflective surfaces incorrectly tagged as markers are eliminated from the dataset. To further improve robustness, we enforce a minimum difference of 10 mm between the inter-marker distance of the 2D markers



(a) The Package and Object with reflective markers

(b) The visualization of markers in Motive 2 Software

Figure 4.8: The package and the corresponding inside object with the motion capture markers. We also illustrate a sample trajectory that is collected from the motion capture. We can observe that at certain time instances, the package markers can disappear, thus entailing the adoption of advanced data-processing methods.

on the package and the spherical markers on the internal object. This prevents marker misidentification, where the system might erroneously associate one marker's identity with another.

A key challenge in post-processing arises due to marker occlusion, where markers temporarily disappear from the tracking system. Since OptiTrack's Motive 2 software does not provide marker identification and tracking across trajectory for deformable bodies,[1] we introduce a comprehensive filtering methodology to estimate missing marker positions:

1. Noise Reduction: We first apply a low-pass Butterworth filter to the timestamped marker trajectories to eliminate high-frequency noise $X_f(t) = \frac{1}{\sqrt{1+\left(\frac{f_c}{f}\right)^{2n}}} X(t)$. where $X_f(t)$ is the filtered signal, $X(t)$ is the raw marker position, f_c is the cutoff frequency, and n is the filter order. The band gain is set to 1.

2. Hungarian Matching-Based Reassignment: If a marker disappears, we check whether it has been reassigned a new ID by using the Hungarian algorithm for optimal matching. Specifically, we solve for the best match between the missing marker p_i and the detected markers at time t, $\mathcal{M}(t)$, based on their available neighbor information. The assignment cost is minimized by:$\hat{p}_i = \arg\min_{p_j \in \mathcal{M}(t)} \|p_j - \hat{p}_i^{(t)}\|$, where $\hat{p}_i^{(t)}$ represents the set of neighbor positions associated with the missing marker at the current time step. If the cost is below a predefined threshold ϵ, the marker is reassigned based on the optimal matching result. If no neighbors are available, we discard the position information by assigning it a zero-mask

[1] This functionality is available in Motive 3, but for the scale of deformations in our setup, it remains intractable.

3. Velocity-Based Interpolation: If reassignment fails, we estimate the missing position using first-order derivative-based interpolation under the assumption of local smoothness: $p_i(t) = p_i(t-1) + v_i(t-1) \cdot \Delta t$, where, $v_i(t-1) = \frac{p_i(t-1) - p_i(t-2)}{\Delta t}$ is the estimated velocity of the marker.
4. 1D Interpolation for Long Occlusions: If data is missing for a longer window, we perform 1D linear interpolation using adjacent marker positions, defined by $p_i(t) = p_i(t_1) + \frac{t-t_1}{t_2-t_1}(p_i(t_2) - p_i(t_1))$. where t_1 and t_2 are the nearest available timestamps.

If none of the above methods successfully reconstruct the missing marker positions, we discard the affected timestamps from our dataset. Ultimately, our filtering strategy enables us to retain 98% of the timestamped data from the collected dataset for downstream parameter learning while ensuring high-fidelity deformation tracking.

4.7 Results

In our experiments, we aimed to evaluate key aspects of our parameter learning framework to assess its effectiveness. Specifically, we sought to answer the following questions:

1. How accurately does our simulation framework predict real-world package deformation and internal object dynamics?
2. To what extent does our method generalize to variations in control parameters and object mass?
3. How well does our proposed simulation model perform on unseen scenarios in a held-out test dataset?

With these evaluations, we aim to provide a comprehensive assessment of our simulation framework's fidelity, robustness, and generalization capabilities,

highlighting its potential for real-world deployment in robotic manipulation tasks.

4.7.1 Parameter Estimation Results

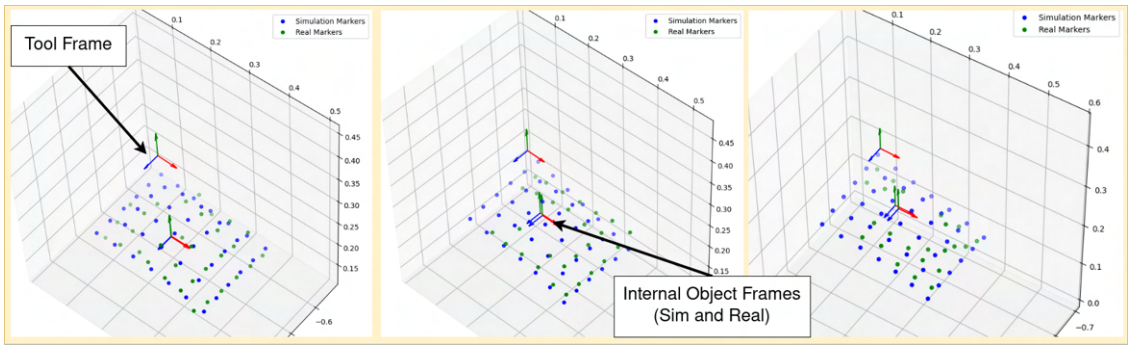


Figure 4.9: Qualitative comparison of simulation vs. real-world predictions. Blue markers represent simulated positions at a given time step, while green markers indicate corresponding real-world positions from our test dataset. The coordinate axes near the markers denote simulated and real object positions, with the topmost axis representing the tool frame.

The dataset of 24 trajectories was preprocessed using our data-processing pipeline, as described in Section 4.6.2. We employed a subset sampling strategy to divide the data into training and testing sets to ensure a fair and representative evaluation. Specifically, for the test set, we aimed to assess the generalization of our method on unseen cases. To achieve this, we selected trajectories with medium velocity, medium acceleration, and medium orientation for a given package and mass combination. These conditions fall within the overall distribution of the training data but are not explicitly included, allowing us to fairly evaluate generalization. Based on this strategy, we allocated four trajectories for testing. This resulted in a final split of 20 trajectories for training and 4 for testing. All data was normalized using a min-max normalization approach before training. We implemented the Bayesian optimization framework with Bayes-Opt library [66]. We trained our model for 100 epochs

and reported the best-performing parameter values on the training data in Table 4.1.

Dataset	Object			Package		
	MSE ↓ (m)			CD ↓ (m)		
	mean	max	std	mean	max	std
Train	0.0006	0.003	0.0006	0.032	0.036	0.003
Test	0.0008	0.001	0.0003	0.033	0.038	0.003

Table 4.1: The performance of the parameter estimation framework. The error values reported here are the chamfer distance of package state and mean-square error in object state prediction in meters.

Table 4.1 highlights the effectiveness of our parameter estimation framework in accurately predicting both the package and internal object positions. The error values reported confirm that our simulation model effectively captures the intricate interactions between the deformable package, the internal object, and the external constraints imposed by the robotic manipulator. The chamfer distance estimation for the package loss, which is 0.033 m, demonstrates that our model successfully captures the package deformation [67], especially for such a complex dynamic deformable object. Additionally, the mean error of 0.0006 m for the internal object shows that the interface contact and object parameters were appropriately learned. These results validate the learned parameters' accuracy and fidelity, suggesting our approach generalizes well to various control inputs and package configurations (Refer Fig. 4.9).

4.7.2 Optimization Performance Analysis

A key premise of our framework is that accurately modeling the deformable package system requires selecting optimal simulation parameters. If the loss landscape were relatively flat, randomly chosen parameters would yield similar results, reducing the need for optimization. However, in practice, random initialization often leads to poor predictions, as the loss landscape contains local

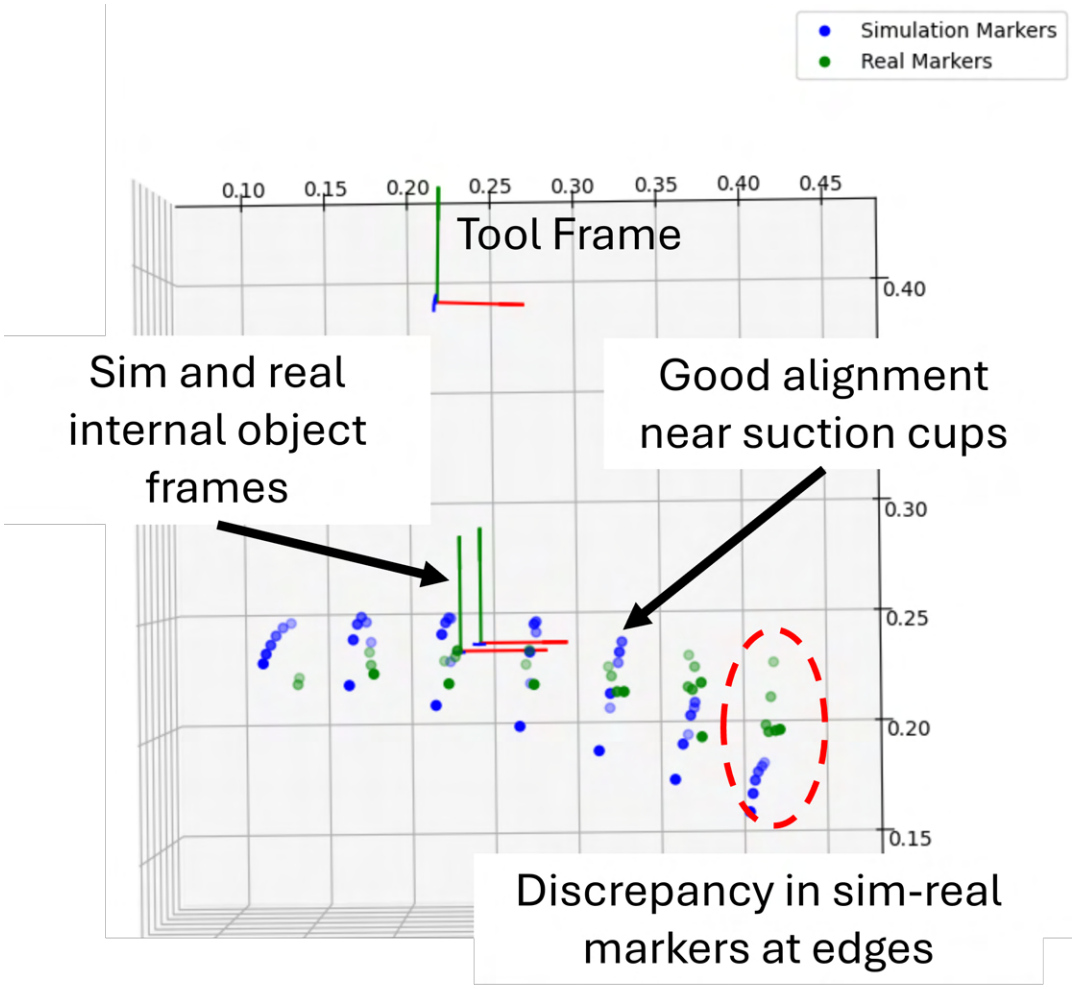


Figure 4.10: The marker prediction discrepancies are primarily at the edges, while accuracy remains high near the suction cups, a critical region for failure detection. This demonstrates that our parameter estimation is well-suited for computing safe and efficient trajectories, as most errors do not impact performance.

minima that significantly impact performance. Our Bayesian optimizer effectively identifies an optimal set of parameters that enables accurate prediction of the package and internal object states, which are critical for learning safe and efficient manipulation policies.

To validate this, we perturb each parameter by a percentage of its optimized value and analyze its effect on prediction accuracy. Our sensitivity analysis shows that perturbations within 20% of the optimized value maintain stable performance with minimal impact on accuracy. However, beyond this threshold, the simulation becomes unstable, requiring reduced time steps for integration, which significantly slows down the overall simulation speed.

To validate that our prediction accuracy is sufficient for computing safe and efficient manipulation policies and trajectories, we analyze the simulation's performance. As shown in Fig. 4.10, the primary discrepancies in marker position predictions occur at the package edges, while markers near the suction cup engagement points, closer to the package center, maintain high accuracy (< 0.01m). This demonstrates that despite a chamfer distance of 0.03m, our simulation model remains effective for generating reliable and safe trajectories.

4.7.3 Simulation Performance Metrics

A critical aspect of our simulation framework is evaluating its computational efficiency in handling complex multi-object interactions. To ensure high fidelity while maintaining acceptable prediction accuracy, we systematically vary the number of particles representing the package and analyze its impact on both prediction performance and simulation frame rate. As shown in Table 4.2, increasing the particle count results in a comparable simulation frame rate but introduces a trade-off between computational efficiency and accuracy. Our findings suggest that practitioners should fine-tune the particle count based

on their specific accuracy and performance requirements. Another key consideration is hardware memory limitations. All our evaluations are conducted on an NVIDIA GeForce RTX 3060 12 GB GPU, where instantiating more than 1612 particles leads to memory overload. Furthermore, Fig. 4.11 demonstrates that reducing the particle count can lead to inaccurate package deformation, ultimately degrading prediction performance, which aligns with our ablation studies.

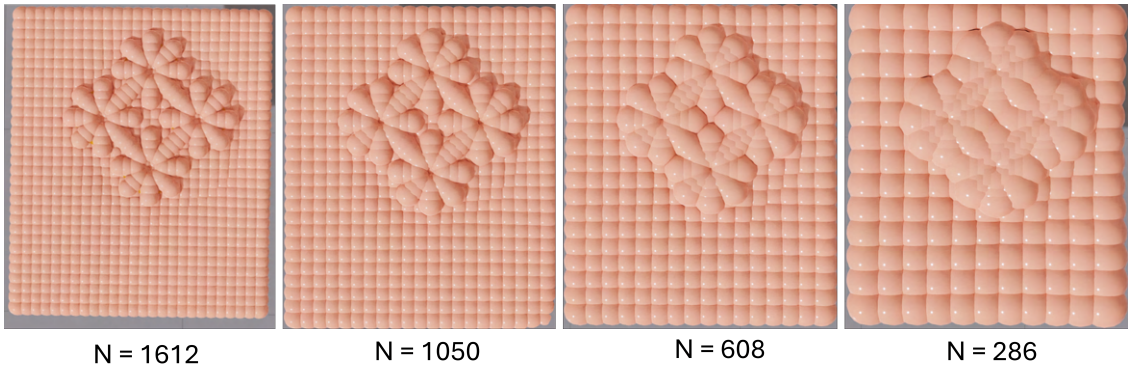


Figure 4.11: Difference in package shape in simulation with reducing the number of particles. We can see that fewer particles can lead to poor performance in capturing the package deformation.

Num Particles	**Frame Rate (FPS)** ↑	**Package CD (m)** ↓	**Object MSE (m)** ↓
1612	9	0.032	0.0006
1050	10	0.033	0.0007
608	10	0.035	0.0008
286	10	0.037	0.001

Table 4.2: Effect of particle count on simulation parameters and frame rate. The prediction performance depends on the number of particles. Our simulation runs in real-time, meaning it takes t seconds to simulate a trajectory of duration t seconds. We can see that increasing the number of particles improves the loss without affecting the FPS. However, FPS remains the same. Increasing the number of particles can cause issues with available GPU memory.

4.8 Discussions

While our primary focus in this work is on developing a deformable simulation model of the package-object system, its broader impact on package handling efficiency is crucial to consider. The proposed model significantly enhances the balance between safety and efficiency across key manipulation stages: pick-up, transport, and placement. Each of these steps demands precise trajectory computation to minimize failures that could result in package damage. In our prior [59], we explored trajectory optimization for fully filled packages, but it did not capture the complexities introduced by partially filled deformable packages with a moving object. In this work, our proposed multi-object model addresses this gap by providing a more adaptable and accurate simulation framework, enabling robots to handle a wider range of package conditions with improved reliability and efficiency in computation that translates to process efficiency.

Our GPU-based simulation framework implementation enables more efficient training times, with an observed 30% reduction compared to non-GPU-based models. By accelerating trajectory optimization and improving convergence speed, our approach reduces the number of iterations required to find an optimal solution. This computational improvement can potentially decrease overall process time by 15% [59]. Furthermore, the ability to explicitly model failure cases [5] enhances robustness, with a potential failure reduction of 20%. Internal object movement within the package plays a key role in suction-based grasp failures, and our approach allows for precise simulation and tracking of these dynamics. By optimizing trajectories to minimize internal object motion, we reduce instability and the risk of package damage. Unlike conventional rigid-body simulators, our approach dynamically adapts to variations in package shape, weight, and contents, allowing robots to respond to disturbances

up to 2-5 times faster, ensuring more reliable real-world deployment without extensive manual tuning.

Although our simulation framework is designed to be differentiable, maintaining smooth and stable gradients within the NVIDIA Warp framework proves challenging. This difficulty arises from the inherent complexities of simulating deformable objects, where small numerical or algorithmic inconsistencies can introduce noisy gradients, particularly when accounting for non-linearities, contact dynamics, and material deformations. The gradients may become unstable or imprecise in high-dimensional, dynamic environments, where the relationships between simulation parameters and physical behaviors are intricate. Such noisy gradients can degrade the performance of gradient-based optimization methods, leading to suboptimal solutions or slower convergence due to erratic updates.

In contrast, Bayesian optimization is particularly well-suited for these types of scenarios. Unlike gradient-based methods that depend on precise gradient information, Bayesian optimization treats the objective function as a black box and constructs a probabilistic model. This allows it to explore the parameter space effectively without relying on gradients, making it more resilient to noisy or unreliable signals. By incorporating prior knowledge and carefully balancing exploration with exploitation, Bayesian optimization can navigate the parameter space efficiently, even in the presence of noise or limited data. Furthermore, since Bayesian optimization updates its model iteratively based on fewer but more informative evaluations, it mitigates the impact of noisy gradients, leading to more stable optimization. Given these advantages, we have opted to rely on Bayesian optimization rather than exploiting the differentiability of our simulation framework in this work.

4.9 Summary

In this chapter, we presented a high-fidelity simulation framework tailored for modeling shell-like deformable objects, specifically, deformable packages containing internal objects that exhibit dynamic motion. Unlike traditional approaches focused on 1D or 2D deformable structures, this work tackled the challenges unique to packages encountered in warehouse and logistics automation, where internal mass movement significantly complicates external deformation behavior.

A physics-based, GPU-accelerated simulation environment that captures both the elastic behavior of the package shell and the coupled dynamics of the internal object is introduced. This framework enables accurate modeling of complex phenomena critical for robotic manipulation, including shape changes under suction contact and dynamic shifts in the center of mass during transport. To ensure that simulated behaviors faithfully match real-world physics, a Bayesian optimization approach for simulation parameter learning, grounded in motion-capture deformation data, is proposed. Furthermore, a structured loss formulation is developed that provides more informative gradients during parameter optimization, improving the robustness and accuracy of the learned simulation models.

In addition to modeling contributions, we created a comprehensive experimental testbed for real-world data collection and parameter identification, and developed a parallelized Gym environment to enable scalable policy training for robotic manipulation of deformable packages.

Extensive evaluations validated that the proposed simulation framework generalizes well to real-world scenarios, accurately capturing both external deformations and internal dynamic effects. By enabling policy learning in a physics-informed environment, this work lays the foundation for safer, more

efficient, and more adaptive robotic handling of deformable packages in semi-structured industrial environments.

Overall, this chapter demonstrates that physics-informed, structure-aware simulation modeling provides a powerful tool for tackling one of the most challenging frontiers in robotic manipulation: the safe, reliable handling of deformable packages under uncertainty. This work underscores a central dissertation theme—that combining physical priors with learning-based approaches is key to enabling the next generation of intelligent, adaptable robotic systems for real-world industrial applications.

Chapter 5

Graph-Based Neural Dynamics of Shell-Like Deformable Objects

5.1 Introduction

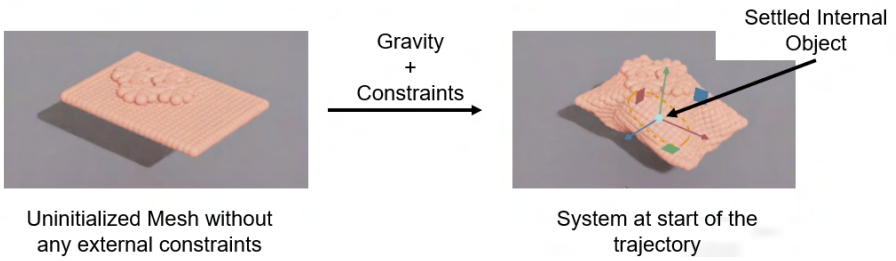


Figure 5.1: Accurate initialization in the FEM-based simulation can be challenging. Since the simulator solves for the complex interactions, the initial state of the package and the object can have uncertainty associated with them, which can propagate as an error for a given trajectory.

Chapter 4 introduced a finite element method (FEM)-based approach for simulating shell-like deformable objects using a particle-based thin-shell representation. While the FEM-based method offers advantages such as high-fidelity modeling, minimal data requirements, and the capability to manage complex interactions, it heavily depends on semi-implicit Euler integration for solving the forward dynamics. The semi-implicit Euler integration method produces

time-discretized approximate solutions for the forward dynamics of the system's state $(X_{t+1}, \dot{X}_{t+1})$. The stability of this integration method is closely tied to the chosen integrator time-step (Δt), necessitating a delicate balance between stability and computational efficiency. Small time-steps enhance stability but at the expense of increased simulation times. Moreover, determining an optimal time-step is complicated further by its sensitivity to simulation parameters and initial conditions, which introduces additional complexity to FEM-based modeling.

Another significant challenge with FEM-based simulations lies in accurately initializing the system state, particularly for composite configurations involving a thin-shell deformable package and an internal rigid object (refer to Chapter 4). The system's settled state is highly sensitive to simulation parameters and the initial positioning of mesh entities, leading to initialization uncertainties that propagate through subsequent simulation steps, causing discrepancies between simulated and real-world behaviors (Refer to Fig. 5.1). While methods exist to alleviate this issue, such as pre-learning static deformation before applying external dynamics, they introduce additional complexity to the optimization process and data collection.

Furthermore, despite successfully demonstrating FEM's capability in modeling various materials, finding feasible solutions under complex constraint scenarios remains challenging. The limited number of FEM parameters (only 11) constrains the accurate representation of intricate interactions among rigid and deformable components within the system. FEM effectively captures the overall shape of the package but tends to fall short in modeling the subtle deformation details and surface curvatures induced by external control inputs [59].

Regardless of all these challenges, retaining a structured particle-based mesh representation is advantageous due to its capacity to effectively encode spatial distributions and relationships. However, what one needs is to overcome the shortcomings of the semi-implicit Euler integration, complex interactions (rigid-deformable), and uncertainty in initialization. To achieve this, we turn our attention to Graph-based Neural Dynamics (GND) [68] that exhibit the desirable properties for simulating complex deformable objects. GND models represent objects as graph nodes connected by edges encoding spatial relationships, and leverage multi-layer perceptrons (MLPs) to predict forward system dynamics. GNDs have recently gained prominence due to their robust capability to simulate large-scale and complex deformations, alongside significantly easier initialization, as the initial node positions can be directly captured from observations.

This chapter extends the graph-based neural dynamics framework to effectively manage intricate rigid-deformable interactions by explicitly grounding them in material parameters. By embedding these physical parameters into our model, we incorporate critical physics-based priors that enhance predictive accuracy. Additionally, we exploit the natural shape-retention property of deformable packages (see Section 5.5.5) by proposing a specialized loss function designed to capture realistic deformation characteristics. To summarize, the key contributions of this chapter are:

1. Development of a graph-based neural dynamics model capable of accurately simulating thin-shell deformable objects interacting with internal rigid entities and deformable suction-cup attachments, while simultaneously predicting internal object dynamics.

2. Introduction of a training methodology that explicitly embeds material parameters and leverages a shape-retention loss, ensuring alignment of simulated dynamics with observed real-world deformation behaviors.

5.2 Related Works

Graph Neural Networks (GNNs) have rapidly emerged as a powerful paradigm for learning the dynamics of physical systems that can be naturally cast as graphs, where nodes represent particles or rigid bodies and edges encode their interactions [69]. Early demonstrations showed that GNN-based models could accurately predict the evolution of rigid-body ensembles and granular media by learning force-like message-passing rules directly from data [70]. Subsequent work extended these ideas to robotic manipulation tasks, using graph-based dynamics models to plan and control interactions with deformable rope, cloth, and soft-body objects [71–74].

Despite these successes, most existing GNN-based simulators treat material properties as either fixed global constants or ignore them altogether, limiting their ability to generalize across objects with varying stiffness, mass, or damping. The recent work by the authors of [68] represents one of the first efforts to incorporate material parameters into a GNN dynamics model, albeit focused on quasi-static, single-object deformations. To our knowledge, no prior study has addressed multi-component systems that combine both rigid-deformable and deformable-deformable interactions, nor has any applied graph-based dynamics to thin-shell, package-like objects in a robotic manipulation context.

This chapter fills these gaps by proposing a physics-informed GNN framework that (i) embeds per-node and per-edge material parameters to capture heterogeneous deformability; (ii) models a coupled suction-cup, thin-shell package, and internal object system, explicitly representing rigid–deformable contact

and suction constraints; and (iii) adds a shape-retention regularizer grounded in physical observations. By doing so, our work extends the applicability of graph-based neural dynamics from rigid or granular media to complex shell-like deformable objects with real-world robot control applications.

5.3 Graph-based Representation

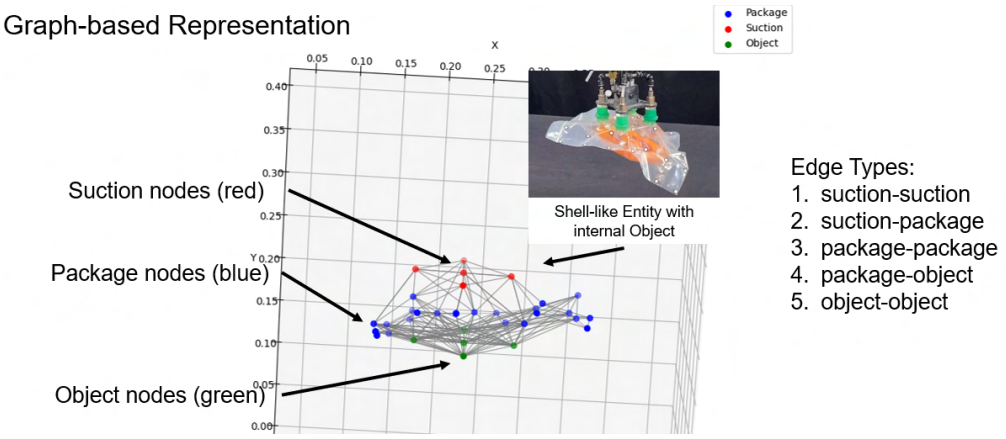


Figure 5.2: Graph-based representation of the suction-package-object system. Nodes represent the deformable suction cup, thin-shell package, and rigid internal object. Edges encode both intra-entity relationships (within the suction cup or package) and inter-entity interactions (between suction cup–package and package–internal object), effectively capturing the complex spatial and dynamic relationships within the system.

The problem formulation in this chapter closely mirrors the one defined previously in Chapter 4, Section 4.3. We again consider a package $\mathcal{P}$ and an internal object $\mathcal{O}$, characterized by parameters collectively denoted as Ψ, and their interactions represented by μ. While retaining the terminologies and objectives from Section 4.3, this chapter introduces a nuanced shift by employing a graph-based representation of the system, as illustrated in Fig. 5.2.

The entire package-object-suction cup system is represented as a structured graph, composed of nodes that correspond to distinct physical entities and edges that encode spatial and relational interactions between these entities.

Specifically, as illustrated in Fig. 5.2, the graph comprises three distinct types of nodes: those belonging to the thin-shell deformable package, the deformable suction cup, and the rigid internal object. Each node is explicitly assigned to its respective entity type to maintain clarity in the representation.

Edges in the graph are introduced to capture the spatial structure, dynamics, and inherent interactions within and between entities. The graph contains two main classes of edges: inter-entity edges and intra-entity edges. Inter-entity edges model the interactions between different types of entities, such as connections from the suction cup nodes to the package nodes and from the package nodes to the internal object nodes. These edges encode interactions like gripping constraints and contact dynamics. On the other hand, intra-entity edges represent the internal structure and intrinsic spatial relationships within each entity, for example, edges connecting neighboring nodes within the deformable package itself.

These edges are constructed based on criteria such as spatial proximity, known physical constraints, and the established relational structure of the entities within the system. Collectively, the nodes and edges form a comprehensive representation that facilitates capturing the complex dynamics inherent in the interactions among rigid and deformable components. The attributes utilized to construct rich and informative embeddings for both nodes and edges, which ultimately enable accurate prediction of the system's forward dynamics, are described in detail in the following sections.

5.4 Message Passing Overview

As previously discussed, a significant advantage of adopting a graph-based representation is its inherent capability to capture the spatial relationships and structural dynamics among entities, akin to the FEM-based method outlined in

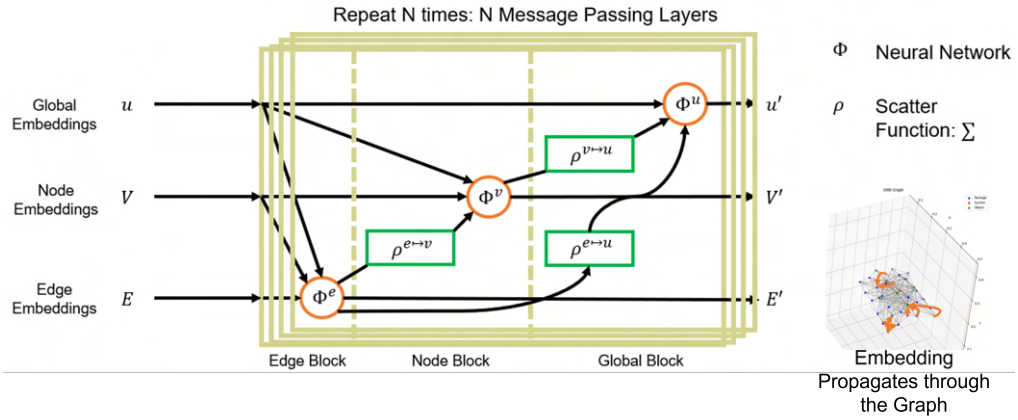


Figure 5.3: Overview of the graph message-passing framework used to capture complex interactions among the suction cup, package, and internal object. Node-level, edge-level, and global-level attributes are embedded and propagated through successive message-passing layers to model both local and system-wide dynamics. The output of these message passing layers is latent representations of the node, edge, and global embeddings, which can then be further passed to a decoder for downstream dynamics predictions.

Chapter 4. A widely used approach for modeling such relationships in graphs is the Graph Neural Network (GNN) framework [75], which leverages message passing to propagate embeddings across graph nodes, encoding relationships through edges to derive meaningful inductive biases.

The suction-package-object system examined in this study inherently provides three categories of attributes critical for an accurate and informative graph-based representation: (1) **Node Attributes**, (2) **Edge Attributes**, and (3) **Global Attributes**. Each attribute type uniquely captures essential characteristics and interactions within this complex, rigid-deformable system. Specifically, node-level attributes include positional data and entity identification (e.g., package, suction cup, internal object). Edge-level attributes encapsulate the relational interactions between nodes, reflecting proximity, constraints, and physical interactions. Lastly, global attributes encompass external control inputs, such as forces applied to the system, and intrinsic system properties like mass distribution and overall dimensions. The details on how we encode these

to generate the corresponding embeddings are given in Sections 5.5.1, 5.5.2, and 5.5.3.

To effectively model these complex interactions, we build upon the Graph Network Block introduced by the authors in [76]. Their framework naturally aligns with our requirements, and we extend this message-passing mechanism to accurately represent and capture the dynamics of our suction-package-object system, integrating the structured graph attributes into our representation to enhance predictive performance and fidelity.

The message passing framework utilized in our approach involves two key functions: the encoder function (Φ), implemented as a Multi-Layer Perceptron (MLP), and the scatter function (ρ), which aggregates and propagates embeddings throughout the graph. Specifically, we employ the sum operator ($\sum$) as our scatter function to collate information effectively. Within a message-passing layer, there are three distinct components: (1) the Edge Block, (2) the Node Block, and (3) the Global Block. Each block consists of its own dedicated Φ and ρ functions, working cohesively to capture and propagate intricate relational dynamics and interactions, as illustrated in Figure 5.3.

Overall, the message passing steps can be encapsulated by the following equations:

$$e'_k = \phi^e(e_k, v_{r_k}, v_{s_k}, u) \quad \bar{e}'_i = \rho^{e \mapsto c}(E'_i) \tag{5.1}$$

$$v'_i = \phi^v(\bar{e}'_i, v_i, u) \quad \bar{e}' = \rho^{e \mapsto u}(E') \tag{5.2}$$

$$u' = \phi^u(\bar{e}', \bar{v}', u) \quad \bar{v}' = \rho^{v \mapsto u}(V') \tag{5.3}$$

Where, $V = \{v_i\}_{i=1:N^v}$ is the set of nodes with cardinality N^v, and each v_i is the node's attribute. Similarly, the set of edges with cardinality N^e is

represented by $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$, where e_k is the edge attribute for the edge between a receiver node r_k and sender node s_k. Furthermore, $E'_i = \{(e'_k, r_k, s_k)\}_{r_k=i,k=1:N^e}$, $V' = \{v'_i\}_{i=1:N^v}$, and $E' = \cup_i E'_i = \{(e'_k, r_k, s_k)\}_{k=1:N^e}$. These update operations are applied over N message-passing layers, establishing an N-hop receptive field depth. As a result, each node's final embedding aggregates information from all neighbors up to N steps away, allowing the GNN to capture both local interactions and broader system-level context.

5.5 Graph-based Neural Dynamics Model

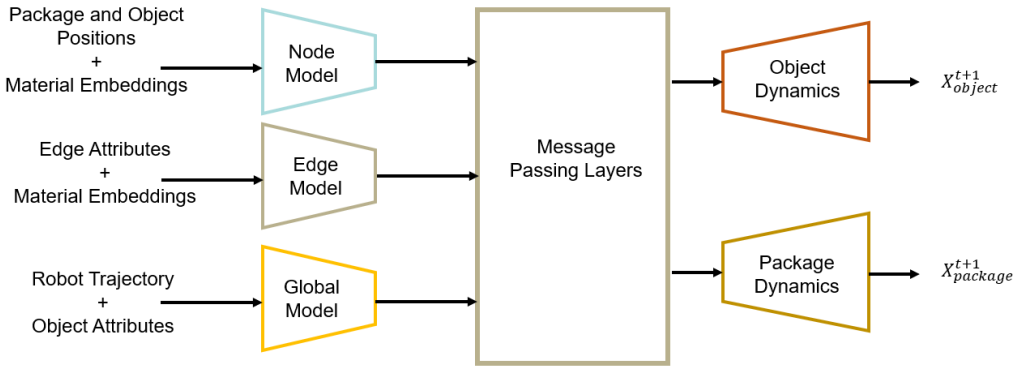


Figure 5.4: Graph-based neural dynamics overview: given node, edge, and global attributes, the model uses successive message-passing layers to generate latent embeddings and predict the system's forward dynamics for the coupled suction-package-object network. The latent embeddings from the message passing layers are then passed to a multi-head decoder that predicts the forward dynamics of the internal object and the package, respectively.

The message passing framework provides a structured mechanism for capturing and propagating spatial relationships within the graph representation. However, the central objective of this work is to accurately predict the dynamics of the suction-package-object system. To effectively learn meaningful inductive biases through message passing, it is crucial to encode attributes that comprehensively represent both the system's state and control inputs.

Analogous to FEM-based methods grounded explicitly in parameters such as stiffness, damping, and adhesion, which serve as surrogates for real-world material properties, it becomes equally important for graph-based neural models to be similarly grounded in explicit physics-based parameters.

Prior work, such as presented by Zhang et al. [68], introduced methods for encoding material parameters within a Graph Neural Dynamics (GND) framework. However, their approach lacked explicit modeling of multi-object interactions, particularly those involving both rigid-deformable and deformable-deformable entities. Additionally, their framework used global, object-level material embeddings rather than entity-specific conditioning, limiting its representational capacity. Furthermore, their predictions were restricted to quasi-static scenarios, failing to capture the continuous and dynamic interactions observed in real-world settings such as the suction-package-object system studied here.

Addressing these limitations, we propose a novel methodology and model architecture that extends beyond simple deformation predictions and global material embeddings. Our approach introduces node-level and edge-level material properties, explicitly encoding interactions such as rigid-deformable and deformable-deformable within the graph structure. By grounding the node and edge attributes directly on physics-based material parameters, our model integrates strong physics-informed priors, enhancing predictive accuracy and generalization. Additionally, we introduce a systematic method for constructing a comprehensive material embedding matrix capable of accommodating diverse package and object materials typically encountered in practical scenarios.

To robustly model the coupled dynamics of the package and internal object, our proposed architecture features two decoder heads with shared parameters (detailed in Section 5.5.4). This design choice significantly improves the model's predictive capability, enabling it to capture the intricate interdependencies and continuous dynamics characteristic of the suction-package-object system, as demonstrated in our experimental results.

5.5.1 Node Encoder

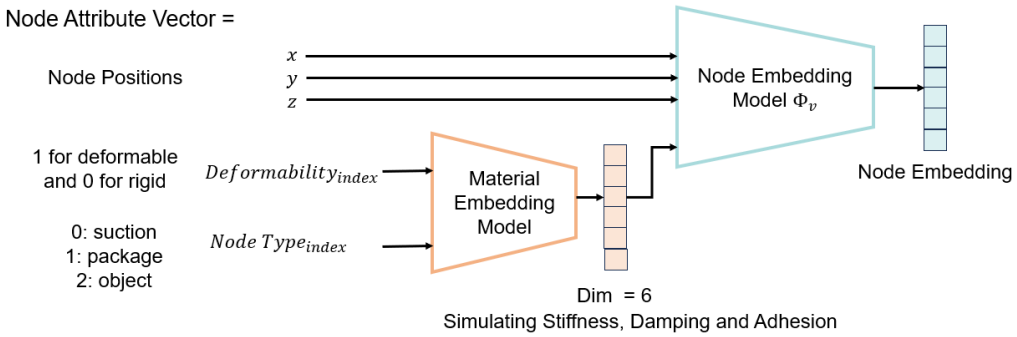


Figure 5.5: Node-level encoder architecture: for each node, the encoder network ingests its spatial coordinates (expressed in the end–effector frame to preserve SE(3) equivariance) along with a learned material embedding derived from its deformability flag (0 = rigid, 1 = deformable) and entity type (0 = suction, 1 = package, 2 = object). The MLP applies these inputs to produce a compact node embedding that seeds the subsequent message-passing layers.

To effectively encode node attributes and ground them in relevant material parameters, we propose the node embedding model illustrated in Fig. 5.5. A critical feature of this model is its input representation: each node's x, y, z position is expressed in the robot's end-effector frame of reference, ensuring that the model maintains $\mathbb{SE}(3)$ equivariance. Additionally, to embed meaningful physical characteristics, we explicitly encode both a deformability index (rigid or deformable) and an entity-type index (suction cup, package, or internal

object). These indices are processed through a dedicated material embedding model, resulting in a compact, fixed-size material embedding.

Subsequently, the positional data and the material embeddings are jointly processed through an MLP. This MLP generates a comprehensive node embedding vector, effectively capturing both spatial configurations and physics-based material properties. Consequently, the resulting node embeddings provide a robust and physically grounded representation, essential for accurate modeling of complex interactions within the suction-package-object system.

5.5.2 Edge Encoder

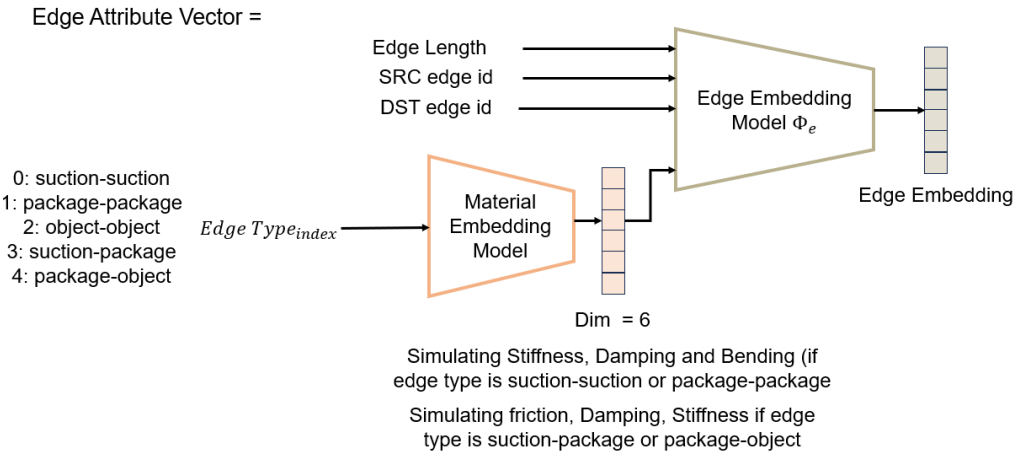


Figure 5.6: Edge-level encoder architecture: each edge's encoder ingests the source and target node IDs alongside their Euclidean distance that preserves $\mathbb{SE}(3)$ equivariance, and an edge-type identifier that specifies whether the connection is rigid–deformable, deformable–deformable, or intra-entity. An MLP then combines these inputs into a rich edge embedding.

The edge encoder model shares a similar architecture to the node encoder model but is specifically designed to represent relationships between node pairs. Since translation equivariance is a fundamental property desirable in any dynamics model, we encode this equivariance by explicitly utilizing the Euclidean distance between the two nodes forming an edge.

Edges between nodes are established based on specific criteria reflective of the system's inherent physical constraints and interactions. Specifically, edges are created either between nodes belonging to the same entity (i.e., internal object, package, or suction cup) or between nodes from different entities based on their proximity with each other, provided they represent physically meaningful interactions—namely, edges between package and suction cup nodes or between package and internal object nodes. Edges are not established directly between the suction cup and internal object nodes since these entities do not directly interact within the system under consideration. Furthermore, as outlined in Section 5.4, each edge explicitly incorporates the IDs of its corresponding source and target nodes, ensuring clear relational identification within the graph.

To further ground the edge attributes on physics-inspired material parameters, we introduce an embedding matrix specifically for edge types, as illustrated in Fig. 5.6. This embedding matrix encodes information about the interaction type, such as deformable-rigid or deformable-deformable interactions. The generated material embedding is then concatenated with the existing edge attributes, subsequently passing through a dedicated edge embedding MLP. This process ensures the resulting edge embeddings accurately capture both the spatial relationships and the physical constraints inherent to the interactions within the suction-package-object system.

5.5.3 Global Encoder

The global encoder plays a crucial role in capturing overarching system-level context essential for precise downstream dynamics prediction. It aggregates key attributes of the package-object system, including intrinsic properties like mass distribution and dimensional characteristics. Additionally, the encoder

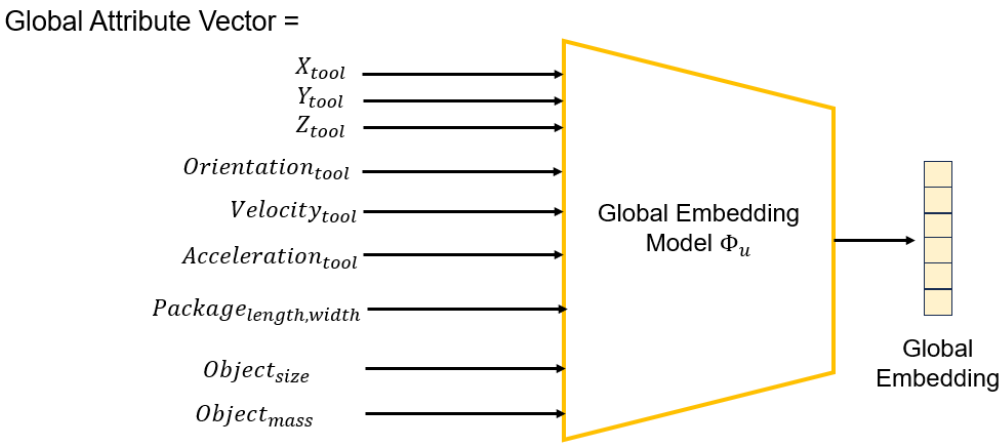


Figure 5.7: Global-level encoder architecture: embeds system-wide attributes—including package and object mass and dimensions—alongside external control inputs (end-effector position, orientation, velocity, and acceleration) into a unified global context vector for downstream dynamics prediction.

incorporates external control inputs—specifically, the robot's end-effector state described by its position, orientation, velocity, and acceleration. By integrating both intrinsic object properties and dynamic control inputs into a unified embedding, the global encoder provides the model with comprehensive contextual awareness, enabling robust and accurate predictions of the coupled rigid-deformable interactions within the system.

5.5.4 Dynamics Decoder

Predicting the coupled dynamics of our suction–package–object system requires a decoder that can disentangle rigid-body motion from deformable behavior while still modeling their mutual influence. In practice, we observed two distinct patterns: the internal object largely follows an independent rigid-body trajectory, especially under high orientation changes and accelerations, while the package retains its general shape but deforms in response to both external forces and the object's motion.

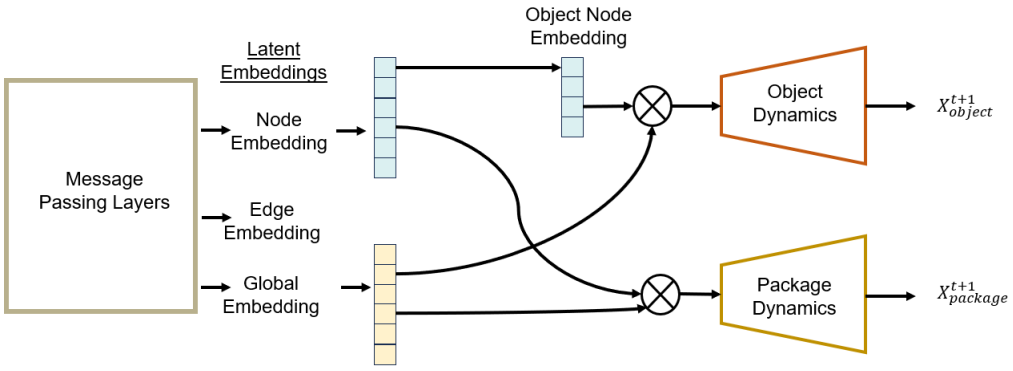


Figure 5.8: Multi-head decoder architecture: two specialized decoder heads jointly predict the coupled dynamics of the internal object and the deformable package. The object head consumes only the object's node embeddings plus the global context to forecast rigid-body motion, while the package head processes all package node embeddings alongside the same global vector to predict continuous deformations. Together, they capture the interdependent behavior of both subsystems.

To capture these nuances, we employ a multi-head decoder (Fig. 5.8) with two specialized branches:

1. Object Head:
   (a) Inputs: Only the internal object's node embeddings plus the shared global context embedding.
   (b) Rationale: The object behaves like a rigid body; its future state depends primarily on its own local interactions and the overall system context, not on the full deformable mesh. Experimentally, isolating these embeddings improved object-motion accuracy by an order of magnitude over a combined decoder.
2. Package Head
   (a) Inputs: All package node embeddings concatenated with the same global context embedding.
   (b) Rationale: The package's deformation arises from both external control inputs and the internal object's movement. Including the entire

set of package nodes allows the decoder to model how local deformations propagate across the shell.

Both heads predict delta-state updates (changes in node positions) rather than absolute positions, further simplifying learning and ensuring stable integration with downstream control modules. By structurally separating the rigid and deformable components—and tailoring each head's receptive field accordingly—we enforce an inductive bias that mirrors the physical interdependence observed in real experiments. This design not only validates our hypothesis about coupled dynamics but also delivers substantial gains in predictive fidelity for both rigid-body and deformable motions.

5.5.5 Model-Training

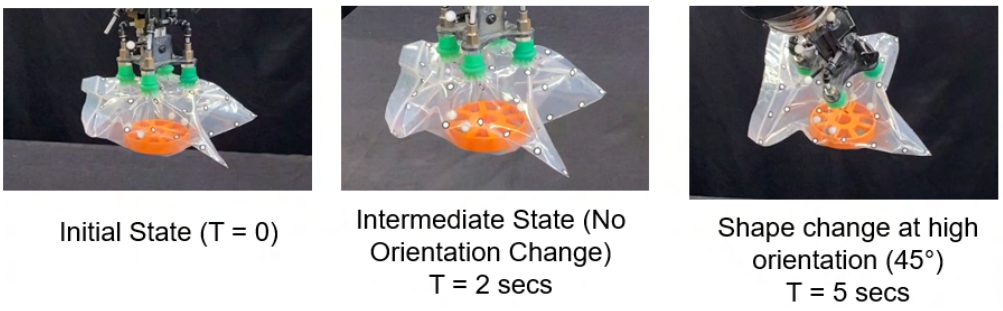


Figure 5.9: Shape-retention under low dynamic loads: when the end-effector's orientation and acceleration remain minimal (up to 2 s), the package maintains its initial shell shape. Once orientation changes intensify and acceleration increases, combined with internal object motion and suction-cup compliance, the package visibly deforms.

We train our graph-based neural dynamics model using a composite loss that is coherent with the objectives from Chapter 4 while introducing a new term to enforce realistic shape retention. Specifically, our total loss is

$$\mathcal{L} \;=\; \lambda_1 \mathcal{L}_{\text{chamfer}} \;+\; \lambda_2 \mathcal{L}_{\text{MSE}} \;+\; \lambda_3 \mathcal{L}_{\text{shape-retention}} \tag{5.4}$$

1. Package Chamfer Loss

$$\mathcal{L}_{\text{chamfer}} = \text{Chamfer}\big(\mathcal{P}_{\text{pred}},\, \mathcal{P}_{\text{gt}}\big)$$

measures the bidirectional Chamfer distance between the predicted and ground-truth package surface point sets (Eq. 4.2). It captures both global and local geometric discrepancies without requiring explicit point correspondences, making it ideal for thin-shell geometry.

2. Object Mean-Squared-Error Loss

$$\mathcal{L}_{\text{MSE}} = \big\|\mathbf{c}_{\text{pred}} - \mathbf{c}_{\text{gt}}\big\|^2$$

penalizes the squared error in the center-of-mass positions of the internal object (Eq. 4.3). Given the object's rigid-body behavior, this simple MSE term suffices to drive accurate rigid-body predictions.

3. Shape-Retention Loss To enforce the experimentally observed behavior, namely, that under small end-effector rotations (below 20°) and low accelerations, the package shape remains essentially unchanged (see Fig. 5.9)—we introduce a physics-inspired regularizer:

$$\mathcal{L}_{\text{shape-retention}} = \lambda_\delta\, \mathcal{L}_\delta \left(1 - \sigma(\alpha\, \Delta R + \beta\, \Delta a)\right), \tag{5.5}$$

where the core penalty term is

$$\mathcal{L}_\delta = \frac{1}{N_v} \sum_{i=1}^{N_v} \max\big(\|x_t^i - x_{t-1}^i\| - \delta,\, 0\big)^2,$$

This term clips small displacements below a threshold δ (set according to the commanded robot trajectory) and penalizes only larger per-node movements. Here, ΔR and Δa are the changes in end-effector orientation and acceleration, respectively; σ is a sigmoid function that attenuates the penalty as motion intensity increases; and $\lambda_\delta, \alpha, \beta$ are tunable hyperparameters.

By predicting delta positions instead of absolute coordinates, and constraining those deltas within physically plausible limits, we enforce smooth, locally coherent updates that honor both the graph topology and the system's inherent dynamics. Empirically, the shape-retention regularizer enhances numerical stability, mitigates spurious deformations under mild motions, and produces gradients that align with real-world physical behavior. Using this composite loss (Eq.5.4), we train the model depicted in Fig.5.4, yielding robust, accurate predictions of the suction–package–object system's forward dynamics.

5.6 Experiments

We evaluated our graph-based model using an extended version of the experimental protocol from Section 4.6, broadening both the physical configurations and the dynamic conditions under which the system operates. In addition to our original package and internal object parameters, we introduced two new sets of package and object dimensions and mass distributions (see Fig. 5.10), challenging the model to generalize across a wider range of material properties. To further stress-test performance, we incorporated higher end-effector velocities into our draping maneuvers, resulting in a dataset of 244 suction-driven trajectories that span gentle to aggressive motion regimes.

As before, we collected time-stamped motion-capture data for both the package mesh and the internal object's center of mass, processed through the same registration and filtering pipeline described earlier. Importantly, we also made

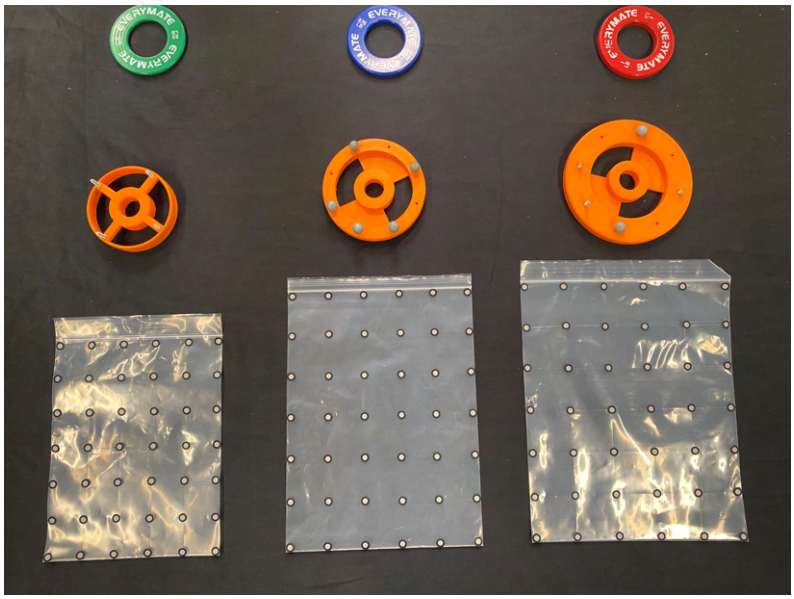


Figure 5.10: The three different classes of package size, object size, and object mass.

use of failure-mode data: whenever a trajectory ended abruptly, detected via a sudden wrench signature on the end-effector, we truncated the sequence at the failure point and included it in training. By combining varied geometric configurations, increased velocity conditions, and both successful and truncated failure trajectories, our experiments rigorously assess the model's ability to capture nuanced rigid–deformable interactions and maintain predictive accuracy across realistic, high-variability scenarios.

5.7 Results

In our experimental evaluation, we focused on two core questions that mirror those from Chapter 4, but under our expanded test conditions. First, we asked: "How does the Graph Neural Dynamics (GND) model's accuracy in predicting package deformation and internal object motion compare to that of our FEM-based simulator?" Second, we probed computational performance:

"Can the GND model deliver comparable—or better—prediction fidelity while running significantly faster than the FEM pipeline?" By directly comparing point-set errors for the package surface, center-of-mass errors for the object, and end-to-end run times across all 244 test trajectories (including both gentle and aggressive motions, as well as truncated failure cases), we were able to quantify each approach's strengths and trade-offs.

Method	**Dataset**	**Num Trajectories**	**Object (MSE) ↓**		**Package (CD) ↓**	
			Mean	Max	Mean	Max
FEM	Train	20	0.006	0.003	0.032	0.036
	Test	4	0.0008	0.001	0.033	0.038
GND	Train	236	0.00001	0.0002	0.027	0.038
	Test	8	0.00001	0.0005	0.029	0.045

Table 5.1: Comparison of prediction errors for our Graph Neural Dynamics (GND) model versus the FEM-based simulator from Chapter 4. GND consistently yields lower mean and max errors for both package deformation and internal object motion, demonstrating more reliable rigid–deformable coupling. Occasional spikes in the maximum error primarily correspond to test frames with missing motion-capture markers.

Table 5.1 summarizes the quantitative comparison between our Graph Neural Dynamics (GND) model and the FEM-based simulator. The most striking improvement is seen in the internal object prediction, where GND reduces the mean center-of-mass error by over 10x. This gain can be directly traced to our multi-head decoder design, which decouples rigid-body and deformable predictions into specialized branches—whereas a single-head decoder yielded performance on par with FEM. We also see a consistent reduction in the mean package-surface error, further demonstrating GND's ability to capture subtle deformations more accurately than FEM. Overall, these results highlight both the architecture's effectiveness at modeling interdependent dynamics and the model's superiority over traditional FEM in our suction–package–object scenario.

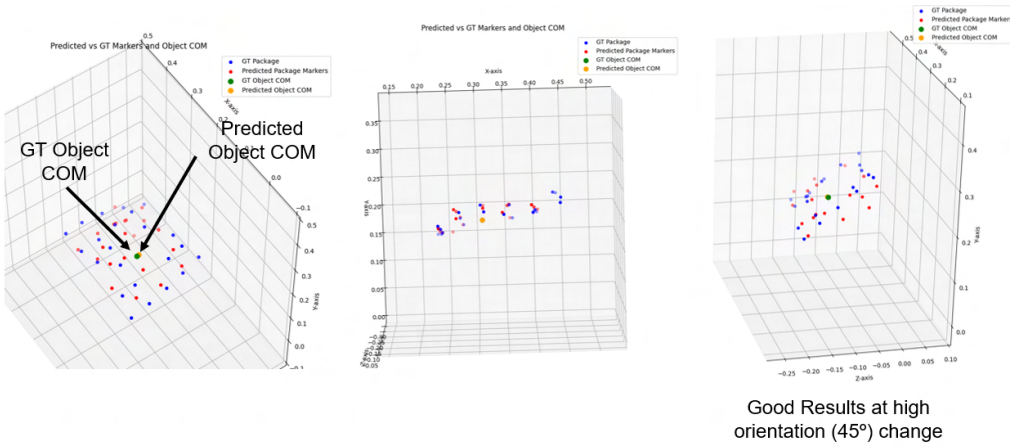


Figure 5.11: Qualitative comparison of predicted versus ground-truth trajectories for both package deformation and internal object motion under high end-effector orientation. The Graph Neural Dynamics model closely tracks the true deformations and rigid-body movements, demonstrating robust performance even in challenging configurations.

Figure 5.11 shows representative predictions of both package deformation and internal object motion under extreme end-effector orientations. Even as the robot tilts the package beyond typical operating angles, the Graph Neural Dynamics model faithfully reproduces subtle shell deformations and rigid-body shifts of the internal object. Error analyses in Figure 5.12 confirm that residual discrepancies are concentrated at the package's distal edges—well outside the suction-cup region where failures occur—mirroring the spatial error patterns observed with our FEM baseline in Chapter 4. Crucially, this localization of error ensures that trajectory planners can rely on the model's outputs to maintain grip integrity and avoid drop events. Finally, the GND model runs approximately five times faster than the FEM simulator in a forward pass, enabling real-time trajectory optimization and closing the loop in responsive, in-situ manipulation tasks.

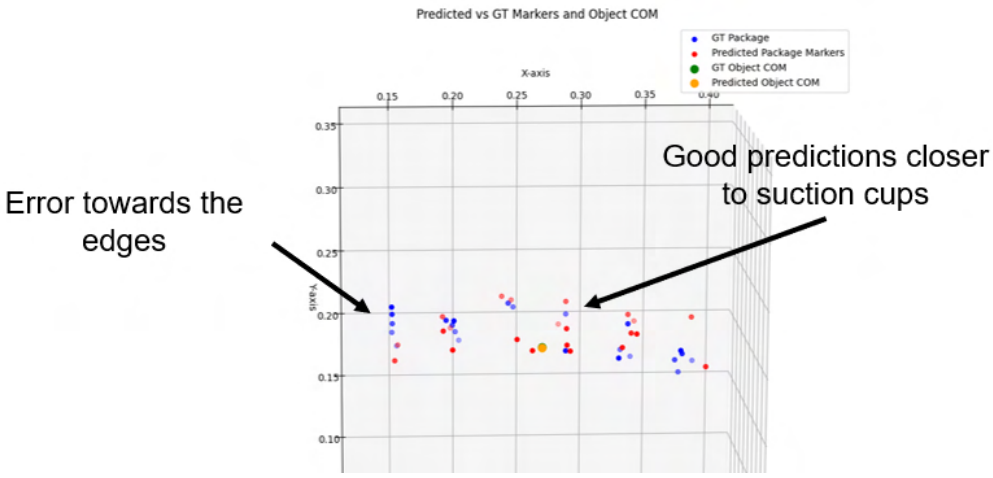


Figure 5.12: Worst-case error scenario for package deformation: the largest discrepancies between predicted and actual meshes occur at the sheet's far edges, while the region near the suction cup, critical for grip stability, remains accurately modeled. This indicates that even under high-error conditions, the model's predictions in the failure-critical zone are reliable enough for robust trajectory planning.

5.8 Summary

This chapter presented a Graph Neural Dynamics (GND) model designed to simulate the coupled behavior of a suction–package–internal-object system. By representing each component—deformable suction cup, thin-shell package, and rigid internal object—as nodes in a unified graph, and by explicitly encoding their spatial and physical relationships through edge and global attributes, a continuous, data-driven dynamics model is learned that is capable of handling complex rigid–deformable interactions. The proposed architecture conditions each node and edge on material-specific parameters, such as stiffness, damping, and deformability indices, via dedicated embedding networks, thereby grounding the learned representations in real-world physics. Furthermore, a shape-retention regularizer is introduced that enforces experimentally

observed behavior (minimal shell deformation under low rotations and accelerations), yielding smooth, stable predictions that respect the system's physical constraints.

Empirically, the GND model matched or exceeded the FEM baseline's predictive accuracy while running an order of magnitude faster, making it well-suited for online trajectory generation. Importantly, errors remained localized away from the suction-cup region, where grip failures occur, demonstrating the model's reliability in safety-critical zones. This performance opens the door to real-time, physics-informed trajectory optimization for such complex suction-based manipulation tasks of deformable objects.

Chapter 6

Learning Task Sequencing Policies for Deformable Object Manipulation

6.1 Introduction

Task planning plays a central role in industrial processes involving deformable object manipulation. These processes often consist of a sequence of interdependent subtasks, where the order of execution significantly influences the final product quality. In many real-world scenarios, an incorrect or suboptimal task sequence can introduce defects that are difficult or impossible to rectify downstream, resulting in costly rework, scrapped parts, or compromised safety and performance. This sensitivity to task ordering becomes even more critical when dealing with deformable components, whose dynamic and compliant behavior can amplify the consequences of early-stage mistakes in the process.

Industrial operations such as composite prepreg layup, surface finishing, and protective coating applications all involve the sequential processing of a part, often by human experts. In composite layup, for instance, the direction and sequence of ply placement affect both the structural integrity and the manufacturability of the part. In surface finishing tasks, the order in which regions

are processed can influence the uniformity of material removal, while in coating applications, sequence affects drying time, adhesion, and coverage. The precision and repeatability required in these tasks make sequence planning not just beneficial, but essential to success.

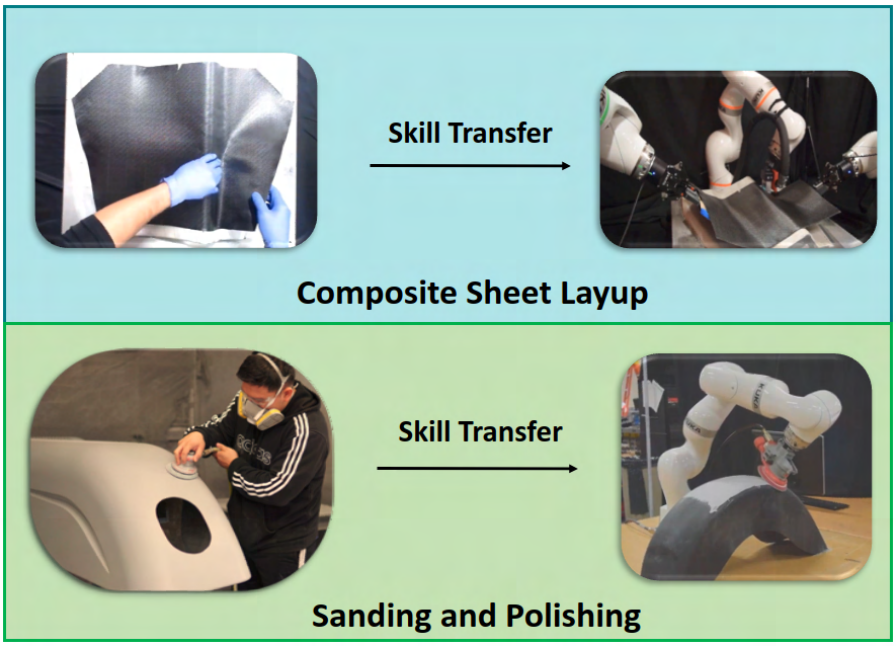


Figure 6.1: Example processes that require human-to-robot skill transfer.

In practice, skilled human operators rely on accumulated experience and process intuition to make these sequencing decisions, balancing quality objectives with practical concerns like tool accessibility and ergonomic constraints. However, a growing shortage of skilled labor [77] is creating urgency around automating these knowledge-intensive tasks. As industries seek to scale production while maintaining quality, there is a need for robotic systems that can perform complex operations traditionally handled by human experts. These processes require both motion planning and high-level task planning to work in concert, encompassing the execution of individual actions and the strategic decomposition and sequencing of tasks to ensure success. Deformable object manipulation further amplifies these requirements, as the interaction between

sequential actions and material response is often nonlinear and difficult to model explicitly.

To address this, the work presented in this chapter focuses on learning task sequencing policies from human expert demonstrations, with the goal of transferring domain knowledge to robots in a structured and scalable manner, rather than relying on heuristic rules or physics-based simulation, which can be prohibitively complex for modeling defect propagation in deformable materials [1, 78, 79], the proposed approach models expert intent through Inverse Reinforcement Learning (IRL). Rather than simply mimicking observed sequences, the IRL approach infers the underlying reward functions that drive expert behavior, allowing for generalization to new tools and unseen process configurations.

As shown in Fig. 6.1, the goal is to enable human-to-robot skill transfer for processes where task sequencing is critical. Experts typically decompose a high-level operation into subtasks, each corresponding to a local action on a specific region of the part. These subtasks are executed in a carefully considered sequence, informed by both performance goals (e.g., quality, coverage) and ergonomic or operational constraints (e.g., tool access, ease of motion). By interviewing domain experts and capturing detailed demonstrations, this chapter investigates how these preferences can be modeled and learned through feature-driven IRL.

A key observation from our expert interviews is that multiple task sequences may be viable for a given tool, but not all are preferred. The preference may stem from various external factors, such as workspace layout, process ergonomics, or tool design. To address this, the learned policy distinguishes between performance-based preferences (which affect outcome quality) and effort-based preferences (which reflect ease of execution), and learns them independently. We further introduce a method for assessing feature interaction

coverage in demonstration datasets to ensure the generalizability of the learned policy.

The resulting framework enables robots to plan and execute high-level operations on deformable objects by sequencing subtasks in a manner consistent with expert intent. This chapter presents:

- A structured representation of process decomposition and region-based task modeling for deformable object operations
- A data collection and feature analysis pipeline to support preference modeling from human demonstrations,
- An IRL-based policy learning framework for sequencing tasks under varying constraints, and Experimental validation on a real-world industrial application involving high-mix, low-volume parts.

6.2 Related Work

IRL Overview: The field of IRL has been mainly categorized under the umbrella of imitation learning, with the objective of learning an expert's policy. Earlier work in IRL focused on imitating an expert's policy on already demonstrated data or simple tasks [80]. There are four main IRL approaches: Max-margin IRL, Max-entropy IRL, Bayesian IRL, and Regression/Classification IRL [81]. Max-entropy-based methods [82] propose a solution to select weights in the presence of incomplete information. In contrast, Bayesian-IRL methods [83] compute a probability distribution over reward functions based on informative priors. Most of the work in IRL assumes a linear reward function, except for [84], which focuses on sub-optimal stochastic demonstrations. In this work, our problem requires us to strictly comply with the expert in the absence of any noise. Besides finding a reward function for the expert,

we also need to penalize sequences that result in poor process performance. Therefore, max-entropy and Bayesian IRL-based methods are not suitable. Max-margin methods proposed by [85, 86] introduce the concept of scaling sequences based on their proximity to the expert's demonstration. Therefore, max-margin-based methods provide the appropriate learning scheme for our problem. However, previous work on max-margin-based methods did not address scenarios with varying levels of preferences, feature interactions, and limited demonstrations. Recent work proposed in [87–89] introduces the active learning element to learn reward functions from preferences. However, these methods do not take into account interacting features.

IRL in Manufacturing: IRL for sequencing has been studied for assembly tasks in [90–92], but the focus is on assisting the expert rather than transferring policy to the robot. Several works also focus on learning rewards for insertion tasks [93, 94]. In [95], the authors mention using IRL for sequencing for machining tasks, but do not account for transferring policies to a robot. Prior work in learning from demonstration has explored the problem of solving surface finishing operations [96–99], but they do not address the region sequencing problem.

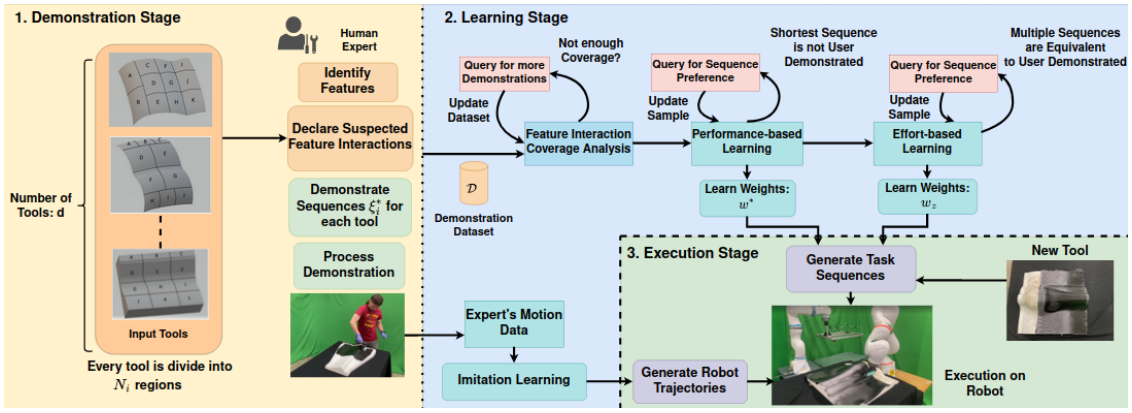


Figure 6.2: Overview of the proposed framework for learning task sequencing policy.

6.3 Problem Formulation

We formulate the problem described in Section 6.1 as a task sequencing problem, where the robot has to perform a set of subtasks to complete the process.

Assumptions: Our assumptions for the task sequencing problem are as follows:

- Demonstrations are optimal and have no noise as they are performed by experts.
- Features and Feature Interaction information can be captured from the expert's process description.
- Only pairwise feature interactions are present.

State Preliminaries: Section 6.1 describes processes that feature a tool, as shown in Fig. 6.2. The tool is divided into N_i regions, $\forall i \in 1, .., d$, where d is the number of demonstration tools. Regions are indexed alphabetically based on their appearance row-wise from left to right. A task for the agent is defined as processing a region on the tool, so the agent's objective is to compute a desirable sequence for performing these N_i tasks.. We denote the human expert by $\mathcal{H}$, whose performance and effort-based preference we are trying to model. The autonomous agent that learns from $\mathcal{H}$ will be represented by $\mathcal{A}$. The state of $\mathcal{A}$ gets denoted as $s \in \mathcal{S}_i, \mathcal{S}_i \mapsto \mathbb{R}^{N_i}, \forall i \in \{1, .., d\}$, and we represent s as a set of tasks that $\mathcal{A}$ completes. The action that $\mathcal{A}$ takes at a state s is denoted as $a \in A$, where A is a set of available actions at state s. An action a signifies the task that $\mathcal{A}$ chooses to perform. In our case, A gets defined by the set of tasks that are yet to be executed by the agent. The state transitions $s \rightarrow s'$ due to action a are assumed to be completely deterministic.

Transition Cost: In this chapter, we will refer to reward as a negation of cost. Hence, our formulation will refer to the IRL problem of maximizing reward as

minimizing cost. We define the policy of the agent as to follow a minimum cost sequence for performing the N_i tasks. In a standard inverse reinforcement learning problem, the cost for transitioning to a state gets defined as a linear function of weights w and an array of feature values ϕ dependent only on the agent's current state [100]. The nature of the sequencing problem we study in this work is such that the feature array is a function of the state s and action a. Additionally, as per $\mathcal{H}$, certain feature pairs in the feature array might interact with each other.

Therefore, we define the cost of transition for $\mathcal{A}$ by Eq. 6.1, where $\phi(s,a) = \{\phi_1, \phi_2, ..., \phi_n\}$ is an array of feature values that are dependent on state transitions. The set of interacting features for a subset of feature values from $\phi(s,a)$ is denoted by $\phi_{int}(s,a) = \{(\phi_i \cdot \phi_j)_1, .., (\phi_k \cdot \phi_l)_m\}, \forall i,j,k,l \leq n,\ i \neq j, k \neq l$. The total number of feature interactions suspected by the expert $\mathcal{H}$ is m. In our formulation, non-linearity is introduced by $\phi_{int}(s,a)$. We will denote our unified weight array as $w = \{w^\frown w_{int}\}$, that is a concatenated array of the linear feature weights w and the interacting feature weights w_{int}.

$$c(s,a) = w^T \phi(s,a) + w_{int}^T \phi_{int}(s,a) \tag{6.1}$$

$$w, \phi \in \mathbb{R}^n, \forall \phi : s \in \mathcal{S}, a \in A$$

$$w_{int},\ \phi_{int}(s,a) \mapsto \mathbb{R}^m$$

Learning Performance-based Preferences: The human expert $\mathcal{H}$ demonstrates a desired sequence ξ^* on a sample tool. The agent $\mathcal{A}$ has access to d demonstrations, each on different tools. We define the demonstration dataset as $\mathcal{D} = \{\delta_1, \delta_2, ..., \delta_d\}$, where $\delta_i = \{\xi_i^*, \mathcal{F}_i, N_i\}$, $\mathcal{F}_i$ is the feature information

for i^{th} demonstration tool from which ϕ is computed. Furthermore, the expert $\mathcal{H}$ also gives information about the interacting feature set ϕ_{int} for the demonstration dataset $\mathcal{D}$. Now, as described earlier, the agent has to solve the following problem:

$$C(\xi_i^*) \leq C(\xi_i^j), \forall \xi_i^* \in \mathcal{D}, \forall \xi_i^j \in \mathcal{P}_i \mapsto \mathbb{R}^p \tag{6.2}$$

where,

$$C(\xi_i) = w^T \Phi(\xi_i)$$

$$\Phi(\xi_i) : \{\sum_{k=1}^{N_i} (\phi(s_k, a_k) \frown \sum_{k=1}^{N_i} \phi_{int}(s_k, a_k)\}$$

In Eq. 6.2, $C(\xi_i)$ is the total cost incurred by the agent for following a sequence ξ_i. We define $\mathcal{P}_i$ as the set of all possible sequences for the i^{th} tool. $\Phi(\xi_i)$ is an array of the sum of individual feature values and interactions for the state transitions in sequence ξ_i. Thus, we find a weight array w^* by solving Eq. 6.2 such that the expert demonstrated sequence ξ_i^* is the lowest cost sequence for all the corresponding tools in $\mathcal{D}$.

Learning Effort-based Preferences: The feature values used for the problems discussed in Section 6.1 are geometric features of the tool. Thus, for a given demonstration in $\mathcal{D}$, there might be several sequences with equivalent feature arrays due to the symmetry of the tool. These equivalent sequences (ξ_i^e) will have identical costs to the user-demonstrated sequence (ξ_i^*). We formulate this feature equivalence property by Eq. 6.3.

$$\Phi(\xi_i^*) \equiv \Phi(\xi_i^e); \xi_i^e \in \Omega_i, \forall i \in \{1, ..., d\} \tag{6.3}$$

where, Ω_i is the set of sequences for i^{th} demo that exhibit the equivalence property in Eq. 6.3. However, as discussed in Section 6.1, the expert $\mathcal{H}$ might prefer specific sequences in Ω_i due to their effort-based preferences. $\mathcal{H}$ might choose sequences that start at a specific task and end at a specific task to improve their ease of operation. In this case, $\mathcal{H}$ is queried to compare their preference for the equivalent sequences in relation to corresponding ξ_i^*'s in $\mathcal{D}$. The objective is to learn a preference penalty function η such that the preferred sequences ξ_i^{pref} have the lowest penalty. Hence, we can formulate the effort-based preference learning problem as follows:

$$\eta(\xi_i^{pref}) < \eta(\xi_i^j),\ \forall i \in \{1, ..., d\}, \forall \xi_i^j \in \psi_i, \forall \xi_i^{pref} \in \psi_i^* \tag{6.4}$$

where, $\eta(\xi) = w_z^T\ z(\xi)$; and $z(\xi)$ are effort-based preference features for an arbitrary sequence ξ. We denote, $\psi_i^* \subseteq \Omega_i$ as the set of preferred sequences, and $\psi_i \subseteq \Omega_i$ as the set of unpreferred sequences $\ s.t.\ \psi_i^* \cap \psi_i = \emptyset$. It is important to note that adding $z(\xi)$ in learning the weights w in Eq. 6.2 would not be appropriate as $z(\xi)$ will change based on external factors mentioned in Section 6.1.

In situations when the expert has no preference between the demonstrated sequence and any other equivalent sequence ξ^e, we do not need to learn the preference penalty $\eta(\cdot)$. We can set the value of $\eta(\xi) = 0$ for any arbitrary sequence ξ.

Feature Interaction Coverage in Demonstrations: To learn the expert's preferences, the demonstrations need to be informative with respect to feature interactions. Assuming access to enough demonstrations to capture all levels of feature values and interactions for the transitions in ξ^* is impractical. We propose feature interaction coverage as a metric that assesses whether the dataset $\mathcal{D}$ enables learning of these interactions. This metric can then be

used to query the expert for more informative demonstrations if needed. We represent this metric as $\rho(\cdot)$ and define it as follows:

$$\rho(\mathcal{D}) \propto \kappa(\{\phi, \phi_{int}\}) \tag{6.5}$$

where, the function $\kappa(\cdot)$ computes the overall extent of coverage of ϕ and ϕ_{int}. Thus, using $\rho(\cdot)$, we should be able to query $\mathcal{H}$ to ask for specific demonstrations that improve feature coverage.

6.4 Method

In the previous section, the formulation that we proposed engenders three main problems: (1) Learning $\mathcal{H}$'s performance-based preference, (2) Learning effort-based preferences, and (3) Feature interaction coverage. The entire framework is depicted in Fig. 6.2. We build upon the structured max-margin approach outlined in [85] to solve the proposed problem. Section. 6.2 gives the reasoning for selecting margin-based IRL methods for our approach. Before we commence learning performance and effort-based preferences, we evaluate the extent of feature coverage in the demonstration data. Once we have enough coverage of feature values, we proceed with learning. Subsequently, we define a graph-based state space representation for each demo tool in $\mathcal{D}$. Then, we introduce our iterative max-margin approach with an active learning element for diversely sampling the training data and a cost function designed to solve the problem in Eq. 6.2.

6.4.1 Estimating Feature Interaction Coverage

To solve the feature interaction coverage problem, we employ a 2-factor factorial design of experiments technique to capture feature values and feature

interactions on three levels: high, medium, and low. For each interaction level, we count the instances such that the remaining feature values are uniformly distributed. We use the standard Chi-squared test for determining whether the features are uniformly distributed with an α value of 0.05 [101]. Once we ensure uniform distribution, we compute a confidence metric of $\mathcal{D}$ for capturing all expected interactions according to Eq. 6.6.

$$\rho(\mathcal{D}) = \frac{\text{no of interactions captured}}{\text{total number of interactions possible}} \tag{6.6}$$

Based on the weakly captured interactions, we query the expert $\mathcal{H}$ to provide demos with a specific feature interaction value. This helps improve the $\rho(\cdot)$ value of the dataset, as shown in Section 6.6.

6.4.2 Graph-based State Sequence Representation

After obtaining $\mathcal{D}$ with sufficient feature value coverage, we represent the state space ($\mathcal{S}_i$) for every demonstration in $\mathcal{D}$ in the form of a separate graph $\mathcal{G}_i, \forall i \in \{1, ..., d\}$. Each node/vertex v in the graph is a state s, and each edge e is the cost to transition between states $s \rightarrow s'$. Therefore, each demonstration $\delta_i \in \mathcal{D}$ has a corresponding $\mathcal{G}_i$.

In $\mathcal{G}_i$, we use a one-hot vector of size N_i to encode each node, where the completed tasks are represented as 1's and incomplete tasks are represented as 0's. We define s_{start} as a state node for which none of the tasks are completed by $\mathcal{A}$ and s_{end} is the state node when $\mathcal{A}$ has completed all the tasks for a given tool. Naturally, s_{start} becomes a vector of 0's of size N_i, and s_{end} becomes a vector of 1's of size N_i. Thus, an arbitrary sequence ξ_i for $\mathcal{G}_i$ becomes the path traversed from s_{start} to s_{end}. As discussed, our objective becomes to learn w^*, such that ξ_i^* will be the shortest cost path from s_{start} to s_{end} for all the corresponding $\mathcal{G}_i$'s. For a demonstration with N tasks, there are a total of

$N!$ possible sequences, which necessitates a specialized approach for learning w^*.

6.4.3 Loss function for performance-based preferences

The Quadratic formulation, as described in [85], is the most commonly used method to formulate the max-margin problem. Such a formulation with constraints becomes inefficient to optimize when we have a large number of sequences with cost values significantly higher than the lowest cost sequence [102]. After computing the cost of a random sample of sequences, we found that 55% of sequences still had a cost value 70 times higher than the lowest cost sequence. Therefore, we resort to an online cost function where the constraints are absorbed into the objective function itself. Our unconstrained loss function is as follows:

$$\mathcal{L}(w) = \frac{1}{d}\sum_{i=1}^{d}\ [\ \frac{\alpha_i}{p_i}\sum_{j=1}^{p_i}(w^T\Phi_i^* - w^T\Phi_i^j)\ + \\ \beta_i(w^T\Phi_i^* - \min_{\Phi_i^j}(w^T\Phi_i^j))\ +\ \gamma_i\ (w^T\Phi_i^* - w^T\Phi_i^s)\ ] + \lambda||w||^2 \quad (6.7)$$

In Eq. 6.7, $\alpha_i, \beta_i, \gamma_i$, are hyperparameters and λ is regularization term. Φ_i^* represents the feature array for the user demonstrated sequence ξ_i^* in $\mathcal{D}$. To train the model, we generate an initial sample of sequences $x_i \in \mathcal{X} \mapsto \mathbb{R}^{p_i}$ of size p_i for i^{th} demo, as per Section 6.4.4. For the initial sample x_i, $\min_{\Phi_i^j}(w^T\Phi_i^j)\forall j \in \{1,..,p_i\}$ returns the cost value for the lowest cost sequence in x_i. The feature array for the overall minimum cost sequence ξ_i^s of the corresponding $\mathcal{G}_i$ is denoted by Φ_i^s. We minimize this loss using a generalized version of gradient descent based on sub-gradients [103]. Section 6.6 gives an intuition of every term in Eq. 6.7.

6.4.4 Learning performance-based preference

The state space of the proposed problem experiences rapid complexity growth due to the problem's combinatorial nature. For a given $\mathcal{G}_i$, there are overall $N_i!$ possible sequences. Hence, solving this problem with one-shot optimization can be inefficient. To address this challenge, we adopt a solution that begins with a nominal sample size p_i drawn from the entire population of sequences for the i^{th} demonstration. However, random sampling may result in a poor representative sample, particularly when many sequences have equivalent feature values (see Section 6.3). To generate a diverse initial sample, we use a similarity metric based on cosine similarity between feature values of a given sample ξ and expert-demonstrated sequence ξ^*. We then iteratively update this sample set with sequences with costs below a certain threshold compared to ξ^* as described in Algorithm 3.

Algorithm 3 takes the demonstration dataset $\mathcal{D}$ as input. Using $\mathcal{F}_i$ we compute the feature array $\phi(s, a)$ and initialize the corresponding graphs $\mathcal{G}_i$ for each demonstrations in $\mathcal{D}$ with w. We generate an initial sample data $\mathcal{X}$ by performing cosine similarity-based diversity sampling. This sample helps max-margin appropriately scale the cost of the truly bad sequences compared to the good ones. Since our initial sample size is a small representation of the entire sequence population, we perform learning in an iterative manner. After every iteration, we compute the first $\mathcal{K}_i$ shortest cost sequences and query $\mathcal{H}$ to compare if they are similar to ξ_i^*. We then update the sample space $\mathcal{X}$ with non-similar sequences for all $\delta_i \in \mathcal{D}$. Using such an iterative update and starting with a diverse sample helped us converge faster [104].

Algorithm 3: Performance-based Preference Learner

Input: $\mathcal{D}$
Initialize:
w: Random Initialization
$\phi(s,a)$: Compute $\phi(s,a)\ \forall \delta_i \in \mathcal{D}$
$\mathcal{G}_i$: Initialize $\mathcal{G}_i = (v,e)$ with w; $\forall \delta_i \in \mathcal{D}$
$\alpha_i, \beta_i, \gamma_i$: Hyperparameters
p_i: Initial sample size for i^{th} demo in $\mathcal{D}$
$\mathcal{K}_i$: Dataset update parameter for i^{th} demo in $\mathcal{D}$
ζ: Learning Rate
T: Total Number of Iterations
$t = 1$: Current Iteration Count
GenerateSample:
$\mathcal{X} : x_1, x_2, ..., x_d; x_i \mapsto \mathbb{R}^{p_i}, \forall i \in \{1, .., d\}$

while *(not converged)* **do**
  while $t < T$ **do**
    Compute$(C(\xi_i^*), \forall i \in \{1, .., d\})$
    Compute$(C(\xi_i^j), \forall i \in \{1, .., d\}, \forall j \in \{1, .., p_i\})$
    Compute $(C(\xi_i^s), \forall i \in \{1, .., d\})$
    Compute Loss: $\mathcal{L}(w)$ as per Eq. 6.7
    Compute Subgradient g of $\mathcal{L}(w)$
    $w \leftarrow w - \zeta g$
    reinitialize $\mathcal{G}_i,\ \forall i \in \{1, .., d\}$ with updated w
    $t \leftarrow t + 1$
  if $(C(\xi_i^*) == C(\xi_i^s), \forall i \in \{1, .., d\})$ **then**
    converged = True
  else
    Compute $\mathcal{K}_i$ shortest cost sequences in $\mathcal{G}_i,\ \forall i \in \{1, .., d\}$
    Query $\mathcal{H}$ to check for similarity between the $\mathcal{K}_i$ sequences and demonstrated sequence
    update $\mathcal{X}$ with non-similar shortest cost sequences
    t = 1
return w

Algorithm 4: Effort-based Preference Learner

Input: $\mathcal{D}$
$z(\xi_i)$: Positional features for a sequence $\forall \delta_i \in \mathcal{D}$
$\mathcal{Q} : \{\Omega_1, \Omega_2, ..., \Omega_d\}$
Initialize:
w_z: Random Initialization
$\psi^* = []$: Set of preferred sequences,
$\psi = []$: Set of unpreferred sequences,

```
for Ω_i in (Q) do
    ψ*.append(z(ξ*_i))
    for ξ_j in (Ω_i) do
        preference = evaluateℋpreference(ξ*_i, ξ_j)
        if preference then
            ψ*.append(z(ξ_j))
        else
            ψ.append(z(ξ_j))
if (len(ψ) == 0) then
    w_z = 0
else
    Learn w_z using max-margin approach with ℒ(w_z) as the loss function (Refer
     Eq. 6.8)
```

6.4.5 Learning Effort-based Preference

Algorithm. 4 depicts how we learn a penalty function based on Eq. 6.4. This penalty function is trained based on positional feature values of the starting and ending regions in a sequence ξ. A simple quadratic loss function gave us the desired results as follows:

$$\mathcal{L}(w_z) = \frac{\lambda_z}{2}||w_z||^2 \tag{6.8}$$

$$s.t. \forall i, j \ \max_{\xi_i \in \psi^*} \eta(\xi_i) + \epsilon < \min_{\xi_j \in \psi} \eta(\xi_j)$$

where, ψ^* is a set of sequences that are preferred by $\mathcal{H}$ and ψ is a set of sequences that are not preferred by $\mathcal{H}$. ϵ is a slack variable. When the expert

equally prefers all equivalent sequences, i.e., $\psi = \emptyset$, we set $\eta = 0$ for all sequences.

6.5 Data Collection

The proposed method is evaluated on the composite prepreg layup process. Chapter 7 [3] showcases how this process is performed in a sequential manner, where the task of the agent is to conform the sheet region-by-region on top of the tool. Our dataset comprises two categories: 1. Real Dataset ($\mathcal{D}_{real}$) and 2. Synthetic Dataset ($\mathcal{D}_{syn}$). $\mathcal{D}_{syn}$ is a simplified version of $\mathcal{D}_{real}$. We use $\mathcal{D}_{syn}$ for model evaluation and ablation studies, whereas $\mathcal{D}_{real}$ is used for training our model and performing physical robot experiments. $\mathcal{D}_{real}$ consists of industrial tools used in the layup process (Refer Fig. 6.3). Overall we have 10 tools each in $\mathcal{D}_{real}$ and $\mathcal{D}_{syn}$. We use a clustering algorithm based on feature attributes such as curvature, relative height, and aspect ratio of the tool [105] to transform the given tool's CAD into local regions. We then record the expert's desired sequence on all the tools in $\mathcal{D}_{real}$. In this case, we use a total of 10 feature values that are selected after consulting with the process expert for learning performance-based preferences[1]. The process expert also provides information on pairwise feature interactions. In our case, the expert suspected interaction between a region's relative height and curvature, as well as region orientation and curvature.

We validate the effectiveness of our model in generating high-quality parts by conducting physical robot experiments on two tools selected from $\mathcal{D}_{real}$, as illustrated in Fig. 6.3. While executing the layup task on these tools, we record the expert's motion and force data. We design a custom handheld tool with an embedded force sensor to capture the force data. The motion of this

[1]More details on the features can be found at: https://sites.google.com/usc.edu/irlfortasksequencing

tool is tracked via an OptiTrack motion capture system with an accuracy of $\pm 0.1mm$.

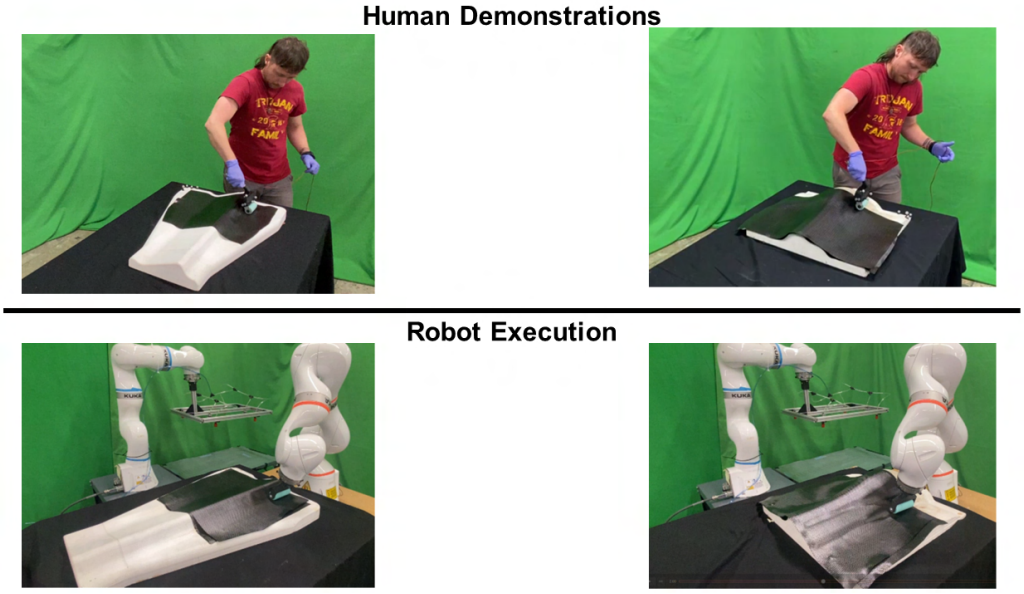


Figure 6.3: Two tools are selected from $\mathcal{D}_{real}$for collecting human motion data, and then the robot performs the same process. The tool on the left is used for training, and the tool on the right is used for testing.

6.6 Results

We perform ablation studies for our loss function on $\mathcal{D}_{syn}$ by setting a known weight value w_{syn}. The weight array w_{syn} represents a hypothetical human preference. We then evaluate the shortest cost sequence and try to recover w_{syn}. We use a 6:4 training to testing split for $\mathcal{D}_{syn}$ and $\mathcal{D}_{real}$. For $\mathcal{D}_{real}$, we train our model and perform a robot demonstration on a testing tool.

Performance-based Preference Learning Results ($\mathcal{D}_{syn}$): Fig. 6.4 illustrates how every term in our loss function in Eq. 6.7 has an impact on performance. The average term with α hyperparameter helps penalize bad sequences by ensuring a good spread of cost values for sample sequences. The

min term for the training sample with β hyperparameter helps effectively penalize sequences close to the demonstrated sequence when we update the initial sample. Lastly, the shortest-path sequence term with γ hyperparameter helps us achieve convergence in fewer updates.

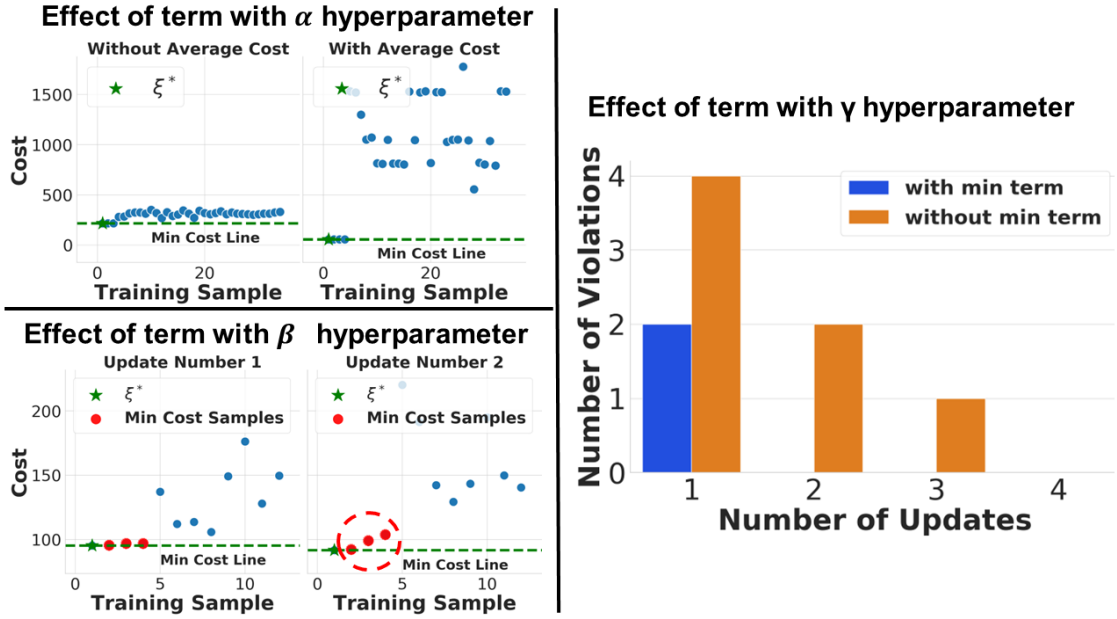


Figure 6.4: In the absence of the α term, other sequences were not appropriately scaled despite ξ^* being the lowest cost. The β term scaled the minimum cost data points properly after iterative updates, as seen from the shift in cost of red dots in Update Number 1 and 2. The γ term required more update steps for convergence. The y-axis violations represent the number of training demonstrations where the desired sequence was not the lowest cost.

Effort-based Preference Learning Results ($\mathcal{D}_{syn}$): Fig. 6.5 shows that initially, the cost for all equivalent sequences for an example tool is the same for a learned weight array w^*. Eventually, the learned η levied an appropriate penalty for the less preferred sequences by the expert. In our implementation, we use the normalized coordinates of the tool's individual region centroids as the positional features for learning. In the example in Fig. 6.5, the expert had a starting preference at a region to the right of the tool.

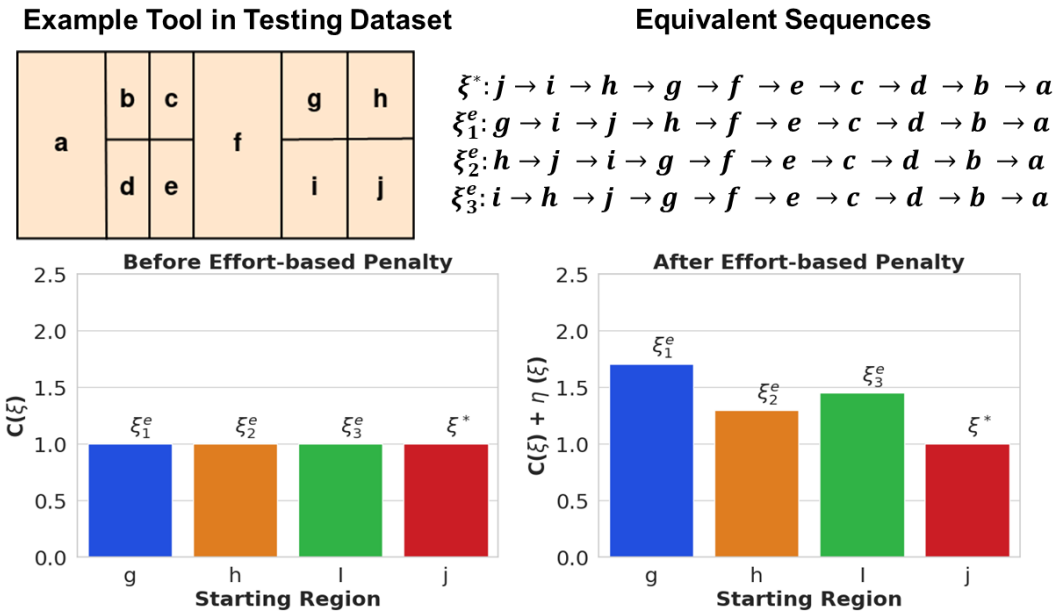


Figure 6.5: Note that $C(\xi)$ is normalized. Also we scale the costs such that η for ξ^* is 0.

Feature Interaction Coverage Results ($\mathcal{D}_{syn}$): In order to evaluate feature interaction coverage, we perform a study where we select different sets of tools from $\mathcal{D}_{syn}$ and assess their influence on ρ.

Set of Tools	{1,2,3,4,5,6,7,8,9,10}	{1,6,7,8,9,10}	{2,3,4,6,7}	{1,2,3,4,5}
Value of ρ	1.0	0.9	0.5	0.4

Table 6.1: Set of Tools means a subset of the 10 tools from $\mathcal{D}_{syn}$. The first column is for all the $\mathcal{D}_{syn}$ tools. We can see as we vary the dataset, ρ value changes, indicating varying feature interaction coverage. For more info on tools: website

Robot trials ($\mathcal{D}_{real}$): We train the model on 6 tools from $\mathcal{D}_{real}$ and test them on 4 tools. We used one of the tools on which we recorded the expert's motion and force data for training and the other one for testing. We trained on a simpler tool and evaluated our model on a complex tool. We executed the robot with motions learned from the human motion data for the sequence generated for the testing tool. To test the effect on part quality, we scanned the

human and robot laid-up parts and computed the error between the two. The error was 0.7 mm ($\pm$ 0.026mm), which is acceptable for the mentioned process [106]. The demonstration data had a ρ value of 1.0, depicting that feature interactions are captured at all levels. Our model evaluated well on both the training and testing datasets, with the demonstrated sequence consistently yielding the lowest cost for all tools in $\mathcal{D}_{real}$.

6.7 Summary

This chapter addressed the challenge of learning task sequencing policies for deformable object manipulation in industrial contexts. Task planning is critical in processes such as composite layup, surface finishing, and coating, where the sequence of operations directly influences the quality, reliability, and efficiency of the final outcome. The sequencing problem becomes particularly important when working with deformable objects, where incorrect task ordering can result in irreversible defects or increased operational complexity.

An Inverse Reinforcement Learning (IRL) framework was presented for capturing expert sequencing behavior through demonstrations and for learning interpretable task policies. This formulation supports generalization across different part geometries by modeling expert preferences based on extracted features and their interactions. The introduction of a feature interaction coverage metric provided a principled way to evaluate the representativeness and utility of demonstration datasets. Additionally, the decomposition of preferences into performance-based and effort-based components enabled more flexible and explainable policy learning.

Importantly, the proposed framework is physics-informed, as it incorporates domain-specific process knowledge and expert-derived priors about material

behavior, task constraints, and operational best practices. These implicit physical cues, captured through demonstrations and feature engineering, guide the learning process toward policies that are not only data-driven but also grounded in the underlying physics of the task.

Experimental results on real and synthetic tools demonstrated that the learned policies effectively reproduce expert task sequences and generalize to unseen configurations. The IRL framework, when trained with well-structured demonstrations and guided by task-relevant features, consistently achieved the lowest sequence cost, indicating alignment with expert strategies. These results underscore the utility of structured demonstrations and physics-informed policy learning in transferring complex skills from humans to robots.

This chapter contributes to the overarching theme of this dissertation by showing how task-level reasoning and sequencing can be learned from data, complementing the low-level manipulation models and simulation tools presented in earlier chapters. Together, these capabilities form a foundation for intelligent, human-aligned robotic systems capable of performing high-skill deformable object tasks in unstructured industrial environments.

The next chapter builds on this framework by exploring how learned simulation models and task plans can be integrated into real-time manipulation planning, enabling robots to execute adaptive actions under physical and task-based constraints.

Chapter 7

Simulation-based Grasp Planning for Deformable Objects

7.1 Introduction

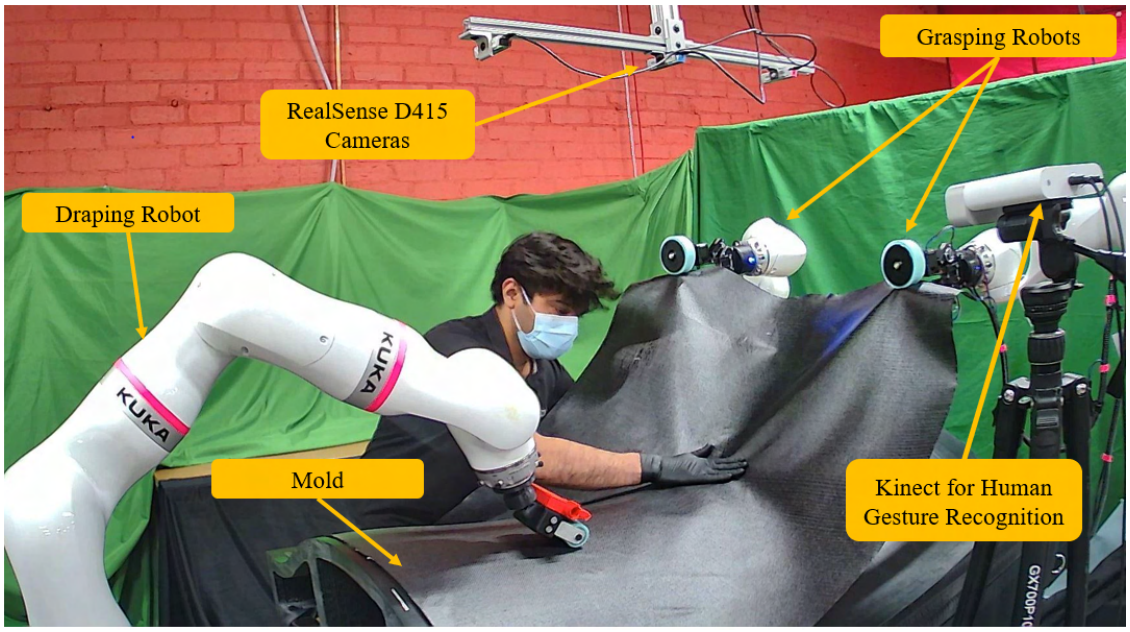


Figure 7.1: A composite sheet layup cell consisting of three robots and one human.

Planning manipulation strategies for deformable objects presents a set of challenges that differ fundamentally from those encountered in rigid-body manipulation. These challenges become especially pronounced when manipulating large, sheet-like objects, which introduce additional complications related to gravitational drooping, material sagging, and non-uniform stress distribution.

As the scale of a deformable object increases, so does its susceptibility to complex deformations. Larger sheets tend to exhibit more pronounced bending and stretching behaviors, which are often difficult to predict and even more challenging to control. The situation becomes more complex when such sheets are functionally engineered materials, such as those impregnated with adhesives, exhibiting anisotropic stiffness, or composed of multiple layers. These physical characteristics significantly influence how the sheet responds to grasping and lifting actions, and any mismatch between planned and actual behavior can result in irreversible failures.

Beyond the physical modeling challenges, task-level constraints also play a critical role in deformable object planning. In many real-world applications, it is not sufficient to simply grasp and lift an object. Robots must do so while satisfying constraints related to geometry, contact conditions, tool accessibility, and process-specific quality requirements. For example, avoiding premature contact with a target surface, maintaining tension limits to prevent damage, or ensuring that follow-on actions (such as folding, draping, or fastening) can proceed without interference. These task-aware constraints must be accounted for explicitly in the planning phase to ensure feasible and safe execution. To address these challenges, this chapter explores the use of simulation-based planning for deformable sheet manipulation. Simulation provides a flexible and scalable solution to evaluate candidate manipulation strategies entirely offline, thereby reducing dependence on costly and time-consuming physical trials. By leveraging a physics-based simulator that accurately captures sheet

deformation under various grasping and boundary conditions, robots can explore and validate grasp plans before deployment. Once a suitable plan is identified, it can be executed online using feedback-based control to adjust for real-world discrepancies.

The motivating use case for this work is automated layup of prepreg composite sheets—a critical step in manufacturing high-performance structural components. While automation in composite manufacturing has progressed significantly for tape and fiber placement on simple geometries, prepreg sheet layup remains largely manual due to its complexity. In this context, large adhesive-backed sheets must be positioned with high precision on 3D molds, often requiring multiple stages of grasping and draping. Small deviations in sheet behavior during the grasping phase can lead to defects such as bridging, wrinkles, or improper adhesion, making accurate grasp planning essential.

This chapter presents a simulation-driven grasp planning framework that utilizes a thin-shell finite element simulator to model sheet deformation and optimize grasp configurations. A state-space search algorithm is used to identify grasp points that satisfy both physical constraints (e.g., tension, drooping, collisions) and task requirements (e.g., region coverage, reachability, minimal repositioning). The system is deployed in a hybrid human-robot collaborative cell involving three robots and one human operator. To bridge the simulation-reality gap, a feedback-based intervention controller is introduced to adjust plans online based on real-time sheet tracking.

Through this approach, the chapter demonstrates how simulation-based planning— when grounded in physical modeling and informed by task constraints — can enable safe, efficient, and scalable manipulation of large deformable sheets in industrial applications.

7.2 Related Work

Work done in [22] was used to predict how a sheet will deform on a solid mold using numerical models. Numerical models were also built to predict prepreg behavior when it is grasped from specific points [107]. Kinematic algorithms to map discrete points on the sheet to a non-developable surface have also been studied [108]. Such models are then used to estimate the draping sequence, which we use as an expert input in our work. Authors in [109] reported different techniques that experts use during layup to enable using motion primitives with robotic manipulators. Researchers have also proposed cell concepts which can be used to automate the layup process in [110, 111]. However, these methods do not address the automated trajectory generation for multiple robots.

Work reported in [112] constructed a cell using an industrial robot and manually programmed it to use custom end-effectors for applying pressure and conforming the sheet. Specialized grippers for handling carbon fiber are also required since we need to prevent the sheet from adhering to gripper surfaces and sheet contamination [113, 114]. The proposed robotic cell extends the functionalities of such custom hardware for automation.

Picking and placing carbon [115] fiber sheets is another area of active research. A detailed review of pick and place operations is provided in [116]. Similarly, state-of-the-art grasping and automation technologies have been reviewed in [117]. Multi-arm manipulation of prepreg is what makes the process challenging since it requires coordination of different arms [118–127]. Robot motion plans need to be automatically generated for making the process economical. Survey papers on the manipulation of deformable objects include [128, 129]. Planning and control approaches have been developed for 1-D problems [130–137], cloth folding [138–150], and ply manipulation [151–154]. Most of these

applications define a final shape of the material, and planning and control algorithms are used to reach this desired shape. Layup, on the other hand, requires several intermediate steps to reach a desired shape on the mold. Each such step consists of applying pressure and deforming the viscoelastic material. Hence, grasp planning algorithms need to account for the underlying physics and uncertainty. Thin-shell simulations are routinely done in multiple engineering communities, and are relatively well-understood [155]. Although methods exist to tune thin-shell material properties to observations [156], thin-shell simulation alone is not sufficient for composite-sheet robotic grasp planning. This is because composite sheets must be laid in stages, and content-specific knowledge must be added to avoid damaging the sheets and ensure that real sheets are actually laid as predicted by the simulation.

Learning-from-demonstration techniques have been used to solve some challenging manipulation problems [157–159]. However, the uncertainty in the process due to changing properties of the viscoelastic material over time makes the manipulation a challenging task. Additionally, the complex interaction between draping and manipulation is also difficult to learn. Our previous work done in [79, 106, 160, 161] proposed a grasp planner that generates tool paths for simpler geometries and executes them under impedance control. In this work, we extend our previous planning algorithms by a high-fidelity physics simulator, account for the uncertainty during graph generation, and also process constraints that can accommodate newer variants of molds and larger sheets.

7.3 Problem Formulation

In this section, we formulate a multi-robot grasp planning problem for the composite layup process. Prepreg composite layup is executed in multiple

stages characterized by the number of draping zones $\boldsymbol{n}$. A subject matter expert determines these zones for the prepreg and mold, as depicted in Fig. 7.2. We represent these zones on the prepreg as RP_i and the corresponding ones on the mold as RM_i, where $i \in [1, 2, \ldots, n]$ is an intermediary draping stage. These zones are defined such that there exists a 1:1 correspondence between RP_i and RM_i $\{RP_i \leftrightarrows RM_i\}$. The layup process is thus defined as a sequential procedure of conforming the draping zones of the prepreg RP_i to the corresponding ones on the mold RM_i.

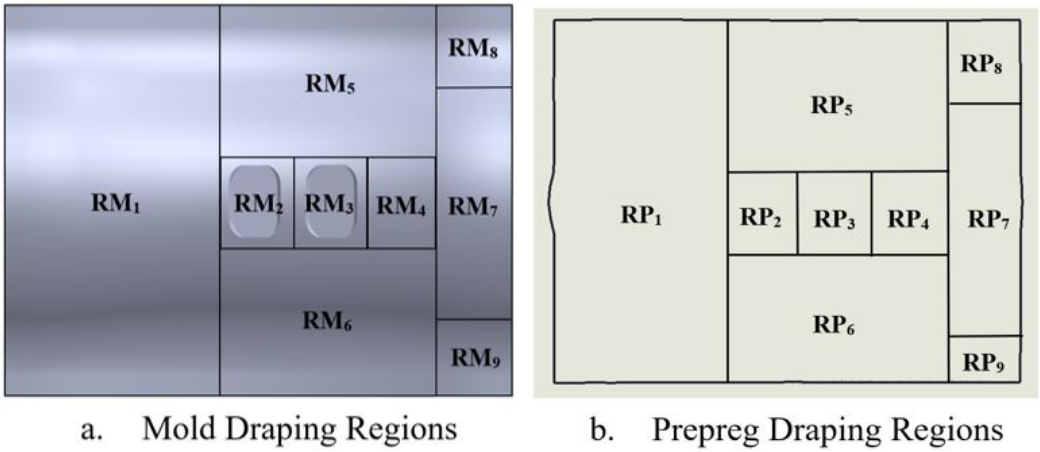
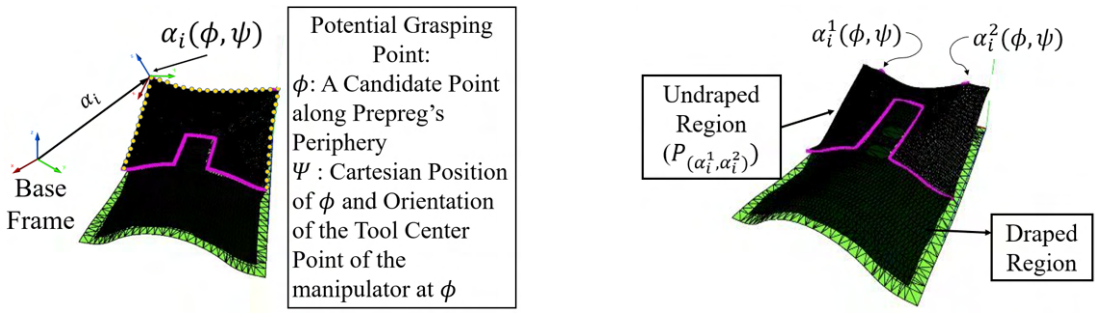


Figure 7.2: (a) Definition of draping zones on the Mold, (b) Definition of corresponding draping zones on the Prepreg Sheet.

Let us consider an intermediate stage i of the draping process. We represent the prepreg composite as a deformable surface mesh where each element of the mesh is modeled with the prepreg's material parameters. At the stage i, the draping zones $1, \ldots, i-1$ have already been conform to the corresponding regions on the mold; hence, we only need to compute the grasping locations for the remaining portion of the prepreg.

To understand how we define grasping locations, let us consider an example of the composite layup for a mold, Part A, shown in Fig. 7.8. During the layup process, a prepreg is always grasped along its periphery. Such potential grasping points along the edge of the prepreg at a draping stage i are depicted



(a) (b)

Figure 7.3: (a) Potential Grasping Location with corresponding $\{\Phi, \Psi\}$ & (b) State Space Representation of the sheet

in Fig. 7.3a. We denote these candidate points by a variable Φ, where Φ represents a potential grasping point along the prepreg's boundary at stage i. Additionally, every Φ can assume a 3D location $\{x, y, z, r, p, y\}$ * within the feasible workspace, as shown in Fig. 7.3a. We denote this 3D location of Φ by another variable Ψ. Here, Ψ represents the Cartesian position and orientation of a particular grasping point Φ. Hence, a grasping location for the prepreg gets characterized as a tuple of variables $\{\Phi, \Psi\}$. We represent this tuple by $\alpha_i = \{\Phi, \Psi\}$ that denotes a grasping location at a stage i in the draping process.

In this study, we have focused on prepregs that can be supported by only two grasping locations. We denote these two grasping locations by α_i^1 and α_i^2 as depicted in Fig. 7.3b. At a particular value of α_i^1 and α_i^2, the undraped portion of the prepreg will assume a certain configuration. We define this portion of the prepreg by $P_{\alpha_i^1,\alpha_i^2}$. Consequently, the free region of the mold at this stage, on which draping is yet to be performed, is represented as $M_{\alpha_i^1,\alpha_i^2}$. Fig. 7.3b gives an overview of these parameters for $i = 4$.

*__note__: The orientation $\{r, p, y\}$ is defined for the Tool Center Point of the manipulator that will grasp the prepreg

Therefore, the overall system's state space representation $S_{\alpha_i^1,\alpha_i^2}$ can be formulated as follows.

$$S_{\alpha_i^1,\alpha_i^2} = \{\alpha_i^1, \alpha_i^2, P_{\alpha_i^1,\alpha_i^1}, M_{\alpha_i^1,\alpha_i^2}\}, \forall i \in \{1, 2, ..., n\} \tag{7.1}$$

A particular state $S_{\alpha_i^1,\alpha_i^2}$ is considered feasible if $S_{\alpha_i^1,\alpha_i^2}$ satisfies a set of process constraints. We have identified eight such process constraints:

1. *Elastic Energy*: The elastic energy [29] represents the degree of deformation experienced by $P_{\alpha_i^1,\alpha_i^2}$ under external forces and constraints. Elastic energy exceeding a threshold value indicates that the prepreg is experiencing excessive deformation.
2. *Sheet to Mold Collision*: Collision between undraped sections of the prepreg and the mold can introduce innumerable defects. In the worst case, it might lead to the scrapping of the currently manufactured part. This constraint determines whether $P_{\alpha_i^1,\alpha_i^2}$ and $M_{\alpha_i^1,\alpha_i^2}$ are in collision.
3. *Sheet Self Collisions*: At a particular state $S_{\alpha_i^1,\alpha_i^2}$, there is a possibility that the prepreg $P_{\alpha_i^1,\alpha_i^2}$ is self-colliding. Self-collisions are undesirable in any configuration as they cause wrinkles and other major defects.
4. *Distance between the current Draping Region and the Mold*: To achieve successful draping for a zone $\{RP_i \leftrightarrows RM_i\}$, the layup technician needs to apply forces without affecting fiber alignment or overstretching the sheet. To ensure this, the distance between RP_i and RM_i should be below a minimum threshold.
5. *Distance between the Undraped Region and the Mold*: The undraped section of the prepreg should maintain a minimum threshold distance from the corresponding $\{RM_{i+1}, ..., RM_n\}$. This constraint ensures that there is no undesirable contact between the prepreg and the mold while the technician is draping a particular RP_i.

6. *Droop Factor*: Drooping is an undesirable phenomenon in the layup process, potentially leading to prepreg misalignment and self-collisions. At a system state $S_{\alpha_i^1,\alpha_i^2}$, we define the droop factor by a linear function $f(d_1, d_2)$. Where d_1 represents the maximum vertical distance between the extremities of the prepreg and $\{\alpha_i^1, \alpha_i^2\}$. While d_2 represents the maximum vertical distance between the vertices of the grasped edge and $\{\alpha_i^1, \alpha_i^2\}$, if this deformation value is larger than a certain threshold, we can conclude that there is excessive drooping.

7. *Sheet Alignment*: A typical mold for composite draping possesses demarcations that define a bounding region for the prepreg draping. Sheet alignment is the measure of undershoot or overshoot of the prepreg $P_{\alpha_i^1,\alpha_i^2}$ beyond this demarcated region.

8. *Robot Manipulability Index*: We introduce this constraint as a planning constraint rather than a process constraint. The robot manipulability index [162] is a quality measure of the closeness of the grasping manipulator to a singular configuration. This index ensures that the manipulator can transition between stages $i \rightarrow i+1$ successfully.

The states satisfying these constraints are then termed as feasible states. We represent one such feasible state at i by a variable ω_i. Additionally, we introduce a new parameter t_{i+1}^i which represents the time required for transitioning between contiguous states $\{\omega_i$ to $\omega_{i+1}\}$. The total time for the overall grasping process then gets defined by T, such that $T = \sum_{i=1}^{n-1} t_{i+1}^i$. We formulate the grasp planning problem as an optimization problem where the objective is to minimize T for the overall draping process. An optimal grasp plan Ω is thus represented by

$$\Omega = \{\omega_1, \omega_2, ..., \omega_n\}, \tag{7.2}$$

such that Ω minimizes the total time T across all possible combinations of feasible grasping locations ω_i for the entire draping process (all n draping zones).

7.4 Grasp Planning

In order to solve the multi-robot grasp planning problem formulated in section 7.3, we simulate the prepreg as a thin shell finite element model with appropriate material parameters using VegaFEM [11]. This model is employed to simulate the sheet deformation at a state $S_{\alpha_i^1,\alpha_i^2}$. The thin shell FEA simulator can compute the Elastic Energy and Droop Factor constraints. The rest of the collision and proximity constraints are evaluated using the Flexible Collision Library [163]. We use this architecture to perform constraint satisfaction across $S_{\alpha_i^1,\alpha_i^2}$ to construct a search space graph of feasible states comprising ω_i's. Subsequently, we compute a shortest-time path across the graph that defines the optimal grasp plan Ω.

7.4.1 State Space Discretization

State space representation for the grasp planning problem was outlined in Section 7.3. The states are defined by $S_{\alpha_i^1,\alpha_i^2}$. The aim is to explore the space of feasible grasping locations α_i^1 and α_i^2. The state space is discretized in order to search for feasible $S_{\alpha_i^1,\alpha_i^2}$'s. Based on experimentation, a discretization factor between 3-5% of the prepreg dimensions is used. The variables Φ and Ψ are discretized accordingly. This makes the graph construction for the composite grasp planning a combinatorial problem, which we tackle by bounding our search space.

7.4.2 Bounding the Search Space

Let us consider an input prepreg sheet P approximated as a m-sided polygon as shown in Fig. 7.4. Since P_0 is a section of P that has already been draped, we won't be considering P_0 in our heuristic design. Similarly, P_1 is the incumbent section on which draping will be performed in direction $\vec{\mathbf{v}}$. The sections $\{P_2, P_3, P_4\}$ represent the undraped portion of the prepreg as described in Fig. 7.4. Parameters $\{a_i, b_i\}$ denote the characteristic length and width of the corresponding sheet region P_i. Based on our experiments with prepreg draping, we have observed that the direction of draping $\vec{\mathbf{v}}$ and the length and width of the entire prepreg $\{L, W\}$ play a crucial role in selecting the edge along which the prepreg should be grasped. The edge, which is orthogonal to the direction of draping $\vec{\mathbf{v}}$ and is positioned at a maximum distance from P_1, is optimal for grasping the prepreg; grasping along any other edges introduces process constraint violations. If a region P_1 has multiple directions of draping, we would choose multiple edges accordingly. This discards the potential $\omega_{i,j}$ along the other edges.

In order to set the bounds on Ψ, we consider the characteristic lengths and widths of P_1 and of the set $\{P_2, P_3, P_4\}$. The selected edge is divided into two sections for $\{\alpha_i^1, \alpha_i^2\}$. In our study, for an arbitrary Ψ, we set the orientation (r, p, y) to a value equal to (r_e, p_e, y_e), which represents the orientation of selected edge $e \in \{1, m\}$ in the base frame. We introduce a bound in the cartesian space for corresponding $\Psi' s$ for $\{\alpha_i^1, \alpha_i^2\}$ as a function of the parameters $\{a_1, b_1, a_2, b_2, a_3, b_3, L, W\}$.

Additionally, we assume symmetry between the positions of the manipulators along the draping axis. This assumption is valid only if draping is performed along the central axis of the prepreg. This further constrains the search space. Hence we achieve a bounding region within which we can search for all feasible

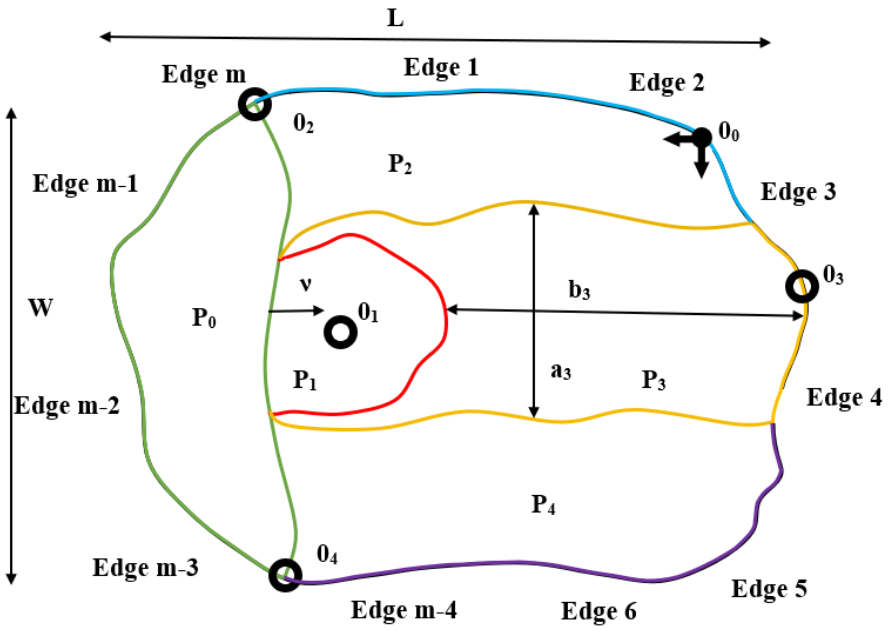


Figure 7.4: The Prepreg P is divided into different sections. P_0: Draped Section, P_1: section about to Be Draped, P_2: Left Undraped section, P_3: Front Undraped section, P_4: Right Undraped section, a_3, b_3: Characteristic length and width of the section with index 3.

grasping locations $\omega_i's$ for every draping stage $\{1, 2, ..., n\}$. Once we generate a set of appropriate $\{\Phi, \Psi\}$ for $\{\alpha_i^1, \alpha_i^2\}$ at i^{th} stage based on our heuristic, we apply the eight constraints and populate the search space graph with all the feasible ω_i. This leads to generating a search-space graph of depth n; recall that n is the total number of draping zones.

The objective of the graph-based search is to compute a grasp plan that can be executed in the minimal amount of total execution time T, as defined in Section 7.3. Currently, we have a set of nodes representing the feasible grasping locations $\omega's$. To create a complete graph $G = \{V, E\}$ we need to define and compute the cost associated with edges E.

During draping, the manipulators operate under impedance control. This is mainly done to avoid any potential deformation of the prepreg under external draping forces. As a result of the impedance control, the robots tend to move in a radial direction towards the line of draping, which is defined by the boundary between the draped and the undraped sections of the prepreg. This movement is proportional to the impedance control parameters. In order to account for this minor displacement, we add an additional layer at each i representing the displacement in the location of the robots. We denote this layer by i'. The manipulators travel to the next region $i + 1$ from this intermediate layer.

7.4.3 Graph Construction

We set the edge cost to the time required for the manipulator to travel from region i' to $i + 1$. Note that edges are added only among nodes in adjacent regions. The transition time cost of the manipulators for $i \rightarrow i'$ can be assigned to zero, as this transition does not influence the overall time for grasping actions. This work assumes that the manipulators will execute the trajectories at nearly constant velocities. They follow a linear path along the edge in case of

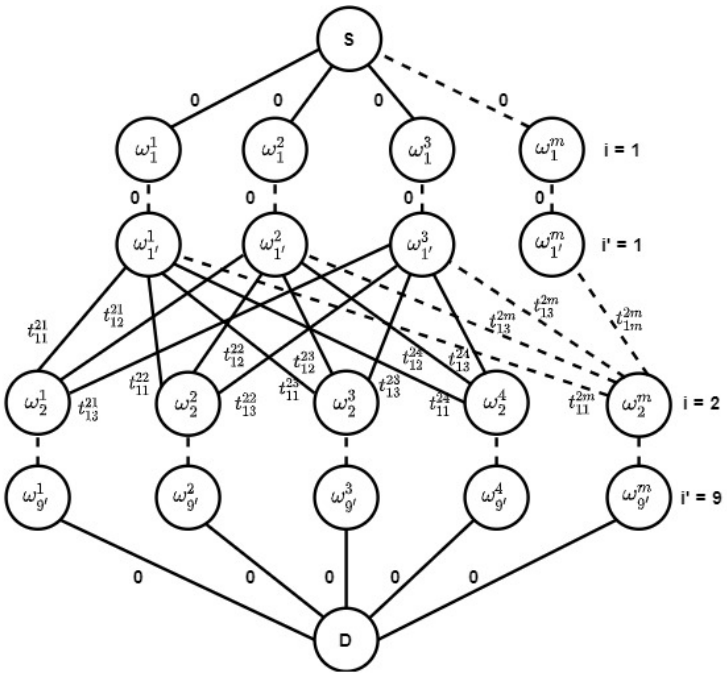


Figure 7.5: Grasp Planner Search Graph. ω_1^m is one of the feasible states at $i = 1$ and t_{11}^{21} is time taken to travel from node $\omega_{1'}^1$to ω_2^1

a change in the grasping vertex, then diagonally to the next grasping location. We compute the time $t_{i'}^{i+1}$, which gives us the edge cost and completes the construction of our search-space graph. Furthermore, to aid in finding the path with the least cost and in the spirit of dynamic programming, we add a pseudo source node connected to the first layer with zero-cost edges. Similarly, a pseudo destination node is added at the n^{th} layer with zero-cost edges. Refer to Fig. 7.5 for the structure of the search space graph.

7.4.4 Grasp Plan Generation

Once the feasible graph G is constructed, the computation of the shortest-time path becomes a shortest-path search problem. Using Dijkstra's algorithm, we compute the shortest path from the source node to the destination node. The computed path is our solution Ω, representing the optimal grasp plan for the two manipulators. Fig. 7.6 depicts the overall grasp planning process.

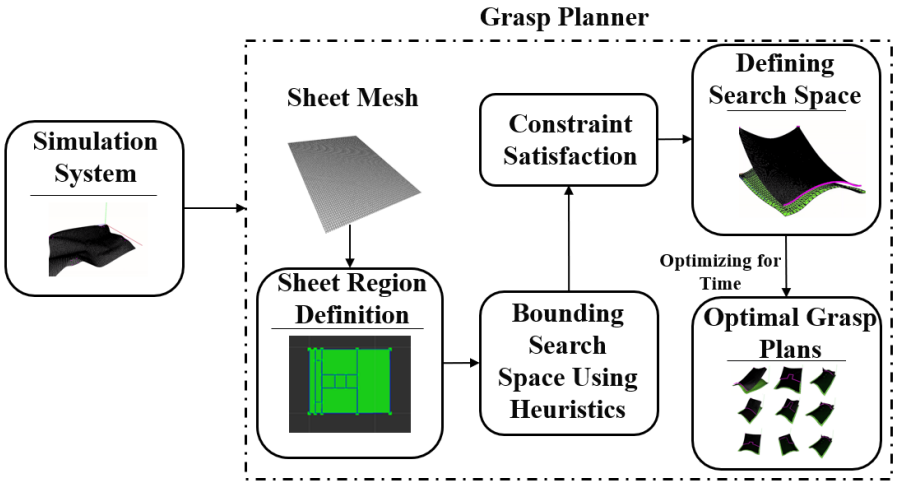


Figure 7.6: Overall Process Flow for Grasp Planning.

7.4.5 Results on Representative Examples

Our planner was successful in computing feasible grasp plans for the molds with a wide variety of complexities. Fig. 7.8 shows three representative examples. We conducted physical experiments on the mold Part A (see Fig. 7.7). Table 7.1 displays the overall time analysis of the search. The time taken for the search is directly proportional to the number of vertices and edges of the prepreg mesh and the number of discrete states across which we conduct the search.

Table 7.1: Grasp Planner Results

Type of Mold	No. of Draping Regions (n)	Plan Generation Time (secs)	Total Grasp Plan Process Time (secs)
Part A	9	945	154
Part B	6	641	122
Part C	8	236	135

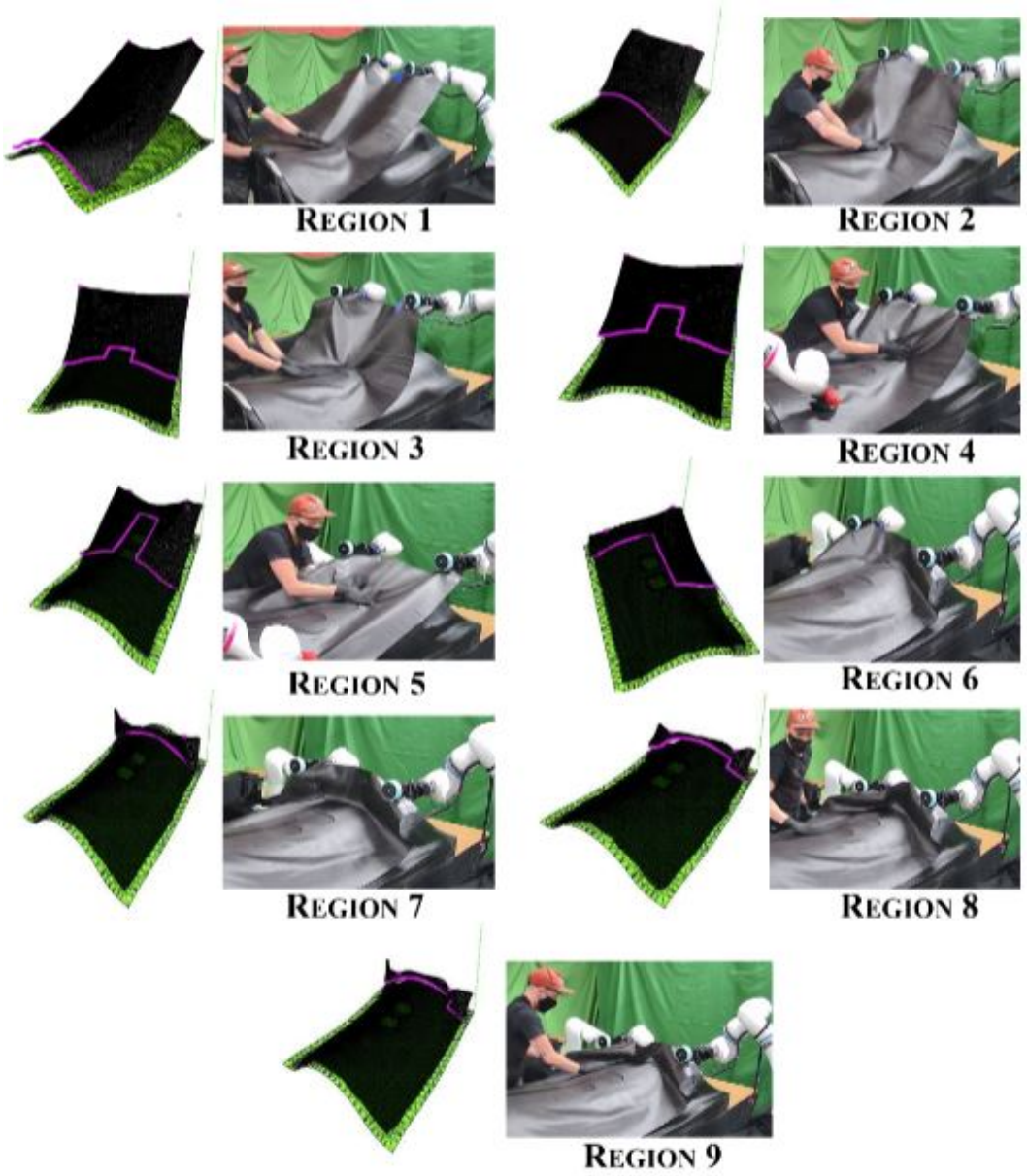


Figure 7.7: Grasping positions for the 9 draping zones in simulation and physical setup for Part A.

7.5 Intervention Controller

7.5.1 Overview

The viscoelasticity of prepreg materials lead to a change in their material properties over time, which introduces errors in the material parameter model. The plans generated from our proposed methodology in section 7.4 exhibit an inherent dependency on the material parameter model. A potential error in estimating the material parameters can lead to inaccuracies in the grasping locations. An online closed-loop system is needed to check the integrity of the prepreg draping process and raise an alert in case of deviations from the ideal planned scenario. We introduce an intervention controller system that acts as an online monitoring and verification system for the grasp plans generated by our planning algorithm.

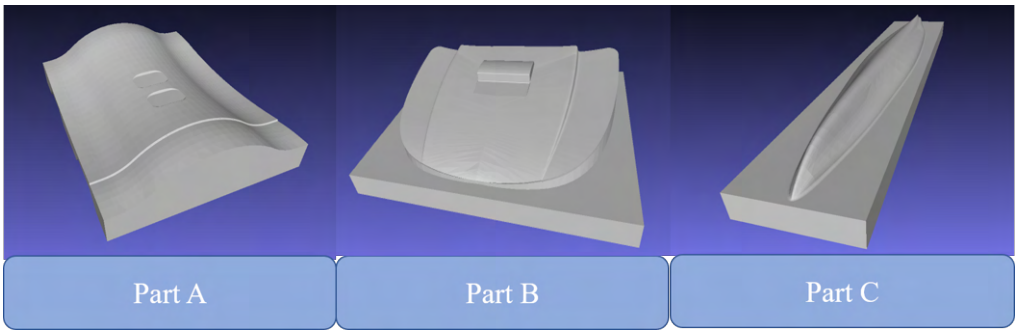


Figure 7.8: The three molds on which the grasp planner was tested. These molds vary in terms of the complexity of surface features and the draping strategy.

The sheet is tracked in real-time by employing a sheet tracking system comprising three RealSense D415 sensors. The primary function of this system is to generate a filtered point cloud of the undraped composite sheet at each grasping location. The raw point cloud data of the prepreg from each of the three RealSense D415 sensors is filtered and merged to create a unified tracked

point cloud of the composite sheet. This data can then be used to compare the simulated and observed data.

7.5.2 Constraint Violation Monitoring

In order to detect anomalous behavior, we employ a similar constraint satisfaction methodology as discussed in section 7.4. We record the prepreg's point cloud $P(t)$ at time t using the RealSense sensors. We perform constraint satisfaction on $P(t)$ for the proximity, collision, and alignment constraints by using a pre-recorded point cloud of the mold. We measure the force and torque experienced by the manipulator's grasping the sheet to monitor the Elastic Energy.

During the planning, we archive the simulation data of the prepreg at each of the feasible states in our optimal grasp plan Ω. This data comprises the constraint values for the prepreg's simulated model at every draping zone i. We compare this data against the constraints evaluated from real-time sheet tracking data to monitor and detect any constraint violations. The overall process flow of constraint monitoring is depicted in Fig. 7.9. We compute the error between the corresponding values of constraints for simulated data and the point cloud $P(t)$.

This error is a measure of the deviation of the sheet behavior at a draping zone i. If the error value exceeds a certain threshold, we trigger appropriate actions to take corrective measures (see section 7.5.3).

7.5.3 Control Actions

The intervention controller executes certain control actions based on the magnitude of the error defined in the section. 7.5.2. As discussed earlier, variations

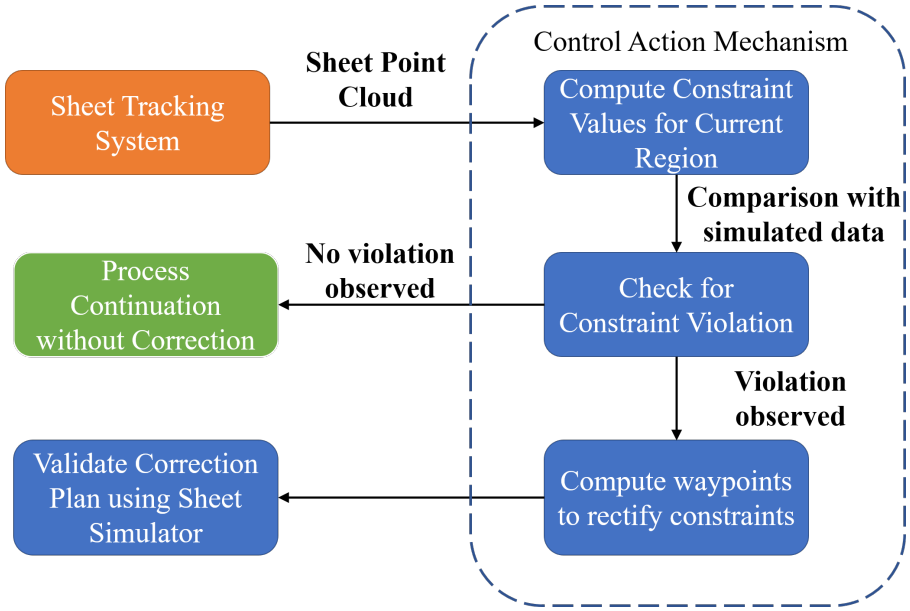


Figure 7.9: Process flow of Constraint Monitoring Method.

in the prepreg material parameter model impact the error's value. We typically classify the required level of intervention into three cases, depending on the level of inaccuracy in the material parameter model (see Fig. 7.10).

Intervention scenarios include the following:

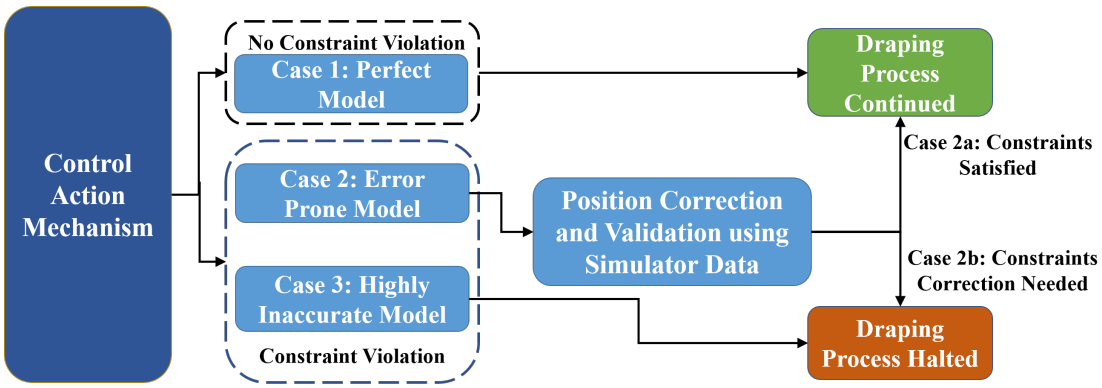


Figure 7.10: Process flow of our Intervention Controller.

- *Case 1: Accurate Material Parameter Model* In this case, the generated plans observe minimal deviations. Our grasp planner will generate feasible plans without the need for intervention.

- *Case 2: Error-Prone Material Parameter Model:* This case occurs when there is a minor discrepancy in the material parameter model. Based on the errors encountered in each of the constraints described in Section. 7.5.2 we take appropriate actions. For example, when the Droop Factor is violated, we move the robots towards the extremity of the prepreg along the grasped edge using a value proportional to the error.

- *Case 3: Highly Inaccurate Material Parameter Model:* In this case, due major discrepancy in the material model, we experience incorrigible deviations from the desirable locations. Consequently, a halt condition is triggered.

7.5.4 Results

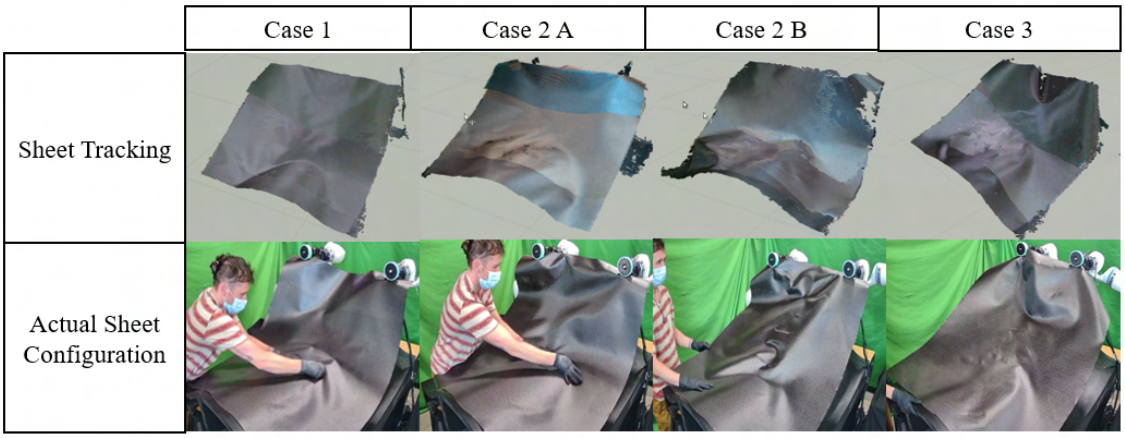


Figure 7.11: Comparison between different cases for material parameter model. Row 1 depicts the simulated data for the four control action cases, and Row 2 depicts the Real Sheet Configuration of the corresponding data.

We evaluated the grasp planning methodology on Part A (see Fig. 7.8) using the composite layup cell described in Fig. 7.1. The experiments were conducted under three different scenarios to assess the robustness of the planner and the effectiveness of the intervention controller.

In Case 1, grasp plans were generated using an accurate material parameter model (Fig. 7.11). The execution proceeded smoothly, with the intervention controller reporting no violations of the predefined process constraints. The planned grasp locations and deformations matched closely with the observed sheet behavior during execution.

In Case 2, we introduced a 10% error in the material parameter model to simulate discrepancies between simulation and real-world behavior. As expected, this led to violations in process constraints, which were detected by comparing the real-time prepreg point cloud data with the simulation predictions. Since the deviation was within the controller's acceptable threshold, the intervention controller successfully recomputed new grasp points. These updated grasp locations were validated through additional simulations and confirmed by comparing the resulting deformation with the actual sheet configuration. The error at the adjusted locations remained within tolerance, allowing the remaining draping plan to proceed with intermittent interventions across all nine draping stages.

In Case 3, the material model was perturbed with a 20% error, exceeding the controller's predefined tolerance. As a result, the intervention controller activated its halt condition, preventing further execution to avoid compromising the quality of the layup.

These results demonstrate the effectiveness of the simulation-informed grasp planner and the resilience of the intervention controller in adapting to moderate model inaccuracies, while also enforcing safe boundaries under significant deviations.

7.6 Summary

This chapter presented a simulation-driven framework for planning and executing grasping strategies for large, deformable sheets in industrial settings. By leveraging a high-fidelity, physics-based model of sheet deformation, the framework enables robots to autonomously generate feasible and task-compliant grasp plans, specifically tailored to the challenges posed by large-scale, adhesive-backed materials commonly used in composite manufacturing.

A novel state-space search formulation was introduced to explore candidate grasping configurations under realistic physical constraints. This approach integrates simulation directly into the planning loop, ensuring that deformation behavior, gravitational sag, and contact interactions are accounted for prior to execution. The resulting grasp plans were validated through physical experiments in a hybrid human-robot layup cell involving three collaborative robots and a human operator.

The grasp plans were successfully validated in a collaborative human-robot cell involving three robotic arms and a human operator. An intervention controller was developed to handle discrepancies between the simulated and real-world behavior, such as those introduced by material aging or parameter estimation error. This controller uses real-time point cloud feedback from a sheet tracking system to detect violations of process constraints and dynamically adjust the grasp strategy as needed. Experimental evaluations demonstrated the system's ability to adapt to moderate inaccuracies and halt execution safely when constraints could not be maintained.

Together, the components introduced in this chapter demonstrate how simulation-informed planning combined with closed-loop feedback control can enable the manipulation of large, high-variance deformable objects with high precision and reliability. This work contributes to the broader dissertation goal of

building intelligent, physics-informed systems capable of executing complex manipulation tasks in unstructured industrial environments.

Chapter 8

Learning the Effect of Compliance on Manipulation under Uncertainty

8.1 Introduction

(a) Satellite Assembly (b) Aerospace (c) Automotive

Figure 8.1: Examples of HMLV settings where screwdriving is performed routinely in non-gravity assisted scenarios and tight spaces. Image Courtesy: (a) `https://nanoavionics.com`, (b) `https://blog.satair.com`, (c)`https://assemblymag.com`

Previous chapters have focused on the challenges of manipulating deformable objects, emphasizing the need to model and plan around their dynamic behavior under contact, force, and environmental variability. However, deformability—or more broadly, compliance—is not limited to the object being manipulated. In fact, compliance has long been a foundational principle in the design of tools that humans use for high-precision or high-uncertainty tasks. From mechanical grippers with spring-loaded tips to hand tools like ratchets and torque-limiting screwdrivers, compliance is intentionally embedded in

tool design to improve adaptability, robustness, and ease of use in uncertain or unstructured environments.

This principle is equally evident in industrial contexts. Many tools used in manufacturing and maintenance rely on mechanical compliance—typically realized through passive elements such as springs and dampers—to absorb positioning errors, reduce force peaks, and improve tolerance to misalignments. Screwdriving, for instance, often involves rotary tools with embedded torsional springs and compliant couplings that allow engagement to occur even when there are small errors in position or orientation. These compliance mechanisms facilitate task success and play a critical role in failure avoidance during insertion.

Humans naturally exploit such compliance in tool use, enabling them to carry out tasks reliably even in uncertain or dynamically changing environments. This is especially important in maintenance or servicing tasks, where the setup is often unstructured, and the object geometry or pose may vary significantly across instances. While mass-manufacturing environments mitigate such uncertainty through expensive fixtures and precision jigs, high-mix, low-volume (HMLV) scenarios—characterized by part variability and small batch sizes—cannot justify the cost of such infrastructure (Refer Fig. 8.1). In these cases, humans leverage both passive compliance in the tool and active compliance in their motor control strategies to complete tasks reliably. With the ongoing labor shortage in manufacturing [164], there is an urgent need to transfer this adaptability to autonomous robotic systems [165].

This chapter investigates how robots can exploit compliance in tooling and control to perform screwdriving tasks in uncertain environments, particularly those representative of HMLV settings. Screwdriving is selected as the central use case due to its pervasiveness in assembly and maintenance operations, its

sensitivity to positional uncertainty, and the availability of established tool designs that incorporate compliance.

In contrast to traditional automation systems designed for structured, repeatable operations, the framework developed here supports autonomous robotic screwdriving under uncertainty (Refer Fig. 8.2 for a demonstration of the system for a servicing operation). The proposed system uses a combination of:

- A passively compliant rotary tool
- A robotic manipulator operating in Cartesian impedance control mode
- Multimodal sensing (vision and force)

Such a setup enables the robot to absorb minor misalignments and to reason over the effects of compliance during the screwdriving process. A key contri-

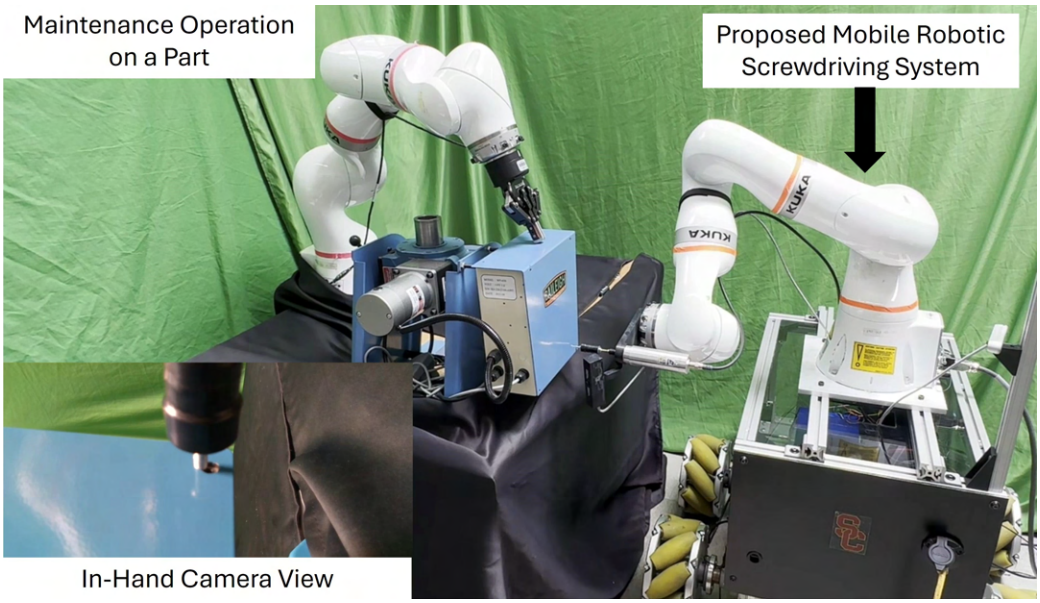


Figure 8.2: The Proposed mobile screwdriving system performing servicing operation. The image from the in-hand camera on the bottom left corner shows how the screw is offset on initial contact. The reader is advised to review the video at Video Link for better understanding

bution of this work is a physics-informed learning model that captures how

compliance—both passive and active—influences the dynamics of the screw tip during the insertion phase. Using Sparse Identification of Nonlinear Dynamics (SINDy), the system learns interpretable models of screw-tip behavior as a function of tool compliance and robot orientation. This model is not only predictive but also plays a critical role in decision-making, enabling the robot to adapt insertion strategies or trigger reattempts when necessary.

Additionally, a robust decision-tree-based failure detection mechanism is introduced to monitor and respond to process anomalies. This system uses force-based signals, rather than image-based cues, to classify failure modes and trigger corrective actions. By integrating model-based prediction with real-time sensing, the system can detect five distinct failure types and respond either by adapting the control strategy or escalating to human intervention.

The contributions of this chapter include:

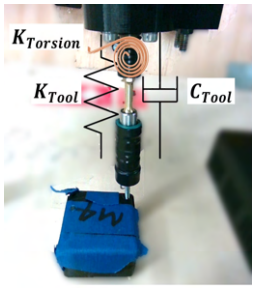
- An autonomous mobile robotic screwdriving system featuring a passively compliant rotary tool mounted on a robotic manipulator operating in Cartesian impedance control mode, equipped with 3D vision and force-sensing capabilities enabling it to perform screwdriving in high-mix, low-volume environments with significant uncertainty.
- A self-supervised, physics-informed model of screw-tip dynamics that correlates system parameters with process success rates and completion times, enhancing the system's ability to adapt and predict performance across various parts and screw types.
- A decision-tree-based failure detection system that identifies four distinct failure modes, enabling corrective actions. Additionally, we introduce a time-based failure detection mechanism that leverages the physics-informed dynamics model to determine when a reattempt at screwdriving is necessary.

Extensive experiments validate the system's ability to perform screwdriving across ten real-world industrial parts and three screw types (M4, M5, M6), including tests in non-gravity-assisted configurations. The system achieves a 100% success rate under hole pose uncertainties up to 4 mm / 3^o, with an average completion time of five seconds, highlighting its suitability for deployment in real-world, high-variability environments.

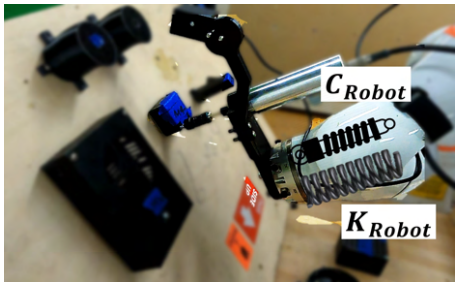
8.2 Background

The importance of compliance in autonomous screwdriving has long been recognized, with nearly all commercially available screwdriving tools incorporating some degree of compliance in their design. Additionally, active compliance strategies, such as impedance and force control, are commonly employed [166]. While compliance is a well-established concept, its precise influence on screw-tip motion, insertion dynamics, and failure modes remains underexplored. Previous work [4] provided an initial investigation into these effects, focusing on how compliance affects success rates under uncertainty. In this section, we build upon that foundation by offering a systematic analysis of both passive and active compliance, detailing their distinct roles in error compensation, failure mitigation, and operational efficiency in autonomous screwdriving.

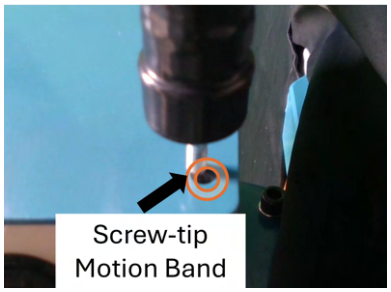
Passive Compliance: Passive compliance arises from the inherent design of the screw-driving tool and its mechanical components, such as the motor, armature, shaft, and springs. These components collectively impart compliance to the system, which can be modeled as a spring-mass-damper system. While passive compliance cannot be actively controlled or adjusted during operation, it can be characterized to understand its influence on screw-tip dynamics during the screw-driving process.



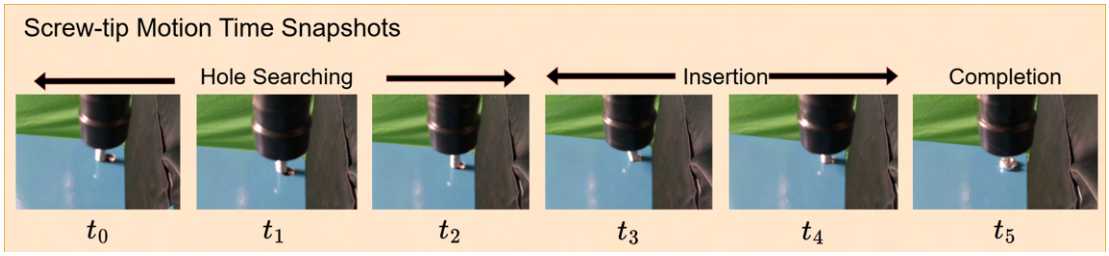
(a) Passive Compliance



(b) Active Compliance



(c) Screw-tip Motion Band



(d) Time snapshots of screw-tip with significant hole-offset

Figure 8.3: (a) Compliance in the screwdriving tool. Here, K is the stiffness, and C is the damping parameter. For the tool, there is compliance even in the torsional direction, as shown in (b) Compliance in the Agent due to impedance control. The robot's end-effector acts as a virtual spring-mass damper system. Here, K_{Robot} is the stiffness and C_{Robot} is the damping. (c) Motion Band Traced by Screw-tip, due to interaction between the compliances. Chances of success are high when this band passes through the hole's attractor basin, (d) Time snapshots of screw tip motion depict how screwdriving can be successful even in the presence of significant hole offset. At t_3, the screw tip enters the hole's attractor basin, initiating the alignment process. The robot's active compliance then facilitates correction, ensuring smooth and successful insertion despite initial misalignment.

Active Compliance: Active compliance is introduced through the use of impedance control during the manipulator's screw-driving operation. By regulating stiffness and damping parameters in the Cartesian impedance controller, the system ensures safe interaction with the part by avoiding excessive forces. These tunable parameters significantly influence the screw-tip dynamics, allowing precise adjustments to the system's compliance behavior and improving adaptability in high-uncertainty environments.

The interaction between passive and active compliance results in a characteristic motion pattern of the screw tip upon contact with the surface, forming what we define as the "screw-tip motion band." This motion band represents the bounded region within which the screw tip moves as a consequence of the combined effects of compliance in both the robot and the environment (Refer Fig. 8.3). The shape and extent of this band are influenced by factors such as impedance parameters, contact forces, and the screwdriving dynamics. Understanding this motion band is crucial for accurately modeling screw-tip behavior, predicting insertion success, and designing corrective strategies for failure recovery.

8.3 Related Work

8.3.1 Robotics and Automation in Screwdriving

Screwdriving is a routine yet crucial industrial operation, and numerous automation methods have been explored over the years [167–169]. Advances have been made in various aspects, including smart end-effector design [170, 171], part-feeder and automated gantry systems [167, 172, 173], visual servoing [174], and vision-based screwdriving systems [175, 176]. Additionally, some research has explored robotic manipulation of hand tools designed for

human use [177]. However, very few studies have systematically analyzed the underlying factors that contribute to successful screwdriving or attempted to quantify them. Moreover, research on screwdriving under uncertainty, particularly in high-mix, low-volume (HMLV) manufacturing, remains limited. In [178], the authors propose a screwdriving approach for uncertain environments, but their focus is on servo-controlled torque estimation without external sensors. Similarly, [179] presents a vision-based approach for operating in semi-structured environments, yet it requires extensive calibration and does not explicitly quantify the effects of uncertainty. Other works, such as [180, 181], have addressed screwdriving for maintenance applications. However, [180] primarily focuses on simulation, while [181] examines a non-standard screwdriving task specific to a single screw type. A comprehensive survey by [166] provides an overview of automated thread-fastening systems, formalizing key terminologies that we adopt in this work. Among the critical factors in screwdriving, compliance has long been recognized as essential for successful operation [182–184]. While several studies have leveraged compliance in screwdriving, no prior work has quantitatively analyzed its impact on success rates. More recently, research efforts have focused on developing robotic frameworks and policies for manipulating human-centric screwdriving tools [172, 177, 185]. However, numerous robot-compatible screwdriving tools are commercially available that could enhance system resilience and efficiency. While previous works have demonstrated the significance of robotic screwdriving across various applications, the proposed work's focus is on quantifying the effects of compliance and uncertainty on success rates, studying different screw types, and investigating applications in HMLV settings.

8.3.2 Defect Detection for Screwdriving Operations

In Section. 8.6, we've defined the five key modes of failure that we're interested in detecting alongside the nominal operation mode. In a high-speed operation such as screwdriving, detecting failures is quite important. Naturally, extensive research has been dedicated to failure detection, [186–189], with a strong emphasis on using wrench signals as the primary sensing modality. However, most prior work has been limited to detecting only a small subset of failure modes. In previous work [4], a multi-modal deep learning approach that fuses vision and wrench data to classify two failure modes was explored. While learning-based methods offer advantages, particularly in structured environments with low uncertainty, they are often highly sensitive to environmental variations and require large amounts of training data to generalize effectively [190–192]. Similar trends can be observed in recent data-driven anomaly detection methods [193–195], which have demonstrated strong performance under controlled conditions but often struggle with scalability and robustness in real-world deployments, especially in high-mix, low-volume (HMLV) settings where uncertainty is high. To address these challenges, we seek a more scalable and generalizable failure detection approach. Incremental learning techniques, such as the one proposed in [196], have been explored for screwdriving applications, but they primarily focus on binary anomaly detection-determining whether a state is normal or anomalous-rather than providing a fine-grained classification of failure modes. Similarly, Hidden Markov Models (HMMs) have been proposed for failure detection [197], leveraging screwdriving process mechanics and a stage transition graph to improve generalization across screw types with minimal labeled data. While HMMs offer an efficient way to model process stages, their reliance on structured stage transitions makes

them less effective in high-mix, low-volume (HMLV) settings, where screwdriving processes often deviate from expected sequences. Additionally, HMMs require sequential inference over multiple time steps, which introduces computational overhead and limits their suitability for real-time failure detection in high-speed screwdriving. In contrast, Decision tree-based methods have been successfully utilized for various tasks in high-mix manufacturing [198], demonstrating their effectiveness in handling diverse and dynamic production scenarios. Authors in [199] demonstrated the advantages of using a decision tree to detect failures for one screw type in the presence of only passive compliance. The work proposed in this chapter draws inspiration from them, where we build upon the decision-tree-based framework for force-based failure detection. Furthermore, the proposed approach extends this decision tree-based methodology by introducing a more refined feature set and allowing for generalization across different screw types (Refer Section. 8.6 for details). Additionally, the data augmentation strategy enables effective failure detection with substantially lower data requirements, making the proposed approach more suitable for real-world HMLV manufacturing environments.

8.3.3 Dynamics Modeling for Screw-tip Motion

The dynamics of the screw tip studied in this work exhibit highly non-linear behavior, primarily due to the interplay between compliance modes and external forces. The primary objective in modeling the screw-tip dynamics is to characterize the effect of compliance on its motion rather than achieving a complete predictive model of the entire screwdriving process. Recent advances in learning-based methods have unlocked new possibilities for modeling such complex, non-linear dynamics [200, 201]. Over the past decades, data-driven approaches have demonstrated remarkable success in capturing intricate object dynamics [202, 203]. However, most neural network-based models require large

amounts of training data and often struggle with generalization over long time horizons. Additionally, black-box learning models lack explainability, making them unsuitable for high-stakes industrial applications such as aerospace manufacturing, where reliability is critical, particularly in high-mix, low-volume (HMLV) settings. Conversely, purely analytical physics-based models, which define dynamics using explicit mathematical formulations, offer advantages in terms of data efficiency and interpretability. However, these models often fall short when attempting to represent extreme non-linear behaviors, such as those encountered in real-world screwdriving operations.

Recently, physics-informed machine learning approaches [204–210] have emerged as a promising alternative, bridging the gap between purely data-driven and fully analytical methods. These hybrid techniques integrate physical priors with learning-based frameworks, allowing for better generalization with a reasonable amount of training data while preserving a degree of explainability. This makes them particularly well-suited for industrial automation and applications where insight into the underlying dynamics is necessary. In this work, the proposed method takes inspiration from Sparse Identification of Nonlinear Dynamical Systems (SINDy) [204, 210, 211], which has demonstrated superior performance compared to both purely data-driven and purely analytical approaches.

Despite prior work on screwdriving process modeling using analytical methods [169, 212] and recent advancements in physics-informed techniques applied to industrial automation [6, 213, 214], no prior research has specifically examined screw-tip dynamics in the presence of compliance. The exploratory work [4] was the first to explore this phenomenon, introducing the concept of the screw-tip motion band and demonstrating preliminary insights into its behavior. However, that study was limited in scope, focusing on only a few compliance configurations and simplified modeling assumptions. In this work,

we build on that foundation by developing a more systematic framework for analyzing screw-tip motion under compliance. The proposed approach extends previous findings by incorporating a broader range of compliance effects and refining the modeling methodology. By integrating physics priors with data-driven insights, we aim to provide a more generalizable and interpretable understanding of screw-tip behavior, which could contribute to improved automation strategies in screwdriving applications.

8.4 System Overview

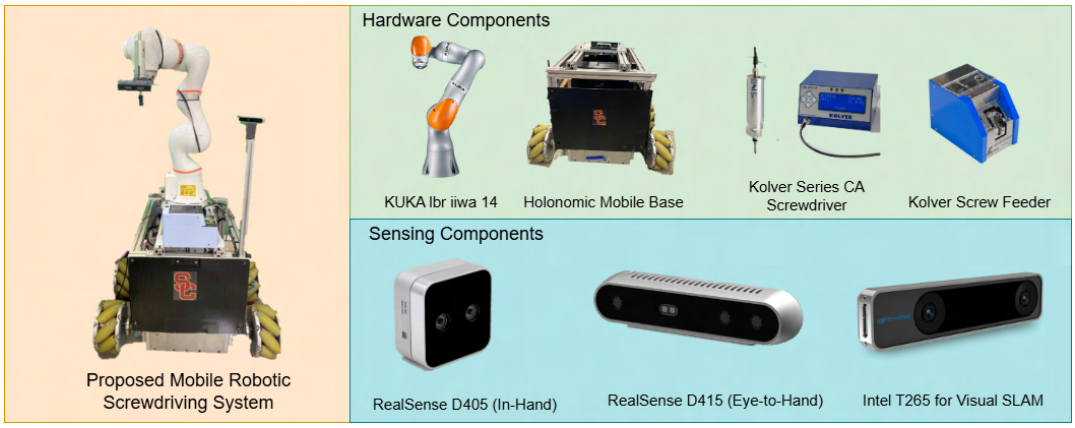


Figure 8.4: Hardware System Components of the mobile manipulation-based screwdriving system.

8.4.1 Mobile-Manipulator-based Robotic Screwdriving System

The proposed system is designed specifically for high-mix, low-volume (HMLV) settings where estimating the hole pose is inherently uncertain due to the absence of fixtures, as discussed in Section 8.1. The primary focus is on screwdriving operations in assembly and maintenance/service contexts, which involve variability in both part sizes and screw types. To successfully operate

in such settings, the system must be mobile, capable of autonomously maneuvering across parts, and able to navigate constrained environments, such as manufacturing floors or workshops.

Taking these requirements into account, we propose a system comprising a 7-DOF manipulator mounted on a holonomic-drive mobile base (Refer Fig. 8.4). The manipulator is a KUKA LBR iiwa, equipped with joint-torque sensors that provide force feedback. This robot also features an inherent Cartesian impedance controller, ensuring safe interaction with parts by preventing damage during contact with the surface for screwdriving operations.

For the screwdriving tool, we use an industrial-grade Kolver Series CA screwdriver, which employs a magnetic holder to accommodate screws with various head types. Screws are reliably supplied by a Kolver NFK UNI Screw Feeder, which is mounted on the mobile base in a fixed location to enable efficient tool pick-up by the robot during operations.

The system also incorporates two Intel RealSense cameras for perception. The first camera is mounted on the mobile base (eye-to-hand configuration) to localize the part within the scene. The second camera is mounted on the robot's flange (eye-in-hand configuration) and provides precise hole pose estimation during the operation. Both cameras provide RGB and depth images, enabling the system to operate under uncertain and variable conditions.

Finally, the tool and the eye-in-hand camera are mounted on the robot's flange using a custom 3D-printed fixture, ensuring proper alignment and functionality. This comprehensive setup enables the system to reliably perform screwdriving operations across a wide variety of parts and screw types, even in the presence of significant uncertainties (Refer Fig. 8.4).

8.4.2 Software System Architecture

The software architecture of the proposed robotic screw-driving system is structured around three core modules that enable successful operation, from screw pick-up to insertion: (1) Planning and Control, (2) Perception and Sensing, and (3) Failure and Fault Detection. Each module plays a crucial role in providing information about the robot's state, guiding the decision-making process, and ensuring robust execution. The screwdriving process is modeled as a finite state machine (FSM) that transitions through various states, starting from screw pick-up and progressing to insertion (Refer Fig. 8.5). This FSM framework enables precise handling of failures and faults, ensuring seamless recovery and adaptability. The specific contributions of each module to the FSM's transitions, from the initial state to the goal state, are detailed below.

8.4.2.1 Planning and Control

The sequence of operations for the manipulator in the proposed robotic screw-driving system involves three key stages: screw pick-up, transportation to a pre-insertion configuration, and screw insertion. The manipulator begins by picking up the screw from the screw feeder, then transports it to a pre-defined position above the target hole, and finally approaches the hole to perform the insertion. The initial steps, from screw pick-up to the pre-insertion position, involve no variability in execution, allowing us to pre-compute the motion trajectories and plans. For motion planning, we leverage MoveIt 2, which utilizes OMPL-based planners [215]. Specifically, we use the Anytime Path Shortening planner with constraints on end-effector orientation and joint velocity. Communication with the robot is facilitated by the LBR-FRI ROS 2 stack [216], enabling real-time control in KUKA's fast robot interface mode. Planning is done in Cartesian space, and the robot is commanded with timestamped

trajectories via the Fast-Robot Interface that the LBR-FRI stack uses by communicating with the controller over the User Datagram Protocol (UDP).

Although this work primarily focuses on the screwdriving operation, we provide a detailed overview of the system's end-to-end functionality to ensure real-world applicability. A critical component of the process is ensuring reliable screw pick-up from the screw feeder. This is achieved by approaching the screw head at low velocity while running the screwdriving tool in a counterclockwise direction. Through experiments, we determined that executing this deterministic motion for at least 5 seconds consistently achieves a 100% success rate in screw pick-up.

For better precision, we use an ArUco marker strategically placed on the part at a calibrated location, enabling accurate hole-pose estimation with minimal uncertainty. Based on this estimate, we compute a motion plan to approach the hole with the screw-driving tool activated. The screw-driving operation itself is performed under Cartesian impedance control mode to minimize the risk of part damage. We use KUKA's built-in Cartesian impedance controller for this purpose. The screwdriving tool, a Kolver Series CA tool, is controlled via ROS Serial communication using an Arduino, allowing for commands to adjust velocity, rotation direction, and start/stop functionality.

The base movement and control are performed via a custom stack developed for the mobile base; all mobile base control-related information can be found at `https://github.com/RROS-Lab/`. The entire software stack is built on ROS 2 [217], which serves as the middleware for seamless communication across the system. This robust planning and control framework ensures the precise and reliable execution of the screw-driving process.

8.4.2.2 Perception and Sensing

Vision: The proposed system relies on two external RealSense cameras, which provide aligned RGB and depth frames at a frame rate of approximately 30 fps. These cameras are essential for perception tasks, with distinct roles in the system. The eye-to-hand camera (RealSense D415) is mounted externally and is responsible for part localization (i.e., detecting ARUCO markers), allowing the robot to identify and position itself relative to the target part. The in-hand (or eye-in-hand) camera (RealSense D405), mounted on the robot's end-effector, plays a crucial role in generating time-series data that captures the dynamics of the screw tip as it interacts with the surface. Both cameras stream their RGB and depth data over ROS 2 topics, facilitated by the RealSense2 ROS package, ensuring smooth integration into the robotic system.

Robot Proprioception: The KUKA LBR iiwa 14 R820 robot enhances perception through its joint-torque sensors, which provide rich proprioceptive feedback. Using the KUKA API, the robot reports estimated Cartesian forces at the end-effector. Additionally, the robot provides the precise position of its end-effector relative to the base, enabling accurate tracking of its movements.

This proprioceptive data is available at a frequency of approximately 500 Hz, offering high-resolution sensory feedback. The LBR-FRI ROS 2 stack is used to retrieve and integrate this information into the system, ensuring reliable and real-time communication.

8.4.3 System Operation

In addition to the software and planning components discussed in previous sections, our system comprises two key modules: (1) the Screw-Tip Dynamics Model and (2) the Defect Detection Model. Each module plays a crucial role in process planning. Our system can be viewed as a finite-state machine (Refer

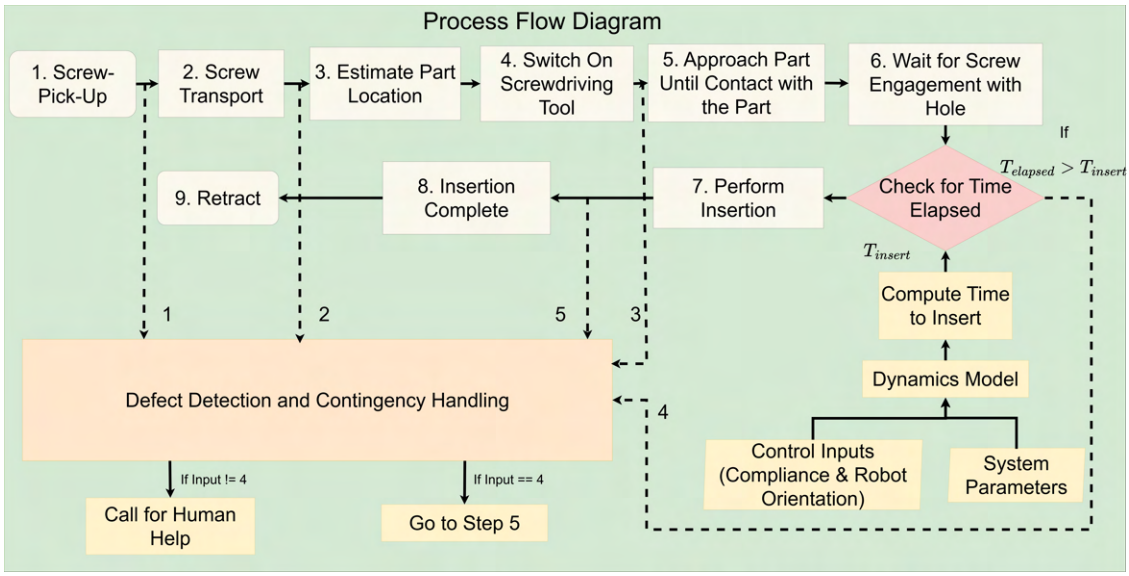


Figure 8.5: We demonstrate how our dynamics model and defect detection module aid in decision-making for our system. Here we depict the entire process flow from the start (screw pick-up) till the end (insertion) for our system. The Numbered Blocks at the top are nominal operation modes. At every stage, the defect detection module evaluates if a failure has occurred or not. If failure occurs due to time elapse (Input 4), then a reattempt strategy is triggered, or else we call for human help.

Fig. 8.5), where high-level decision-making is required to perform screwdriving efficiently while minimizing failures.

Currently, when a defect is detected, an alert is triggered, prompting human intervention for failure recovery before the system resumes nominal operation. However, in an HMLV scenario, uncertainty in hole pose estimation can cause the screw tip to keep moving in a motion band described in Section. 8.2 without successfully latching onto the hole. To handle such contingencies, our dynamics model can be leveraged to compute an upper time bound (T_{insert} in Fig. 8.5) for remaining in this state. If the screw tip does not engage within this threshold, the system can autonomously reattempt screwdriving by retracting and repositioning rather than relying solely on external intervention. We demonstrate how every piece of our system interacts to enable this in Fig. 8.5.

By integrating the screw-tip dynamics model into our system design, we can fine-tune state transitions within our finite-state machine. Specifically, this enables us to define time thresholds for transitioning from an unsuccessful insertion attempt to a reattempt strategy, reducing unnecessary dwell time in failure-prone states. Sections 8.5 and 8.7.3 further elaborate on these concepts and provide quantitative analyses of their impact.

8.5 Physics-Informed Discovery of Screw Tip Dynamics

As discussed in Section 8.2, the interplay between passive and active compliance results in the screw-tip executing motion within a bounded region. If the hole lies within this band of motion, the operation remains feasible even when the screw tip initially makes contact away from the hole. While this motion is inherently non-linear, it is fundamentally governed by the physics of compliance. Our goal is to characterize this non-linear behavior by learning

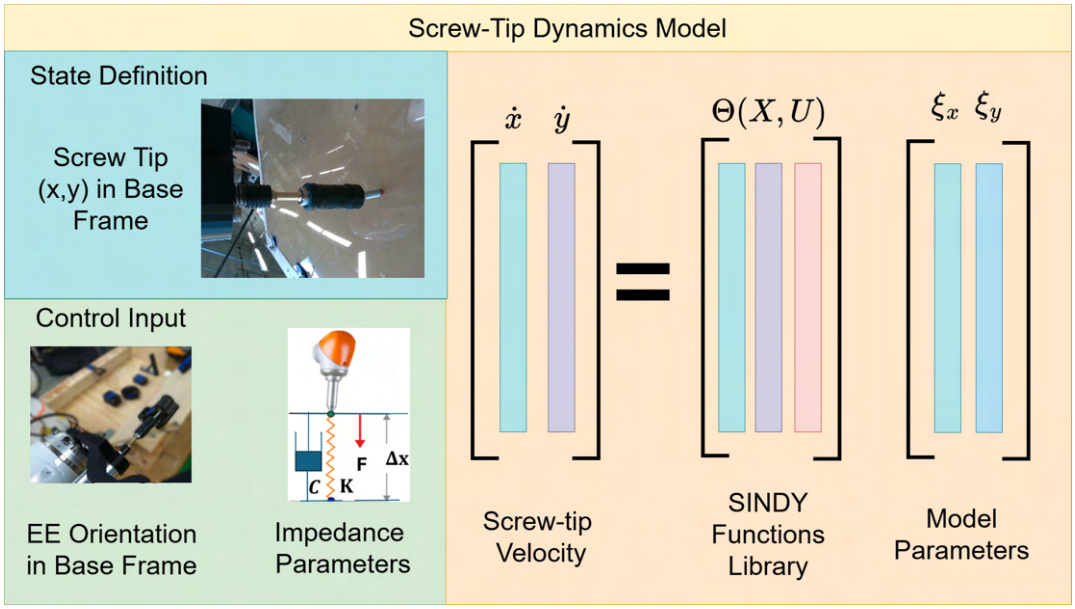


Figure 8.6: Screw-tip Dynamics Model that takes the state information, impedance control parameters, and robot orientation as input. The model predicts the first-order time derivatives, i.e., the velocity of the screw-tip in Cartesian space, in the robot's base frame of reference. We achieve this by first converting the screw-tip coordinates from image space to camera frame and then to base frame.

a dynamics model of the screw tip. Such a model would elucidate the effects of compliance on the screw-tip's trajectory and provide actionable insights for system design.

The primary motivation for learning such a model lies in its utility for process planning, particularly in determining when to abandon an operation. For instance, consider a scenario where the screw-tip has made contact with the surface and is following the motion described in Section 8.2. Suppose we can accurately predict the screw-tip's position over time. In that case, we can probabilistically estimate the time required for the screw to fall into the hole's attractor basin (Refer Section. 8.7.3). This estimated time can then serve as a reference threshold, guiding the system on when to reset and reattempt the operation if successful insertion has not occurred within the expected duration, thus underscoring the importance of such a dynamics model.

There are multiple approaches to learning this dynamics model for given compliance parameters. A fully data-driven approach, such as employing neural operators or networks, is one option. However, these methods often lack explainability, require extensive time-series data for training, and struggle to generalize to new scenarios or varying operating conditions [218]. In applications with high-mix, low-volume (HMLV) constraints, where data availability is limited, these shortcomings become significant barriers.

To address these challenges, we adopt the Sparse Identification of Nonlinear Dynamics (SINDy) framework. SINDy directly identifies the governing equations of screw-tip dynamics by leveraging physics-informed assumptions. Specifically, we hypothesize a family of candidate functions to describe the dynamics and use sparse regression techniques to learn their parameters. This approach ensures explainability and requires minimal retraining, making it

well-suited for systems that must operate under uncertainty and adapt to changes in operating parameters.

8.5.1 Model Definition

Preliminaries: In order to formally define the dynamics modeling problem, let us initially define a set of parameters and variables. Let $X \in \mathbb{R}^2$ denote the state of the screw tip. Let $\dot{X}$ denote the first-order time derivative of X. We only consider screw-tip dynamics once the tip has made contact with the surface of the panel on which screwdriving is to be performed. Hence, t = 0 signifies the time instant at which the rotating screw tip makes contact with the surface and executes the motion band. At a given time t, the system is excited by an external control input $U \in \mathbb{R}^{15}$. The objective of the dynamic model is to learn a function $\dot{X} = f(X, U)$. We also define Θ that denotes a family of candidate functions comprising polynomial and sinusoidal functions, in our case, that potentially define the dynamics of the screw-tip.

SINDy for Screw-tip Dynamics: In our system (Refer Fig. 8.6), X is defined by the $[x, y] \in \mathbb{R}^2$ cartesian position of the screw-tip with respect to a fixed frame of reference T_{base}, i.e., the robot's base. We ignore z since the screw-tip performs the described motion on a plane, i.e., the surface on which screw-driving is performed. Consequently, $\dot{X}$ gets defined by the tuple $[\dot{x}, \dot{y}]$. The control inputs U are a function of the controller's compliance parameters and the robot's state. Specifically, U is the robot's end-effector orientation $\mathcal{Q} \in \mathbb{R}^3$ and the Cartesian impedance controller parameters $K \in \mathbb{R}^6$ and $C \in \mathbb{R}^6$. Where K is the stiffness and C is the damping that decides the controller gains. Now, to model $f(X, U)$, we employ SINDy methodology, such that for a given family of candidate functions Θ, our objective is to learn parameters Ξ, such that the following can be satisfied.

$$\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X}, \mathbf{U})\boldsymbol{\Xi} \tag{8.1}$$

Here, $\mathbf{X} = [X_1, \ldots, X_m] \in \mathbb{R}^{m \times n}$ represents a collection of m time snapshots of the state variable $X \in \mathbb{R}^n$, with $\dot{X}$ denoting the corresponding time derivatives. In our case, we consider $n = 2$. The term $\boldsymbol{\Theta}(\mathbf{X}, \mathbf{U}) = [\theta_1(\mathbf{X}, \mathbf{U}), \ldots, \theta_p(\mathbf{X}, \mathbf{U})] \in \mathbb{R}^{m \times p}$ defines a set of p candidate basis functions, while $\boldsymbol{\Xi} \in \mathbb{R}^{p \times n}$ represents the coefficient matrix corresponding to these functions in $\boldsymbol{\Theta}$. The primary goal of the model is to estimate the coefficient parameters in $\boldsymbol{\Xi}$.

In order to learn these coefficients, we optimize for the loss function as per Eq. 8.2.

$$\mathcal{L}_{dynamic} = \lambda_1 ||\dot{X} - \boldsymbol{\Theta}(X, U)\ \Xi||_2^2 + \lambda_2 ||\Xi||_2^2 \tag{8.2}$$

Here, λ_1 and λ_2 are hyperparameters that control different aspects of the model. The λ_1 term corresponds to the Mean Squared Error (L2-loss) function, which drives the minimization of the error between predicted and observed values. Meanwhile, λ_2 enforces regularization, helping to prevent overfitting.

Additionally, we incorporate sequential thresholding, where for every i^{th} training step, we set coefficients below a predefined threshold ϵ to zero. This encourages sparsity in the model, ensuring that only the most significant terms with dominant coefficients are retained. For $\boldsymbol{\Theta}$, we construct a feature library consisting of polynomial terms up to the third order, along with sinusoidal functions.

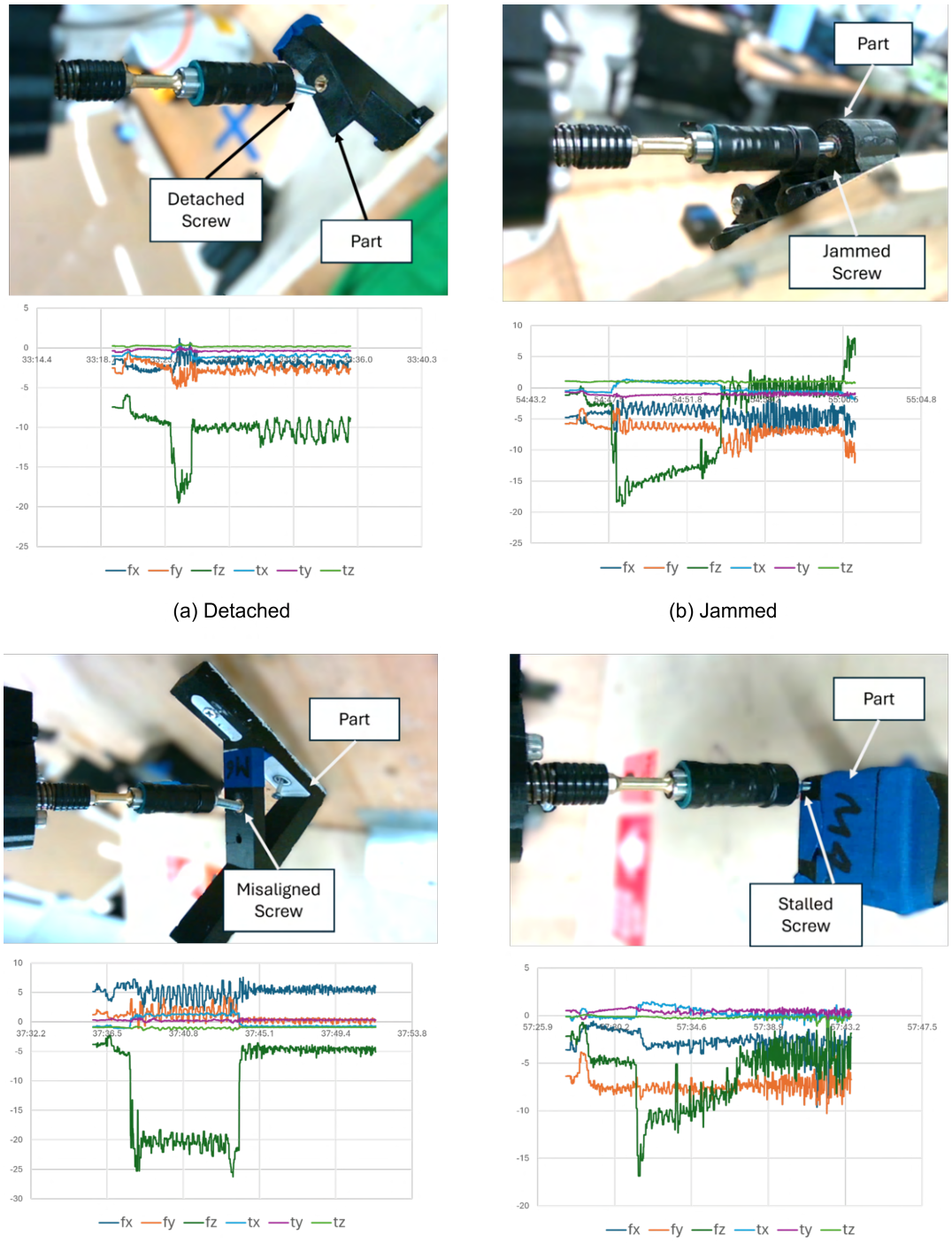


Figure 8.7: The four different failure modes studied in this work. We can observe that for each of them, the wrench signals have distinctive characteristics.

8.6 Failure Mode Detection

Defect detection is a critical component of any industrial robotic system, ensuring resilience and the ability to handle various failure modes effectively. In a robotic task execution framework, the system relies on a set of controllers to perform motions, receiving post-condition feedback to determine task success. A failure mode is identified when these post-conditions are not met, indicating an issue in execution.

In the context of screwdriving, failure detection follows a similar principle. A successful insertion can be defined based on specific post-condition metrics, such as wrench signals and the robot's final position upon task completion. Through analysis, we observe that the primary failure modes in screwdriving can be characterized as functions of the external wrench (Refer Fig. 8.7). Throughout the screwdriving process, from screw pickup to final insertion, multiple failure modes may arise. To ensure smooth operation and enable efficient failure recovery, we develop a defect detection system capable of identifying five key failure modes:

- Stalling of Operation - Occurs due to equipment malfunction, particularly when the system fails to execute a controller properly.
- Detached Screw - Happens if the screw detaches from the tool-head during the process
- Thread Jamming - Arises when excessive friction builds up due to improper thread engagement, potentially leading to damage or deformation, thereby preventing further insertion.
- Misaligned Screw - A failure mode caused by improper screw pickup, leading to unsuccessful insertion as the screw-driving tool loses control

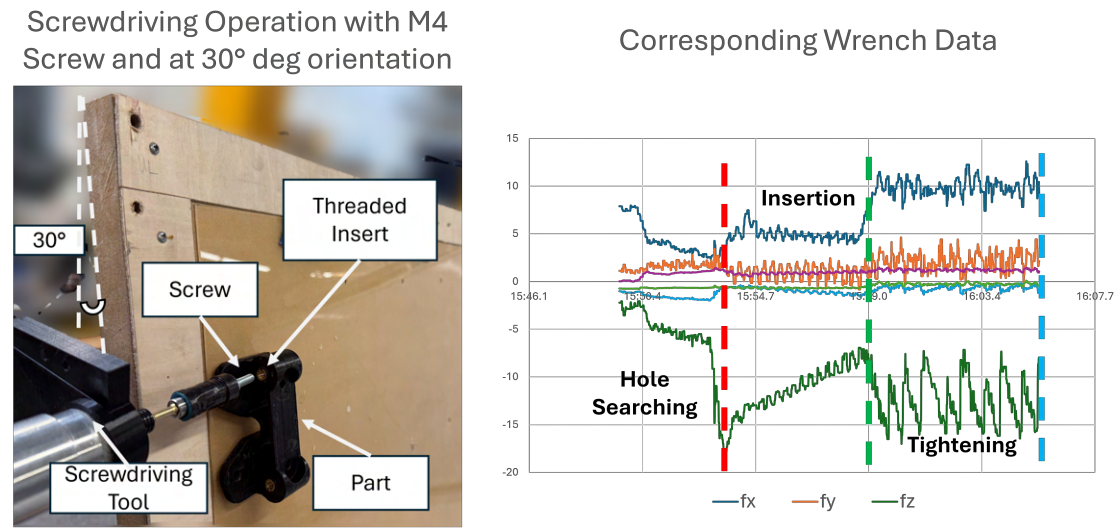
over the screw's motion. This is also the main cause of cross-threading [166]

- Timeout Failure: This failure occurs when the screw tip continues moving within the motion band without reaching the hole's attractor basin, preventing successful insertion. The failure is triggered when the elapsed time surpasses a predefined threshold.

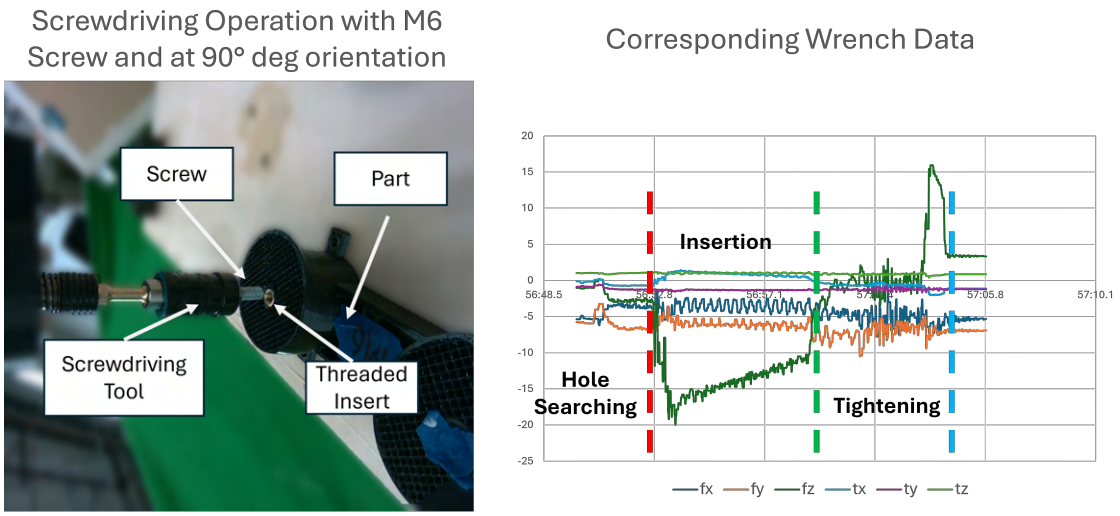
The defect detection system operates in parallel with the execution process, facilitating real-time intervention by activating corrective controllers or alerting human operators when automated recovery is not feasible. This framework enhances the reliability and adaptability of robotic screwdriving operations, minimizing downtime and ensuring robust failure mitigation strategies.

In prior work [4], a deep learning-based multi-modal failure detection framework was explored. While such methods can be highly effective, they are often sensitive to environmental variations (e.g., lighting conditions, component colors) and require large amounts of data, which can be difficult to collect in high-mix, low-volume (HMLV) manufacturing settings. Furthermore, vision-based deep learning methods can introduce deployment challenges, particularly in dynamic industrial environments. To address these limitations, we need a more scalable and generalizable approach that relies exclusively on force signatures. Therefore, we did not pursue deep learning-based methods in this work.

In [199], the authors demonstrated the robustness of a decision tree-based methodology compared to other approaches, such as Long Short-Term Memory (LSTM) networks, Support Vector Machines (SVMs), and Logistic Regression (LR) for screwdriving applications. Their results highlight the strength of decision trees in failure detection for screwdriving, making them an attractive choice for industrial applications. Our observations and experiments align with



(a) Part A with M4 Screw and 30 Deg Part Orientation



(b) Part B with M6 screw and 90 Deg Part Orientation i.e, part laid flat horizontally

Figure 8.8: Variation in Wrench Signals Across Different Screw Types and Orientations. This figure illustrates how wrench signals vary when performing screwdriving operations on different parts, screw types, and orientations. Note that these wrench signals were recorded for a successful screwdriving insertion for the fairness of comparison. Notably, while the f_z signal exhibits similar overall characteristics, we observe shifts in sign, gradient, and distinct variations in other force and torque signals during both the insertion and tightening phases. These variations highlight the challenges of directly applying prior methods and underscore the need for a more adaptable approach—as proposed in our work-to ensure robustness in high-mix, low-volume (HMLV) manufacturing.

these findings, reinforcing the efficacy of decision trees in structured failure mode classification based on sensor patterns. Consequently, we also adopt a decision tree-based classifier, leveraging its advantages in terms of robustness, interpretability, and ease of deployment, which are critical factors in real-world industrial automation where adaptability and reliability are critical. Of the five failure modes mentioned, the decision tree is used to detect the first four, while the dynamics model provides the necessary information to trigger the fifth mode, as previously discussed.

The approach presented in [199] explored an instance of a problem with the following attributes: (1) a single screw type, (2) screwdriving is performed in a gravity-assisted orientation, and (3) Compliance is in the tool (Passive), with no active compliance strategies in the robotic system. Our work generalizes the previous work. Our experiments (Refer Fig. 8.8) indicate that even subtle differences from variations in screw geometries, torque requirements, and insertion orientations can introduce significant variabilities in sensor readings. These variations make it challenging to directly use the results reported in [199]. Moreover, a fundamental constraint in HMLV settings is the inability to collect large failure datasets due to the diverse and evolving nature of tasks. To overcome this, we propose a novel data augmentation strategy that systematically enhances the collected data, ensuring that the decision tree is trained with appropriate inductive biases. This augmentation enables our methodology to generalize across different screwdriving conditions, making it significantly more deployable in HMLV automation.

8.6.1 Data Augmentation and Pre-processing

To enable real-time defect detection, we temporally segment sensor signals, creating a structured state representation with a fixed time interval, ΔT,

which can be adjusted based on operational requirements. We assign state labels within each interval using self-supervised signal analysis, ensuring a well-defined mapping between sensor readings and failure modes.

For model training, we initially process wrench data collected from the robot's joint torque sensors during experiments. Each data segment is labeled according to its corresponding failure mode. However, given the limited availability of experimental data, augmentation is necessary to enhance model robustness against variations and sensor noise. Additionally, raw wrench signals often exhibit inherent noise and fluctuations within a given sampling window, which can affect classification accuracy. To address these challenges, we apply a series of pre-processing operations to extract informative features while making the model resilient to noise. Specifically, for each sampling window, we perform the following:

- Gaussian Noise Injection: We introduce noise sampled from $\mathcal{N}(\mu, \sigma)$ to simulate sensor variations.
- Low-Pass Filtering: The noisy signal is passed through a low-pass filter to retain essential low-frequency components while reducing high-frequency fluctuations.
- Periodic Disturbance Modeling: We incorporate sinusoidal perturbations to account for periodic disturbances commonly observed in real-world settings.
- Time Shifting and Amplitude Scaling: Time shifting is applied to handle phase variations, while amplitude scaling ensures robustness against intensity fluctuations.

By leveraging these augmentation techniques, we generate high-quality synthetic variations of real-world data, enabling the decision tree-based classifier to generalize effectively across different operating conditions. This ensures a

reliable and scalable defect detection system suitable for industrial deployment.

8.6.2 Feature Extraction

Following the data augmentation process described in Section 8.6.1, we implement a multi-stage feature extraction methodology to handle the non-stationary characteristics of wrench signals. The proposed pipeline computes nine temporal and statistical features through sliding window analysis with window size t.

Feature Definitions For a signal window $\mathbf{X} = \{x_1, x_2, ..., x_N\}$ with $N = t \cdot f_s$ samples (f_s = sampling frequency): (1) Mean:$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$, (2) Std. dev.: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$, (3) RMS: $x_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$, (4) Skewness: $\gamma_1 = \frac{\mathbb{E}[(X-\mu)^3]}{\sigma^3}$, (5) Kurtosis: $\gamma_2 = \frac{\mathbb{E}[(X-\mu)^4]}{\sigma^4} - 3$, (6) Zero Cross Rate (ZCR): $\frac{1}{N-1}\sum_{i=1}^{N-1} \mathbb{I}(x_i x_{i+1} < 0)$, (7) Energy: $E = \sum_{i=1}^{N} |x_i|^2$, (8) Entropy: $H(X) = -\sum_{k=1}^{K} p_k \log_2 p_k$, (9) Correlation: $\rho_{xy} = \frac{\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)}{\sigma_x \sigma_y}$

Each of these features was selected through rigorous analysis of the wrench data and validated using Principal Component Analysis (PCA). The model's performance benefits significantly from this feature set because they capture complementary aspects of the wrench signal characteristics: temporal statistics (μ, σ, $x_{\max}$, $x_{\min}$) quantify the signal's amplitude variations, higher-order moments (γ_1, γ_2) detect subtle changes in signal distribution, while information-theoretic measures ($H(X)$, ZCR) effectively identify transient anomalies and mode transitions. The pairwise correlation coefficients further enhance failure mode discrimination by capturing the coupling between different wrench components. Through PCA, we verified that these features exhibit minimal redundancy while maintaining high explanatory power. The window size t was optimized through cross-validation on the dataset, with $t = 150$ ms providing

an optimal trade-off between stationarity assumptions and detection latency requirements.

8.6.3 Decision Tree-based Defect Detection

The features described in Section. 8.6.2 provides a state signature for detecting the defects. The focus in this work is real-world deployment and scalability. Considering the HMLV nature of the task and challenges pertaining to real-world data, we choose the decision tree-based modeling framework that is well-suited for this task due to its ability to handle non-linear decision boundaries, its robustness to small dataset sizes and its interpretability, which allows for insight into the key factors contributing to each failure mode.

The classifier takes as input the set of features f_{input} computed over a fixed sampling window ΔT, which encapsulates relevant statistical and temporal characteristics of the wrench signals and robot state. Given these feature representations, the decision tree learns a hierarchical set of rules that partition the feature space into distinct failure mode classes. This structured decision-making approach enables the classifier to generalize effectively to unseen data while maintaining computational efficiency for real-time deployment.

To train the decision tree, we use labeled data collected from experimental trials where each instance corresponds to a specific failure mode. The training process involves feature selection, as outlined in the previous section. Subsequently, the tree is constructed by iteratively splitting the feature space based on decision rules that maximize information gain using the Gini impurity criteria. In order to prevent overfitting and enhance generalization, we apply pre-pruning (depth constraints) and post-pruning techniques. The decision tree model is integrated into the robotic system to perform failure mode classification in real-time. During execution, the system continuously monitors

the wrench signals and robot state, computing features over each sampling window ΔT. The trained decision tree then classifies the failure mode and triggers the appropriate response. The system invokes predefined contingency controllers if the detected failure mode is recoverable. However, if failure recovery is infeasible, an alert is generated for operator intervention.

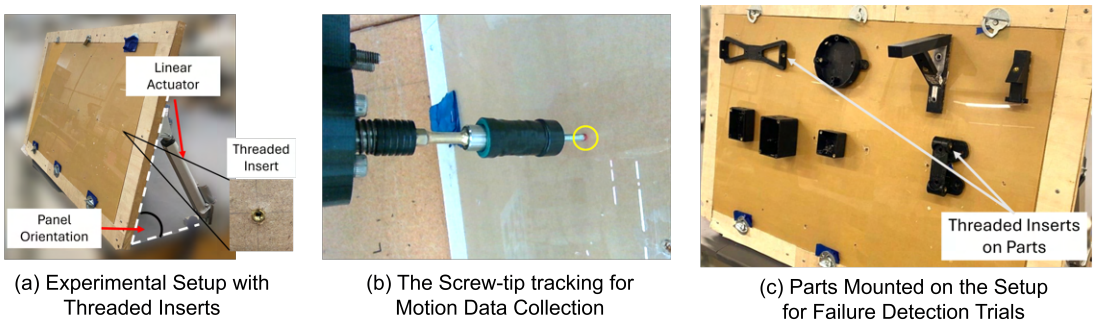
8.7 Experiments and Results

The experiments aim to evaluate the effectiveness of the proposed system and validate the following key hypotheses:

- **Real-World Suitability and Reliability**: Does our system perform reliably across a diverse set of parts and screw types in real-world applications?
- **Generalization of the Physics-Informed Dynamics Model**: Can our model accurately predict screw-tip motion across different scenarios?
- **Prediction of Completion Time**: Can the dynamics model effectively estimate the time required for task completion?
- **Defect Detection Reliability**: Does our failure detection method accurately classify and identify different failure modes?

8.7.1 Experimental Setup and Test Parts

Experimental Setup: As discussed in Section 8.1, the primary objective of our system is to operate in high-mix, low-volume (HMLV) manufacturing settings. In such environments, fastening operations must accommodate varied part orientations and multiple screw types, requiring a flexible and reliable screwdriving system. To evaluate our system's capability under these conditions, we design an experimental setup as illustrated in Fig. 8.9. The



(a) Experimental Setup with Threaded Inserts

(b) The Screw-tip tracking for Motion Data Collection

(c) Parts Mounted on the Setup for Failure Detection Trials

Figure 8.9: The experimental setup serves as a testbed for data collection for our dynamics model, failure detection model, and for performing screw-driving trials. (a) Depict the panel at an orientation that can be adjusted by two actuators on each end, (b) Depicts the in-hand camera's RGB images used to detect the colored screw-tip, and (c) Shows our trials for failure-mode detection, with all the parts mounted on the panel at an orientation

setup consists of an actuated mounting panel, where test parts can be securely attached. The panel can be precisely oriented using two linear actuators, enabling us to simulate real-world scenarios where screwdriving must be performed in non-gravity-assisted configurations. By controlling these actuators, we can systematically vary the panel's orientation to assess the system's robustness across different inclinations.

Beyond serving as a testbed for screwdriving experiments, the proposed setup also facilitates data collection for modeling screw-tip dynamics. By varying part orientations, we gather diverse motion trajectories, enabling us to build and validate our physics-informed dynamics model. This setup ensures a controlled yet adaptable environment for evaluating our system's reliability, scalability, and generalization across different fastening conditions.

Test Parts and Screw Types: To evaluate the robustness of our system across different fastening scenarios, we conduct experiments using three screw types, each varying in width and thread size: (1) M4 $\times$ 0.7 mm, (2) M5 $\times$ 0.8 mm and (3)M6 $\times$ 1.0 mm. All selected screws are pan head with a 20 mm length. The choice of these screws is based on their compatibility with our

Figure 8.10: Left: The three Screw Types (M4, M5, and M6) used for the trials and Right: The ten test parts selected for performing screwdriving trials. Each of them has a different geometrical complexity, size, and shape, and they are representative of different industrial settings (E.g., Electrical Components, Automobile Parts, Refrigerator Parts, etc.). Furthermore each of them have different screw types and at different orientations to simulate a realistic HMLV scenario

screwdriving tool and their prevalence in HMLV manufacturing environments [219].

For part selection, we focus on representative industrial components that reflect the diversity of products typically encountered in HMLV settings. The test parts, illustrated in Fig. 8.10, exhibit variations in: (1) Geometrical complexity - influencing the system's ability to reach and perform screwdriving operations. (2) Size categories - classified into small, medium, and large to test scalability. (3) Threaded insert locations - affecting alignment and insertion precision. All parts used in our experiments are scaled 3D-printed replicas of real-world industrial components, allowing us to systematically test different fastening conditions while maintaining controlled experimental variables. This selection ensures a comprehensive evaluation of our system's ability to handle diverse part geometries, sizes, and insertion challenges.

8.7.2 Dynamics Model Evaluation

Data Collection: The dynamics model proposed in Section 8.5.1 characterizes screw-tip motion as a function of compliance parameters. Therefore, our goal during data collection is to obtain timestamped screw-tip motion data in a fixed reference frame for different screw types and impedance control parameters, defined as: (1) Stiffness: $\mathcal{K} = [k_x, k_y, k_z, k_\alpha, k_\beta, k_\gamma]$ and (2) Damping: $\mathcal{C} = [c_x, c_y, c_z, c_\alpha, c_\beta, c_\gamma]$. Additionally, to ensure that our model generalizes well to changes in part orientation, we collect motion data at three different part orientations of $[30^o, 60^o, 90^o]$.

Tracking the high-velocity motion of the screw tip during data collection is a key challenge. To address this, we use a color-based marker on the screw-tip, which is tracked using an in-hand RealSense D405 camera. This camera is well-suited for close proximity (0.01-0.02 m) tracking with < 1 mm precision. The

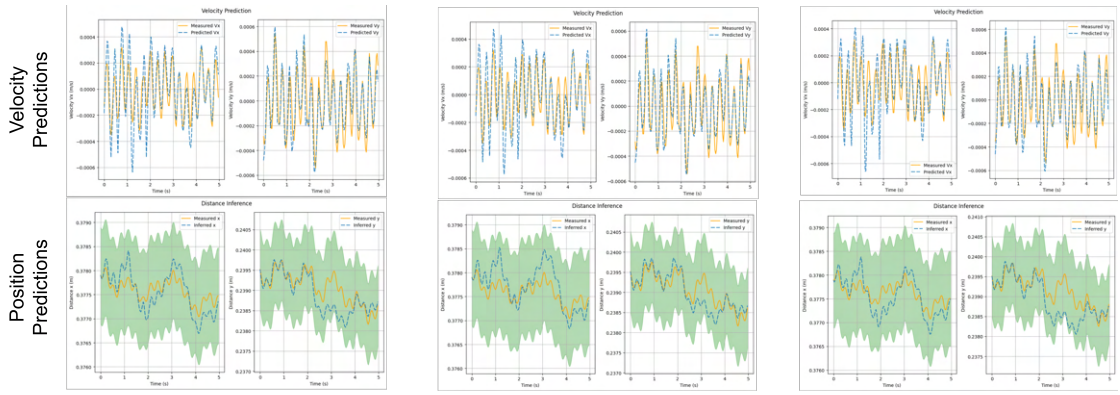


Figure 8.11: The Predicted vs Ground truth for screw-tip motion. Here the predictions are for all three screw types used in our study. The model predicts velocity $\dot{X}$. However, we also compute the cartesian trajectory of the screw tip given an initial state. We can see all our predictions are within the green zone, which signifies a 1 mm deviation for the screw tip from the reference trajectory. Also, if we observe closely, our predicted characteristics follow the ground truth characteristics of the screw-tip motion. This underscores the robustness of SINDy in modeling such highly nonlinear dynamics.

camera provides time-synchronized RGB and depth frames at 30 FPS, allowing us to reconstruct the screw-tip trajectory with high accuracy. This data is then post-processed to detect the screw-tip with a color-based filtering technique (Refer Fig. 8.9). Specifically, we employ a simple HSV (hue-saturation-value) based filter in OpenCV [220] to get the colored marker's mask. Once we have the mask, we approximate the centroid of this mask, which gives us the position of the tip in the image frame. Since the depth frame is aligned, we acquire the Z value of the screw-tip. We then deproject the identified pixels into Cartesian space using the camera's intrinsic parameters, allowing us to accurately determine the (x, y) position of the screw-tip in the camera frame. This streamlined data collection process enables efficient large-scale data acquisition across all three screw types. It is important to note that the color marking on the screw-tip is only required during the data collection phase for building the dynamics model. During execution, no visual markers

are necessary since the learned dynamics model can reliably predict the screw-tip position in real time.

Our initial experiments revealed that fine-grained variations in impedance parameters $\mathcal{K}$ and $\mathcal{C}$ had minimal impact on screw-tip motion. To simplify the analysis while preserving essential dynamics, we discretized impedance values into three categories: High ($\mathcal{K}_{trans} = [2500Nm], \mathcal{K}_{rot} = [250Nm/rad]$), Medium ($\mathcal{K}_{trans} = [1500Nm], \mathcal{K}_{rot} = [150Nm/rad]$), and Low ($\mathcal{K}_{trans} = [500Nm], \mathcal{K}_{rot} = [50Nm/rad]$). For each screw type and part orientation, we conducted approximately 50 trials, with each trial lasting at least 20 seconds. Given our system's 30 FPS * recording rate, each trial generated around 600 frames (30 FPS $\times$ 20 sec). In total, we collected 30,000 samples, which were used for model training and evaluation. The dataset was split 80:10:10 into training, validation, and testing sets. We employ appropriate subset sampling to ensure that data points from different screw types and orientations are uniformly represented across each of the dataset splits.

Model Performance: We train our model for 100 epochs using the loss function defined in Eq. 8.2. Training is conducted on an NVIDIA GeForce RTX 3060 GPU with an Intel Core i7 processor and 32GB of memory. We employ the Adam optimizer with a batch size of 32 to ensure stable convergence. The SINDy framework is implemented in PyTorch (code available at: `https://github.com/l0g1c-80m8/screwdriving-model-sindy-rcim`). Fig. 8.11 presents our model's performance in predicting screw-tip velocity, evaluated on a held-out test set of 3000 samples. The predicted velocities are integrated to obtain the screw-tip position, which closely follows the ground truth reference trajectory. Notably, our model maintains a prediction error within 1 mm of the reference position, as highlighted by the green region in

*Note: 30 fps is due to depth data

Fig. 8.11, demonstrating its accuracy and reliability in capturing screw-tip dynamics.

Model's Dominant Terms: SINDy enforces model sparsity through sequential thresholding, where terms with coefficients below a predefined threshold (10^{-4} in our case) are set to zero during training. This process ensures that only a few dominant terms remain by the end of training, making the model more interpretable and computationally efficient. In our case, the most significant terms were third-order polynomial terms, followed by sinusoidal components, second-order polynomial terms, and a few constant terms. This enforced sparsity enhances interpretability and improves computational efficiency, making our model well-suited for real-time applications where high-frequency predictions are critical.

Baseline Comparison: To highlight the suitability of the SINDy model in capturing the nonlinear dynamics of the screw tip, we benchmark its performance against fully data-driven end-to-end models, such as Multi-Layer Perceptrons (MLPs) and Long Short-Term Memory Networks (LSTMs) that have been a popular choice for modeling system dynamics.

Model	Mean MSE (m)	Max MSE (m)	Std. Dev.
Ours	**0.00035**	**0.0009**	**0.00025**
LSTM	0.0653	0.1305	0.038
MLP	0.07	0.13	0.04

Table 8.1: Comparison of SINDy model with LSTM and MLP for predicting screw-tip dynamics. These numbers are reported on a rollout of the model for a time horizon of 5 seconds on our held-out testing trajectories. What we observe is that even though the loss for LSTMs and NNs is low, the divergence is significant, leading to poor predictions beyond a couple of timesteps.

As shown by the error values in Table 8.1, SINDy significantly outperforms MLP and LSTMs in predicting the screw-tip's position. These findings are consistent with previous results reported by the authors of the SINDy method

[204]. Additionally, we observe that for both MLP and LSTM, the loss drops sharply early in the training cycle, suggesting that these models quickly converge to suboptimal local minima. While their overall loss values appear low, they fail to generate accurate predictions beyond a few timesteps, indicating poor long-term generalization. This highlights the superior accuracy of SINDy in modeling nonlinear screw-tip dynamics and also its ability to provide interpretability, as it explicitly identifies the fundamental governing equations of the system.

8.7.3 Predicting Time to Completion with Dynamics Model

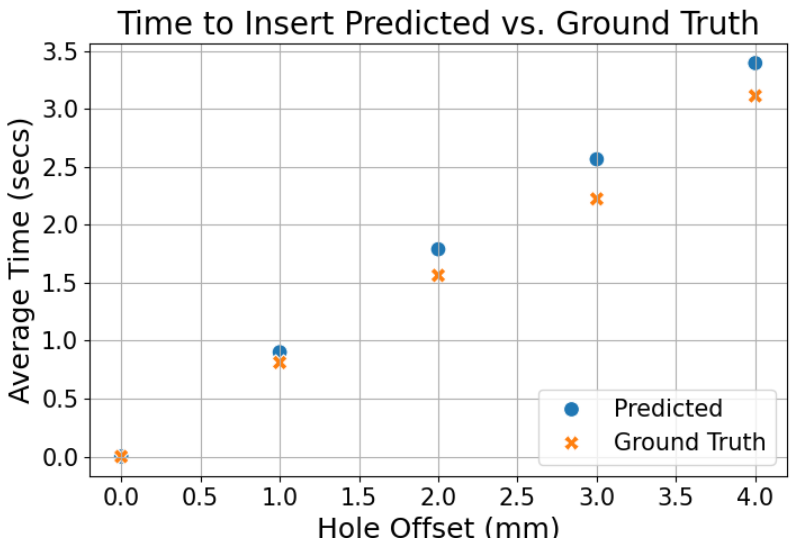


Figure 8.12: The Predicted vs Ground truth time to insertion for a given offset of the screw-tip. This data is collected during our experiments when we keep the screwdriving tool rpm at a fixed value of 600 rpm

The screw-tip dynamics model we developed provides insights into the level of uncertainty a designed system can tolerate. For example, given a set of screw types, we can estimate the maximum hole pose offset a system can handle. Additionally, this model allows us to approximate the time required for the screw tip to reach the hole's attractor basin. Fig. 8.12 illustrates

these computations by reporting the average time for the screw tip to fall into the hole's attractor basin for various hole pose offsets. The mean insertion time is computed across all trials for each offset. To determine this time from experimental data, we analyze the wrench data, specifically monitoring the f_z force component and the z value of the robot's end-effector. Once initial contact is made, an increase in f_z beyond a predefined threshold serves as an indicator of successful engagement between the screw and the hole. Only successful trials are considered in this analysis.

Using our dynamics model, we further predict the insertion time by randomly sampling hole poses within an offset range $\eta \in [0, 4]$ mm and computing the average time it takes for the predicted screw-tip trajectory to pass through the hole's center. For a given screw tip diameter, we define successful insertion when the tip overlaps with at least 90% of the hole's area. This threshold ensures that the screw tip is sufficiently aligned with the hole's attractor basin, allowing it to fall in and engage reliably. As shown in Figure 8.12, our predictions align well with the ground truth insertion times. Notably, our model exhibits a conservative bias, consistently overestimating the insertion time. This insertion time acts as a benchmark for defining T_{insert}, the threshold beyond which a reattempt is initiated to maintain efficient operation. The average time required to complete a screwdriving operation once the screw tip makes contact with the near plate i.e., the part surface is 5 seconds across all our trials.

8.7.4 Failure Detection Results

The first four failure modes out of the five discussed in Section 8.6 exhibit distinct state signatures based on wrench data. We collect data using the same testbed described in Section 8.7.1 to train our decision tree model for failure

detection. However, generating failure cases presents a significant challenge, as some failure modes-such as screw misalignment-occur infrequently under nominal operating conditions. This scarcity of failure data makes the data collection process inherently complex. To address this challenge, we leverage the data augmentation method outlined in Section 8.6.1, enabling us to enrich our dataset with realistic failure cases.

To systematically capture failure cases, we conduct 45 trials by intentionally introducing positional offsets from the hole and varying the approach orientation during screwdriving. This controlled experimental design ensures that our training dataset encompasses a diverse range of failure modes. In these 45 trials, we also include successful screwdriving trials to help the failure detection model learn normal operating conditions and distinguish failures from nominal behavior. Importantly, all training data is collected exclusively on the flat panel with threaded inserts described in Section 8.7.1.

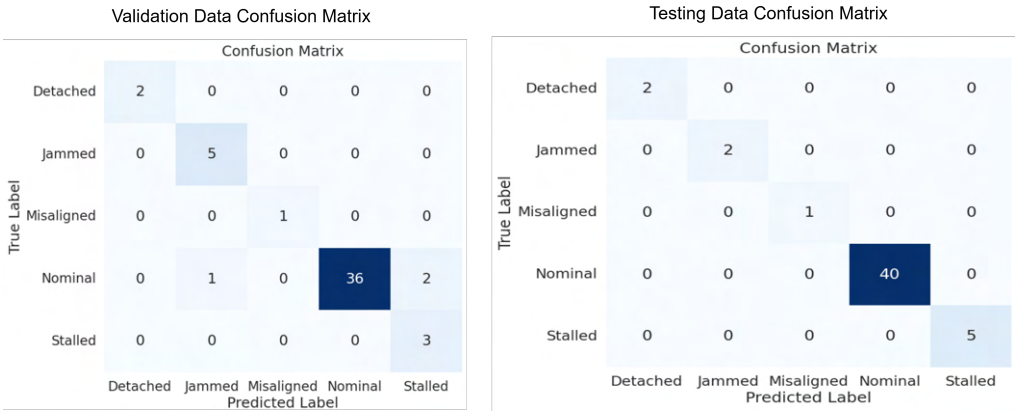


Figure 8.13: Left: The confusion matrix for our validation dataset collected on a flat panel with an F1-score of 0.94. Right: We perform classification on testing data collected on our 10 different parts, where our model accurately classifies all modes of failure as shown in the figure

To evaluate the robustness and generalization of our decision tree model, we conducted 50 screw-driving trials using 10 different 3D-printed parts, each equipped with appropriate threaded inserts. It is crucial to highlight that while

the training data is collected on the flat panel with threaded inserts (Refer Fig. 8.9), all testing data is collected exclusively on these 3D-printed parts, allowing us to assess the model's ability to generalize to unseen geometries and assembly conditions. The distribution of failure modes across the 45 training trials (On flat Panel) and 50 testing trials (On 10 Parts) is summarized in Table 8.2.

Table 8.2: Distribution of Trials for Training and Testing. Training data is collected on the flat panel with threaded inserts while testing is performed by executing a screwdriving operation on 10 real-world parts.

Category	**Training Trials**	**Testing Trials**
Successful Completion (Nominal)	25	40
Operation-Stalled	5	4
Misaligned-Screw-in-Tool-Holder	5	1
Screw-Detached-Without-Successful-Completion	5	3
Thread-Jamming	5	2
Total	45	50

Table 8.3: Test Accuracy and Classification Report. Classes 0,1,2,3,4 are the same as in the order they appear in the columns of Fig. 8.13

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	2
1	1.00	1.00	1.00	2
2	1.00	1.00	1.00	1
3	1.00	1.00	1.00	40
4	1.00	1.00	1.00	5
Accuracy			1.00	50
Macro Avg	1.00	1.00	1.00	50
Weighted Avg	1.00	1.00	1.00	50

We train our decision tree model using data from 45 training trials, along with augmented data generated through the methodology outlined in Section 8.6. Each data point undergoes five augmentation steps, resulting in an additional 225 training samples to enhance model robustness. We implement the decision tree using scikit-learn [221], employing Gini impurity as the splitting criterion. The dataset is split 80:20 for training and validation, with all augmented data used exclusively for training.

The confusion matrix in Fig. 8.13 demonstrates that our model achieves perfect recall (1.0) for failure mode detection, ensuring that all defects are correctly classified. In validation, we observe an F1-score of 0.94, with nominal operation modes being the primary source of misclassification. These misclassifications typically occur in cases where the screw was near a failure mode but ultimately succeeded due to the robot's active compliance compensating for the offset. For qualitative insights, we provide videos illustrating such scenarios at: https://sites.google.com/usc.edu/physicsinformedscrewdriving.

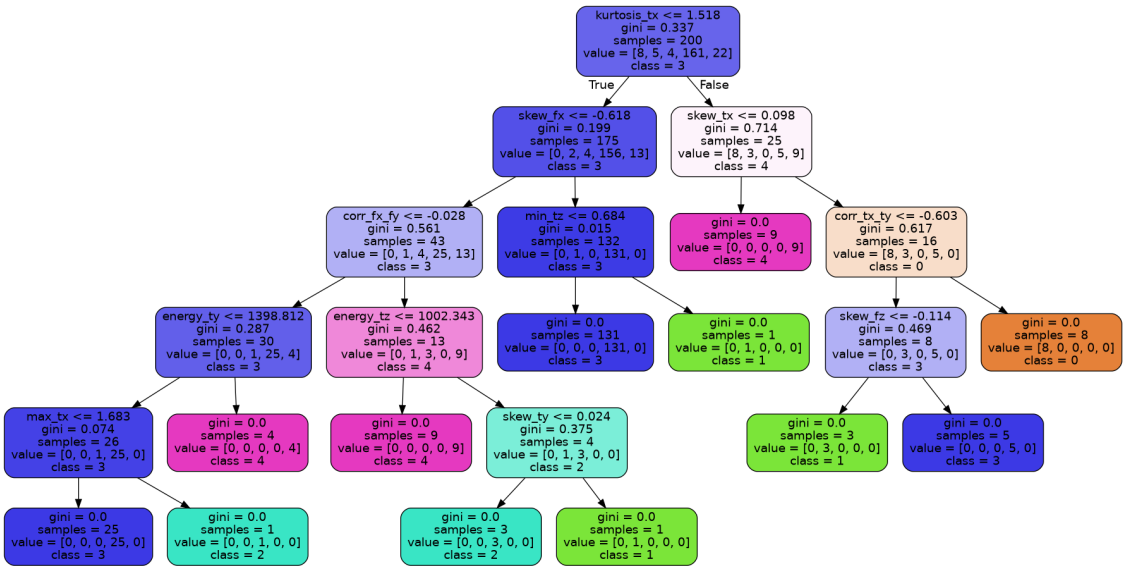


Figure 8.14: The learned decision tree for the five classes (modes) of operation. This figure is auto-generated using sci-kit learn graphviz functionality and depicts how the decision tree is performing the splits.

Our model's robustness is evident in the testing results, where it successfully identified all failure cases, including nominal operation modes, without any misclassifications (Refer Table 8.3). These trials were conducted across the 10 distinct parts shown in Fig. 8.9, simulating a realistic HMLV setting. This scenario is representative of an HMLV setting, underscoring our failure detection method's applicability to such cases. The corresponding learned decision tree model is visualized in Fig. 8.14.

Effect of Data Augmentation: To evaluate the impact of our augmentation strategy, we conduct an ablation study where no data augmentation is applied. Without augmentation, the validation F1-score drops significantly to 0.70, with a corresponding recall of 0.70, indicating a reduced ability to classify failure cases correctly. On the test set, the F1-score declines to 0.94, with recall decreasing to 0.92. These results underscore the importance of augmentation, demonstrating that it enhances the model's robustness by improving its ability to distinguish failure modes more effectively.

8.8 Summary

This chapter investigated how compliance—both passive in the tool and active in robot control—can be harnessed to achieve robust screwdriving under real-world uncertainties, particularly in high-mix, low-volume (HMLV) settings where expensive fixturing is infeasible. The key contributions and findings are as follows:

- Autonomous compliant screwdriving system: A mobile robotic cell was developed that combines a passively compliant rotary tool with Cartesian impedance control, along with vision and force sensing, to perform autonomous screwdriving on varied parts without relying on rigid fixtures.
- Physics-informed dynamics modeling via SINDy: Sparse Identification of Nonlinear Dynamics (SINDy) was used to learn an interpretable, data-efficient model of the screw-tip behavior as a function of compliance parameters and robot orientation. This model outperformed end-to-end neural network approaches and enabled accurate predictions of insertion dynamics across a range of hole pose offsets.
- Data-driven process planning: By leveraging the SINDy model to estimate time-to-insertion, the system can automatically trigger reattempts

or escalate for intervention when a screw fails to insert within an informed time threshold—thereby closing the loop between prediction and action.

- Robust decision-tree-based failure detection: A decision-tree framework was introduced to classify five distinct failure modes—four force-based anomalies plus a time-elapsed failure, achieving perfect precision and recall on test data. A targeted data-augmentation strategy further boosted detection performance.
- Extensive real-world validation: Trials on ten industrial parts and three screw sizes (M4, M5, M6) demonstrated 100% success under hole pose uncertainties up to ± 4 mm/$\pm 3^o$, with average insertion times around five seconds. These results confirm the system's adaptability and reliability in practical HMLV environments.

Together, these results underscore the critical role of compliance modeling and physics-informed learning in enabling reliable robotic manipulation under uncertainty. By integrating dynamic simulation, interpretable modeling, and real-time feedback, the framework advances the state of the art in autonomous screwdriving and offers a blueprint for other precision tasks involving deformable or compliant elements. This chapter's insights on compliance, physics-informed modeling, and failure resilience feed directly into the overarching dissertation goal of building intelligent, explainable, and robust robotic systems for deformable/compliant object manipulation.

Chapter 9

Bi-manual Manipulation for Shell-like Deformable Objects

9.1 Introduction

Deformable object manipulation has traditionally focused on 1D structures like ropes and cables, or 2D structures like cloths and sheets, as discussed in previous chapters. While these settings have offered essential insights into planning and control for deformable materials, the evolving landscape of logistics, e-commerce, and small-batch manufacturing is introducing new classes of deformable objects that demand a different perspective. In particular, shell-like deformable packages—such as polybags containing internal objects—are becoming increasingly prevalent. These packages present unique challenges: unlike simple sheets or ropes, they consist of two thin surfaces enclosing a freely moving internal mass, producing complex, coupled deformation behaviors. Depending on the properties of the external material and the contained object, these packages can exhibit unpredictable bending, sagging, and folding patterns during handling. Despite their growing industrial importance, such shell-like deformable objects remain understudied in the robotics community, creating a critical gap between current research and real-world application needs.

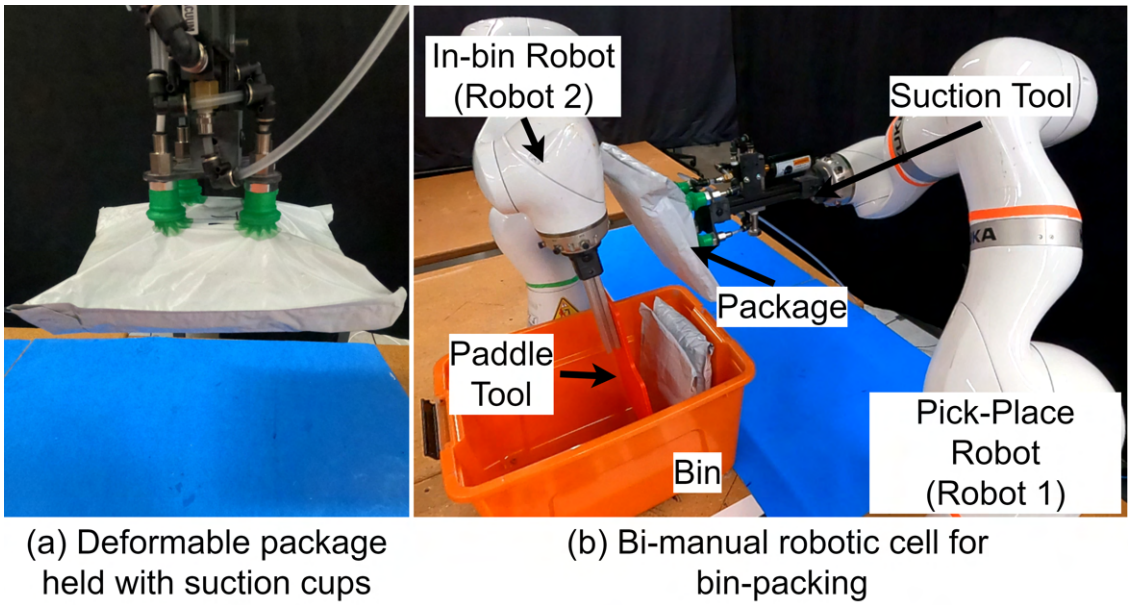


Figure 9.1: (a)The deformable packages studied in this work. (b)The proposed bimanual cell. The in-bin robot stays inside the bin during the packing process, while the pick-place robot transports and places packages inside the bin.

One domain where these challenges manifest acutely is bin-packing in warehouse logistics. Efficient bin-packing is essential for optimizing space, minimizing costs, and meeting the growing demands of rapid fulfillment. While rigid packages have been extensively studied with conventional packing strategies [222], deformable packages introduce a different class of problems: (1) Their deformation depends heavily on internal contents, making shape prediction unreliable, (2) Visual inspection alone cannot fully infer their physical properties, and (3) Traditional packing methods, which assume rigid or uniformly stackable items [223], often fail when applied to such deformable items.

Fig. 9.1 illustrates the class of deformable packages addressed in this chapter. Unlike rigid boxes, these packages demand adaptive handling strategies that can stabilize and manipulate their pliable structure during packing. In preliminary investigations, we observed that humans naturally employ bi-manual strategies when packing such deformable packages: one hand stabilizes the

partially packed contents (providing local structure and control), while the other places new items (exploiting open space and compressibility). Bimanual manipulation becomes particularly important when dealing with deformable objects for two reasons: (1) Stabilization and active control are needed simultaneously to prevent undesirable deformations during placement, and (2) Coordinated force application across two points allows for gentle correction of object shapes without causing damage.

However, enabling bimanual robotic systems introduces significant new challenges. Coordinating two arms demands precise synchronization, particularly under conditions of uncertainty where object behavior may be unpredictable. Beyond synchronization, reasoning about the coupled effects of grasping and sweeping actions becomes complex, as internal and external deformations of the object are deeply interdependent. Effective planning must therefore navigate a delicate balance—optimizing task objectives, such as maximizing packing density, while simultaneously respecting physical constraints, such as preventing package rupture or minimizing excessive force application.

Drawing inspiration from these human strategies, this chapter presents a bimanual robotic system that mirrors this dual-role principle. One manipulator is responsible for stabilizing and adjusting package positions within the bin, while the second manipulator performs suction-based pick-and-place operations for incoming packages. Unlike rigid bin-packing tasks, where simple heuristics are often sufficient, packing deformable objects demands a more adaptive and responsive approach. Heuristic-driven strategies frequently fail in these settings, especially when faced with the dynamic, unpredictable deformations that arise during real-world handling (see Section 9.7.3).

To tackle this complexity, we develop a learning-based action prediction framework that jointly reasons about both arms' actions to maximize a bin-packing

efficiency score. The problem is further complicated by the fact that bin-packing must operate in an online fashion, where packages arrive sequentially without prior knowledge of their characteristics [224]. Moreover, purely simulation-trained models often suffer from sim-to-real transfer gaps [225, 226], limiting their effectiveness when deployed in physical systems. Extensive real-world data collection is also impractical due to the vast diversity of package types encountered in operational warehouses.

To overcome these challenges, this work adopts a hybrid learning strategy that leverages the strengths of both simulation and real-world experimentation. A simulation environment built in MuJoCo is first used to pre-train the model on a broad range of deformable package interactions, imparting strong physical priors into the learning process. The model is then fine-tuned using a small but carefully curated set of real-world trials to adjust for sim-to-real discrepancies. Finally, an optimizer-in-the-loop is integrated into the framework to predict action parameters that directly maximize packing efficiency without requiring perfect simulation fidelity. This combined approach allows the system to generalize across a wide range of packages while maintaining robust real-world performance.

The key contributions of this chapter are summarized as follows:

- Development of a bimanual robotic system capable of securely and efficiently packing deformable packages, validated with real-world experiments across 18 distinct objects.
- A parameterized action prediction framework that learns to predict bin-packing efficiency, achieving a mean squared error (MSE) of 0.003 across 100 real-world trials.
- A physics-informed simulation environment that models shell-like deformable object interactions in MuJoCo, enabling scalable pre-training.

By extending the study of deformable object manipulation beyond classical sheets and ropes to more complex, deformable packages, this work bridges an important gap between academic research and the practical challenges faced in modern warehouse automation. Through careful integration of bimanual control, learning, and simulation, we take an essential step toward deploying intelligent, resilient robotic systems in real-world logistics environments.

9.2 Related Work

Bimanual object manipulation: Bimanual robotic cells have been extensively studied for handling various objects, including rigid, deformable, articulated, etc [227, 228]. Due to their inherent ability to mimic human manipulation skills, such robotic cells are preferred for complex manipulation tasks. However, they present motion and task planning challenges due to their increased complexity [229]. In [230], the authors study the bimanual manipulation of garments, where the objective is to fold clothes. Bimanual manipulation has also been applied in medical robotics, where intricate dextrous manipulation capabilities are essential [231]. Prior research often employs a primitive-based approach, predicting parameters for specific movement primitives [232], but none have addressed deformable packages.

Bin-Packing: Bin-packing in robotics has been a long-standing problem [233]. Traditionally, bin-packing refers to the problem of maximizing the number of objects packed into a bin [234]. Our focus lies in manipulation planning within this context, specifically addressing the online bin-packing problem of object manipulation and placement [222]. Past research has predominantly examined bin packing for homogeneous, rigid, regular-shaped objects [235–237]. However, the challenge of packing deformable objects remains largely unexplored [238, 239].

Deformable object manipulation: Recently, deformable object manipulation has gained significant interest [227]. In [240], the authors propose a system for manipulating deformable packages. However, the focus is on efficiently picking packages from a pile rather than stowing them in a bin. In [226], the authors model object interactions and demonstrate the robotic stowing task with a single robot arm. However, they focus on rigid objects easily perceived by an image or point cloud rather than objects hidden inside packages. Most of the other work mainly focuses on manipulating objects such as sheets [3], garments, elastic cables [228], etc. In [241, 242], the authors study the bagging task but focus on manipulating the deformable bag for packing. Our work differs in its emphasis by modeling the impact of robot actions on bin-packing efficiency rather than inter-object interaction.

9.3 Bimanual Robot Setup for Packaging

Fig. 9.2 illustrates the deformable package bin-packing task facilitated by the proposed bimanual setup. This process involves picking a package, ensuring its safe transportation, and appropriately positioning it within the bin. The overarching goal is to optimize packing efficiency and process time. A crucial consideration for achieving high efficiency in this task lies in maintaining the stability of packages within the bin; this entails ensuring that in-bin packages remain upright relative to the bin's base and that new packages are safely deposited at their intended locations. Packages need to be upright and snugly packed due to the downstream requirements of the bin that would typically traverse across a large fulfillment center on a mobile platform or conveyor. Other packing strategies can lead to a potential risk of the package being dislodged from the bin during transportation.

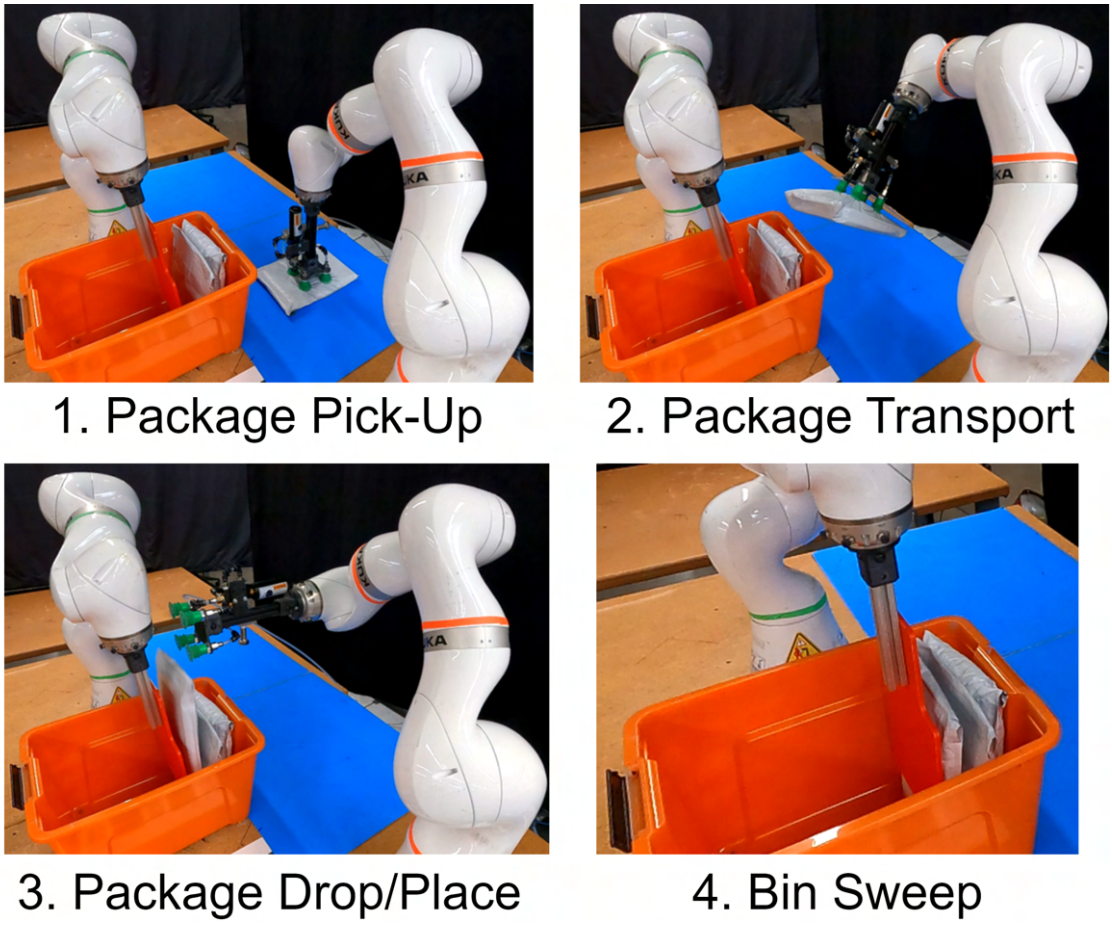


Figure 9.2: Overview of the entire pick-and-place pipeline with the bimanual robotic cell.

Our setup (Refer Fig. 9.2) comprises two KUKA LBR IIWA robots equipped with joint torque sensors capable of operating under impedance control mode. The in-bin robot operates with perpetual compliance, while the pick-place robot exhibits compliance only during picking. The compliance of the pick-place robot is enabled to compute package characteristics online (Refer Section. 9.6). The in-bin robot is retrofitted with a rigid paddle-shaped tool to manipulate the in-bin packages, while the pick-place robot has a suction-based gripper for safe picking and transporting packages. In the context of bin-packing, both robots perform certain action primitives that enable the task objective. A RealSense D415 camera provides RGB-D image observations of the bin to extract a comprehensive bin-state representation.

9.4 Problem Formulation

The objective of the bin packing problem is to optimize packing efficiency. Thus, given an observation O of the bin state S at a given instance, the goal is to compute robot actions a^1 and a^2 for the pick and place robot (Robot 1) and the in-bin robot (Robot 2), respectively. We assume that packages are handled one at a time in the order of arrival, following an online approach for bin packing.

Let $a_i = \langle a_1^i, a_2^i \rangle \in \mathcal{A}$ be the action the bimanual system executes at state S for packing the i^{th} package, where $i \in [1, N]$ and N is the total number of packages. We also define a scoring function $\eta(\cdot, \cdot)$ that computes the packing efficiency of a bin, given the observation of the bin state and characteristics of the package that is queued to be placed in the bin. Thus, the bin packing problem can be represented as solving the following program:

$$\max_{a_i^*} \ \eta(O, p_i) \tag{9.1}$$

Here, a_i^* is the optimal action that maximizes the packing score $\eta(\cdot,\cdot)$ and p_i are the characteristics of the i^{th} package that is going to be placed in the bin. The actions here are robot end-effector poses for both the robots with respect to a base frame of reference. Assuming that the bin dimensions and position are available, the actions for the robots can be precisely defined as $a_1^i = (x_1, \beta_1)$, and $a_2^i = (x_2, z_2, \beta_2)$. Here, x_1 & x_2 are the x movement of the corresponding robots, z_2 is the z movement of the robot-2, β_1 & β_2 are y euler angles w.r.t a nominal frame $T_{package} \in \mathbb{SE}(4)$. We fix the other degrees of freedom due to their redundancy in influencing $\eta(\cdot,\cdot)$ (Refer Section 9.6.1 for details). Additionally, we presume access to a time-optimal trajectory planner with collision-checking capabilities to facilitate the minimization of process time.

9.5 Approach

9.5.1 Packaging Pipeline as a Finite State Machine (FSM)

Our bimanual bin-packing pipeline can be represented as an FSM to facilitate control and task planning (Refer Fig. 9.3). Upon the arrival of a new package, the system transitions from pick pose detection to successful placement in the bin. The perception module provides the system with image and point cloud observations to compute nominal pick and drop locations. However, relying solely on perception module data can lead to failures due to calibration issues and the deformable nature of the packages. Hence, developing a strategy for learning the optimal actions for both robots involved in the bin-packing process is crucial. We outline the key states governing task completion as follows.

Pick State: The system estimates the pick pose of the package using the RGB-D image from the overhead camera. We generate the segmentation

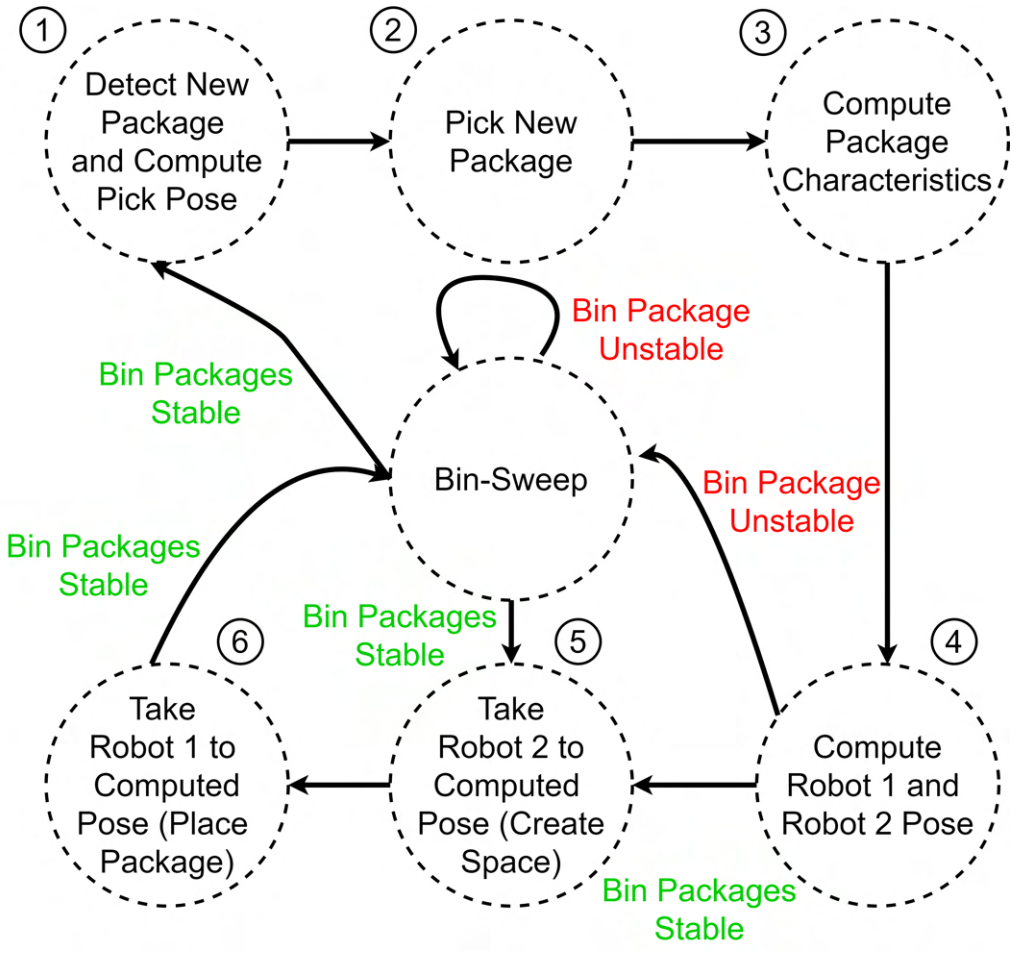


Figure 9.3: The pick and place pipeline for bin-packing is represented as a simplified FSM due to its sequential nature. Such representation guides the system's high-level actions.

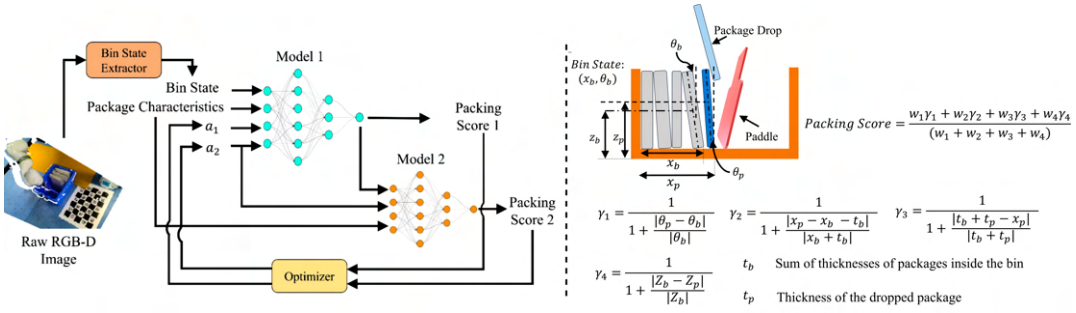


Figure 9.4: Left: Our entire model and real-time optimization pipeline to compute optimal actions that maximize the packing score. Packing score prediction functions are modeled with a Multi-Layer Perceptron (MLP). Right: The packing score representation and bin state definition. Definitions remain the same for computing both packing score 1 and packing score 2

mask of the package using YOLOv8 on the RGB image. Subsequently, an oriented bounding box is generated for the observed point cloud. Since packages are placed on the table, only yaw rotation is considered for the pick pose. The trajectory planner computes a feasible trajectory to the pre-pick pose. The pick-place robot moves linearly in the z-direction, interacting with the package under compliance control mode while recording external forces on its end-effector. The stiffness (K) and thickness of the package are determined using the recorded F_z on the robot's flange and the z-deviation (ΔZ) from its commanded goal pose.

Transport State: After picking up the package, the robot receives a drop location from the system. Subsequently, the transport robot follows a time-optimal trajectory to safely deliver the package, considering maximum velocity and acceleration constraints to prevent mid-trajectory drops. Force-torque data is recorded during this motion to determine the package's mass. This enables the characterization of package attributes $P_t = (m, l, b, h, K)$, where m is the mass in kgs, l, b, h are dimensions in meters, and K is stiffness in N/m.

Bin-preparation State: The in-bin robot ensures package stability and upright positioning within the bin. Once the bin is stable, the system provides the P_t and current state observation for the in-bin robot to compute the optimal action a_1. We observe that the height at which the paddle is placed while sweeping plays a crucial role for a successful sweep.

Package Drop State: Once both the robots are at their intended locations, the suction pressure is halted, allowing the package to drop into the available space. Subsequently, the in-bin robot performs a sweep motion under compliance control mode. During sweeping, the paddle aligns with the package while maintaining the z height computed in a_2. Sweeping is repeated until the packages are stable. This process iterates until the bin reaches full capacity.

9.5.2 Learning Packing Score Function

Defining a metric to quantify bin-packing quality can be approached in several ways. Our proposed bin-packing action prediction framework maintains flexibility in accommodating various metric definitions. However, for our specific downstream application, we required a metric tailored to our unique setup. We recommend that practitioners redefine the packing score metric to align with the specifications of their individual cells. Given our bimanual setup and the deformable nature of packages, we had to design a packing score that captures the influence of both robots on packing efficiency. Thus, we define packing score as depicted in Fig. 9.4. The γ_1 and γ_2 components promote bin stability by encouraging package configurations aligned with the last package in the bin. The γ_3 and γ_4 components ensure that the packing fraction of the bin is maximized and the configurations that disrupt the current bin state are penalized accordingly. Moreover, we define packing scores using the same elements for the pick-place robot and the in-bin robot. Throughout training

data collection, these scores are computed using the point cloud observations (Refer Section 9.6.1), and our objective is to predict these scores for a given action set at a nominal bin state.

One potential approach involves predicting these scores for given actions analytically. However, due to the deformable nature of packages, predicting package dynamics using finite element methods or analytical approaches poses significant challenges [227]. Thus, we aim to learn a model to predict the defined packing score given a new package and the corresponding robot actions for a given bin configuration. This model serves as the function approximator for $\eta(.,.)$. Our empirical investigations revealed that the final packing score of the bin for a package placement is influenced by the sequence of actions corresponding to the drop of the package (robot-1 actions) and the sweeping of the package (robot-2 actions). Suboptimal package placements during the suction robot's package-dropping phase hinder the in-bin robot's ability to achieve a high packing score.

Consequently, the model architecture for predicting the packing score must account for this sequential dependence. As illustrated in Fig. 9.4, our proposed architecture addresses this challenge by simultaneously optimizing the packing scores of both robots. Specifically, we feed the packing score of the bin after the transport robot's actions η_1 into the network responsible for predicting the packing score after the in-bin robot's actions η_2, ensuring coordinated optimization.

The model comprises two multi-layer perceptron networks, MLP 1 and MLP 2 (Refer Fig. 9.4), predicting the corresponding packing score. The input to MLP 1 is the robot actions a_1 and a_2, as well as the bin-state defined by the last package's orientation with respect to the bin's vertical face and the distance of the package's base from the bin's origin (Refer Fig. 9.4). The

bin state is computed using point-cloud observation of the bin (Refer Section 9.6.1 for details). We also tried using PointNet [243] architecture as a state encoder. However, our minimalist state representation gave us satisfactory performance (Refer Section 9.7). Moreover, this approach aids in reducing the overall model complexity, thereby improving the parsing time and memory footprint necessary for real-time execution. The MLP 2 takes the output of MLP 1, package characteristics, and the robot-2 actions a_2. This design choice is motivated by the observation that η_2 is conditioned only on η_1 and robot-2 actions a_2. Here, η_1 serves as a surrogate for the bin-state post package drop is complete. The model is trained by optimizing the following loss function with L2-regularization:

$$\mathcal{L}_{packing}(w) = \sum_{i=1}^{2} \lambda_i \mathcal{L}^i_{packing} + \lambda_3 ||w||_2^2 \tag{9.2}$$

Where $\mathcal{L}^i_{packing}$ (Refer Eq. 9.3) is the Huber loss that motivates reducing mean as well as median error and is suitable for the packing score's data distribution,

$$\mathcal{L}^i_{packing} = \begin{cases} \frac{1}{2}(y^i - \hat{y}^i)^2 & if \left|(y^i - \hat{y}^i)\right| < \delta^i \\ \delta^i * ((y^i - \hat{y}^i) - \frac{1}{2}\delta^i) & \text{otherwise} \end{cases} \tag{9.3}$$

Here, $\mathcal{L}^1_{packing}$ and $\mathcal{L}^2_{packing}$ are the loss values for both robots' packing scores, $\lambda_1, \lambda_2, \lambda_3$ are regularization hyperparameters, while δ^1 & δ^2 are thresholds at which change gets triggered between L1 & L2 loss.

9.5.3 Learning Optimal Robot Actions

The model described in Section. 9.5.2 predicts the corresponding packing scores for the actions executed by robot 1 and robot 2, given the bin and package state. Our objective is to solve the inverse problem, i.e., to compute

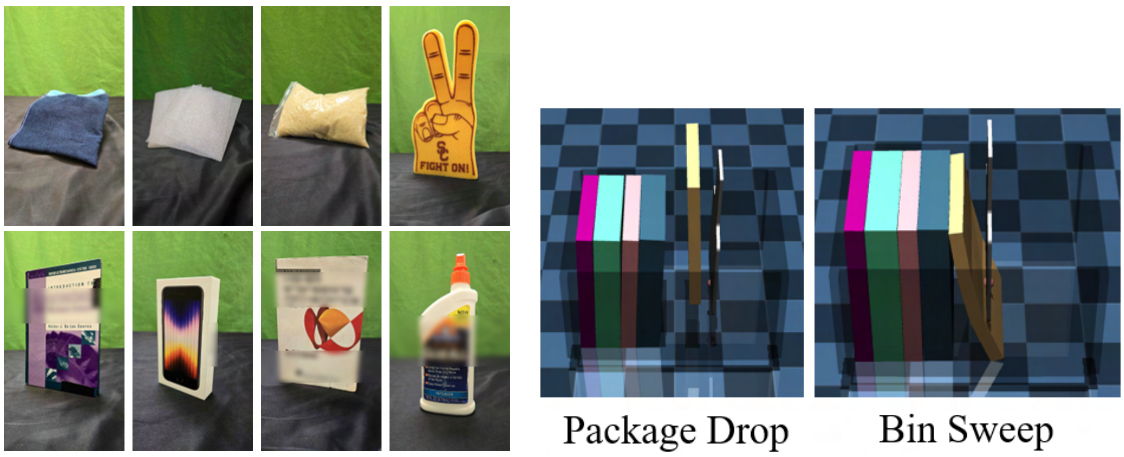
the actions that can maximize both the packing scores. In order to achieve this, we devise an optimization loop in conjunction with the MLP model (Refer Fig. 9.4). The objective of this optimizer is to maximize the weighted mean of score 1 and score 2.

For such an optimization scheme to be viable in real-time execution, a crucial design consideration is the convergence time. Furthermore, the loss landscape for our score prediction model (Refer Eq. 9.3) can be saddled with several local minima, leading to suboptimal action computations. Thus, we opt for the parallel basin-hopping method, a standard global optimization routine widely used for such problems [244]. This method reinstates numerous local optimizers with initial conditions in the vicinity of the current best local minima. The local optimizers use a gradient-based optimization scheme based on the L-BFGS-B algorithm, which supports bounded-constrained optimization. The bounds are placed on the domain of the action space. Thus, in this manner, we overcome the problem of being stranded in suboptimal local minima, and parallel basin-hopping ensures convergence can be achieved in the minimum possible time for it to be suitable for real-time deployment.

9.6 Experiments

9.6.1 Real-World Experiments

We collect the data necessary to train the packing score prediction model. 18 different types of objects (6 Rigid, 12 Deformable) are packed inside an LPDE polyethylene-based padded package (Refer Fig. 9.5a). We aim to collect data for packing scores for a given bin state and a package. Thus, we initially used 12 packages (9 Deformable + 3 Rigid) for training data collection. We uniformly generate a sequence of packages of different thicknesses



(a) Sample objects packed inside the package in real data collection

(b) Simulation data generation in MuJoCo

Figure 9.5: Simulated and real data generation for model training. We replicate the setup in MuJoCo with deformable packages

and execute the bin-packing pipeline described in Section. 9.5.1. During execution, we uniformly sample the values for actions a_1 ($x_1 : U(-0.02, 0.02)$ m, $\beta_1 : U(0.0, 20.0)$ degrees) and a_2 ($x_2 : U(0.0, 0.03)$ m, $z_2 : U(-0.07, 0.05)$ m $\beta_2 : U(0.0, 20.0)$ degrees) from the bin state. Additionally, we collect the raw point cloud of the bin to compute the bin state. The cropped point cloud data comprises the container, paddle, and package stacks. The force data collected while package pickup is used to compute the package stiffness and thickness using the impedance control parameters *. In this manner, we generate about 1000 bin-packing scenarios and compute corresponding packing scores. The data collection is executed in a self-supervised manner, thus obviating the need for expensive human labeling.

Generating Bin State Data and Ground Truth Packing Scores: To compute the bin state, we collect the raw point cloud of the bin. We then employ a density-based spatial clustering algorithm (DBSCAN) [245] and plane patch detection [246] to extract the position and orientation of the package

*Refer to our project website for implementation details: https://sites.google.com/usc.edu/bimanual-binpacking

stack. Once the suction robot drops a new package, to segregate this new package from the previous package stack and container, we perform a difference operation between the old and newly captured point cloud and subsequently perform DBSCAN and plane patch detection for computing the pose of the new package. The same process is repeated after the bin-sweep operation is completed. Then, using these point clouds (refer to Fig. 9.6), we compute the packing scores as per definitions in Fig. 9.4.

Dataset	Size (Real Data)	MSE↓				MAE ↓				Max Error↓			
		Trained on Sim + Real		Trained on Real Only		Trained on Sim + Real		Trained on Real Only		Trained Sim + Real		Trained on Real Only	
		Score 1	Score 2	Score 1	Score 2	Score 1	Score 2	Score 1	Score 2	Score 1	Score 2	Score 1	Score 2
Training	695	**0.0025**	**0.0026**	0.0042	0.005	**0.012**	**0.062**	0.036	0.085	**0.040**	**0.070**	0.102	0.110
Validation	100	**0.0032**	**0.0035**	0.0067	0.0081	**0.063**	**0.062**	0.096	0.098	**0.161**	**0.182**	0.325	0.356
Test	100	**0.0033**	**0.0034**	0.0075	0.0081	**0.065**	**0.076**	0.098	0.101	**0.165**	**0.191**	0.372	0.395

Table 9.1: The packing score prediction model's performance, evaluated with 5-fold cross-validation, shows that the model pre-trained on simulation data outperforms others across all metrics, reducing the maximum error by 46-48%

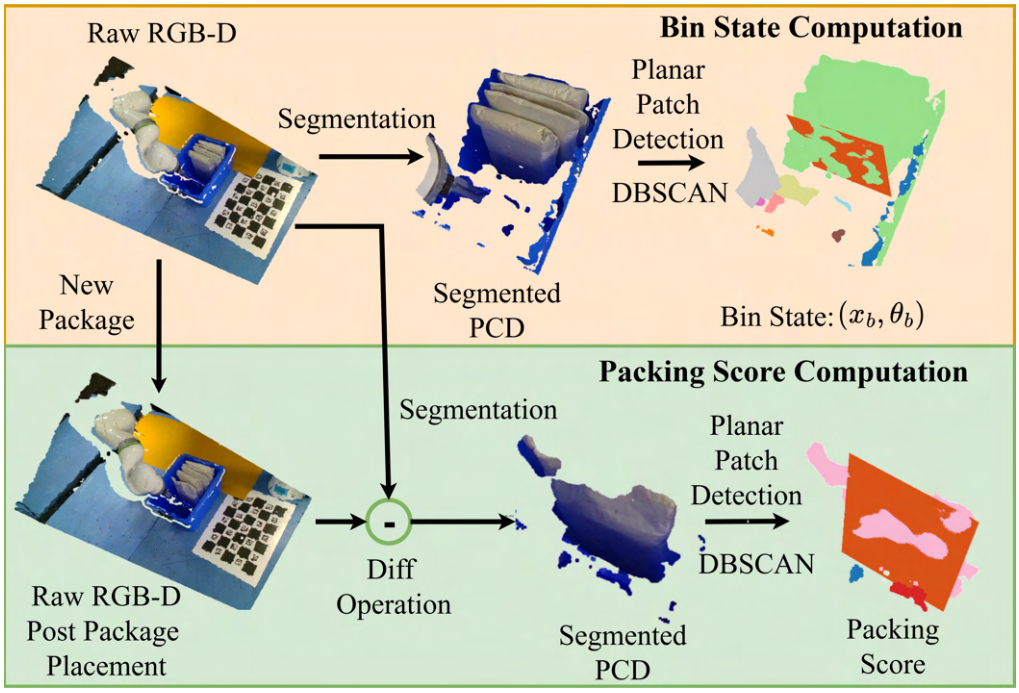


Figure 9.6: The raw point cloud and the corresponding processed point clouds used for bin state and packing score computation. Process for computing packing score 1 & 2 is the same

9.6.2 Simulation Experiments

In simulation, we replicate the data collection strategy adopted in the previous section. We use MuJoCo 3.0 [247] and model each package as a flex object. We generate instances of packages with varying thicknesses ($U(0.005, 0.07)m$) and stiffnesses ($U(10, 8000)$ N/m) to ensure that the dataset represents the type of objects that can be encountered during real-world execution. In the simulation, the key is to generate diverse data; thus, we sample different bin states ranging from empty to almost full bins. The paddle-shaped tool and the bin are modeled as rigid objects. We generate 20000 simulation scenarios to pre-train the packing score prediction model (refer to Fig. 9.5a).

9.7 Results

9.7.1 Failure State Estimation and Packing Score Predictions on Real Data

Failure State Estimation: Prior to training a model to predict a packing score for a given state and action, we train a state estimation model that classifies a given state action pair into two categories : (1) feasible and (2) infeasible (Refer Fig. 9.7). Infeasible states are those with a packing score below 0.4. Our observations indicate that packing scores below 0.4 typically result in the system's inability to recover to a high packing score after the bin sweep, a trend consistent across both simulation and real-world data. Therefore, we focus on predicting packing scores only for feasible states. We identified 1138 infeasible cases out of 20,000 in simulation data and 105 infeasible cases out of 1000 in real-world data. We label these cases as infeasible. Consequently, we train the state estimation model with simulation and real datasets. The

training dataset comprises feasible states (Real: 695 + Sim: 18,862) and infeasible states (Real: 55 + Sim: 1138), with the model trained using binary cross-entropy loss. We evaluated the model's performance on test data consisting of 200 feasible and 50 infeasible real-world cases. The model performs classification with an F1 score of 0.99 (Precision: 0.995 and Recall: 0.987).

Packing Score Prediction: Our packing score prediction model results are illustrated in Table. 9.1. We train the model on a combination of real and simulated data. Initially, we pre-train the model with 18862 simulation data points. This model is then fine-tuned on a batch of 695 real-world data points. Model 1 and Model 2 (Refer Fig. 9.4) consist of 3 fully connected layers with ReLU activations. We learn the model parameters using an Adam-W optimizer with a fixed learning rate of 0.002 and batch size of 30. To demonstrate the impact of simulation, we present the results by training two model instances: (1) A model pre-trained on simulation data and then fine-tuned on real data and (2) A model only trained on 695 real data points. Table. 9.1 depicts that simulation data significantly boosts model performance for all metrics.

Our mean squared error (MSE) values of 0.0033 and 0.0034 for packing scores 1 and 2 suggest a strong fit to the test data. These low MSE values indicate that our model's predictions closely align with the observed packing scores. Cases with error > 0.15 occur infrequently, specifically in only 4 cases out of 100. Notably, these instances typically occur at the fringe, with packing scores between 0.4 and 0.5. All the models are trained on a system configured with an Intel i7 4.9 GHz and equipped with NVIDIA GeForce RTX 3060.

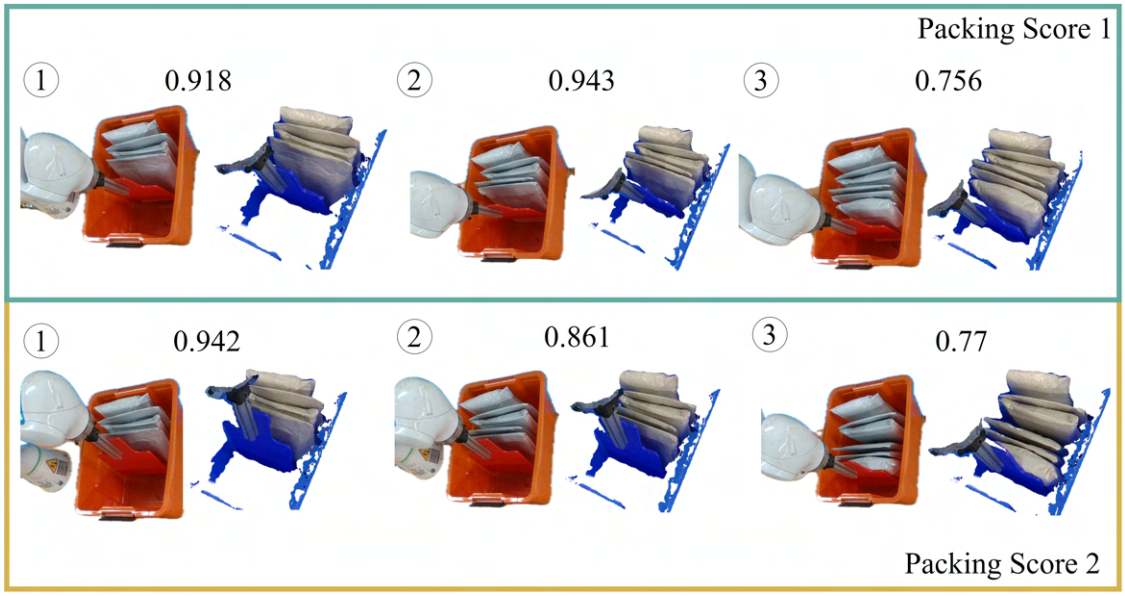


Figure 9.7: Intermediate dropping and sweeping performances of the system achieving high-quality bins. We also demonstrate the bin packing instances when scores were lower than 0.8.

No. of Trials	MSE ↓		MAE ↓		Max Error ↓		Latency (ms)↓
	Score 1	Score 2	Score 1	Score 2	Score 1	Score 2	
20	0.0019	0.0021	0.033	0.038	0.108	0.107	226

Table 9.2: The performance of the action prediction framework in predicting bin packing score during online execution. The mean packing score during these trials was 0.88 for score 1 and 0.91 for score 2

9.7.2 Action Prediction Performance

To assess the effectiveness of the optimizer in predicting optimal actions, we conducted real-world trials involving 20 distinct bin states not previously encountered in our data collection. These trials utilized the remaining six packages from our dataset, ensuring a comprehensive evaluation. Subsequently, we executed the actions recommended by the action prediction framework for these novel bin states and recorded the resulting packing scores.

Table 9.2 presents a comparative analysis between the observed packing scores and those predicted by our framework. Our findings illustrate the optimizer's ability to effectively predict actions that maximize packing scores. Moreover, the calculated Mean Squared Error (MSE), Mean Absolute Error (MAE), and maximum error values between the predicted and observed packing scores underscore the robustness of our system. The max errors of 0.108 and 0.107 occurred in scenarios where the bin approached full capacity. The average bin packing scores during these trials were 0.88 for score 1 and 0.91 for score 2, reaffirming the efficacy of our system in consistently producing well-packed bins.

Additionally, our framework exhibits computation efficiency, with an average processing time of 226 milliseconds for computing actions for a given state. This time encompasses bin state computation, model parsing, and optimizer convergence, indicating the viability of our model for real-time execution and its applicability in practical scenarios.

9.7.3 Bin-packing Performance:

To demonstrate the need to learn optimal actions for our bimanual setup, we benchmarked our method against two approaches: (1) Random and (2) Heuristic-based. In random trials, we uniformly sample values for actions

No of Trials	Type of Trial	Avg. Score 1 ↑	Avg. Score 2 ↑	Success Rate ↑
20	ours	**0.88**	**0.91**	**20/20**
	heuristic	0.76	0.82	16/20
	random	0.62	0.66	13/20

Table 9.3: Our approach outperforms random and heuristic-based approaches with a high final packing score of 0.91

similar to Section. 9.6.1 and record the bin-packing score and whether the package placement was successful or not. For the heuristic approach, we use package thickness and stiffness as parameters to decide the action values and assume that package deformation plays no role in packing. As shown in Table 9.3.

9.7.4 Sensitivity Analysis:

Number of Samples	Perturbation in Actions	Mean Change in Packing Score	Max Change in Packing Score
100	10 %	1.5 %	2.6%
	20 %	2.3 %	4.1 %
	30 %	4.6 %	10.2 %
	40 %	7.4 %	17.6 %
	50 %	18.2 %	24.5 %
	>50 %	25.3 %	34.7%

Table 9.4: Effect of adding noise in the actions computed by the optimizer. Here, a 50% change in action values corresponds to 2 cms in position values and 10^o in orientation values, respectively.

The optimizer plays a pivotal role in predicting actions that maximize packing scores. Its effectiveness hinges on whether the loss landscape for action vs. packing score at a given state warrants such optimization. Indeed, if a random selection of actions consistently yields high packing scores, the necessity for an optimizer diminishes. To test this hypothesis, we conduct a sensitivity analysis of the optimizer.

In this analysis, we initially compute the optimal actions that maximize the packing score for the bin states in our test data. Subsequently, we systematically perturbed the action space by random increments and observed the corresponding effects on packing scores. Table 9.4 presents our findings, revealing an exponential decrease in packing scores when optimal actions computed by the optimizer were randomly perturbed. Notably, instances where action values were perturbed by 50% resulted in an average packing score decrease of 18.2%. These results underscore the indispensability of optimization in our problem domain. They suggest that arbitrary action selections are unlikely to yield high packing scores consistently, reaffirming the critical role of the optimizer in maximizing packing efficiency.

9.7.5 Simulation Results:

To demonstrate the fidelity of our simulation, we recreate the scenarios from our real-world dataset and conduct a qualitative and quantitative evaluation of the resulting discrepancies. Fig. 9.8 illustrates a few such cases for the bin state and action pairs that either lead to failure conditions that are crucial to recognize if we want to rely on the simulation data to provide us a robust enough representation of reality and a high-quality inductive bias for model training. Our simulation effectively replicates failure cases, including scenarios where (1) dropped packages disrupt in-bin arrangements, (2) the dropped package lands on top of other packages by colliding with the paddle tool, and (3) the bin sweeps by the in-bin robot that led to instability in the bin. These cases are the primary reasons for cases with low packing scores, and the ability of our simulation to replicate these scenarios underscores its proficiency in generating high-quality data.

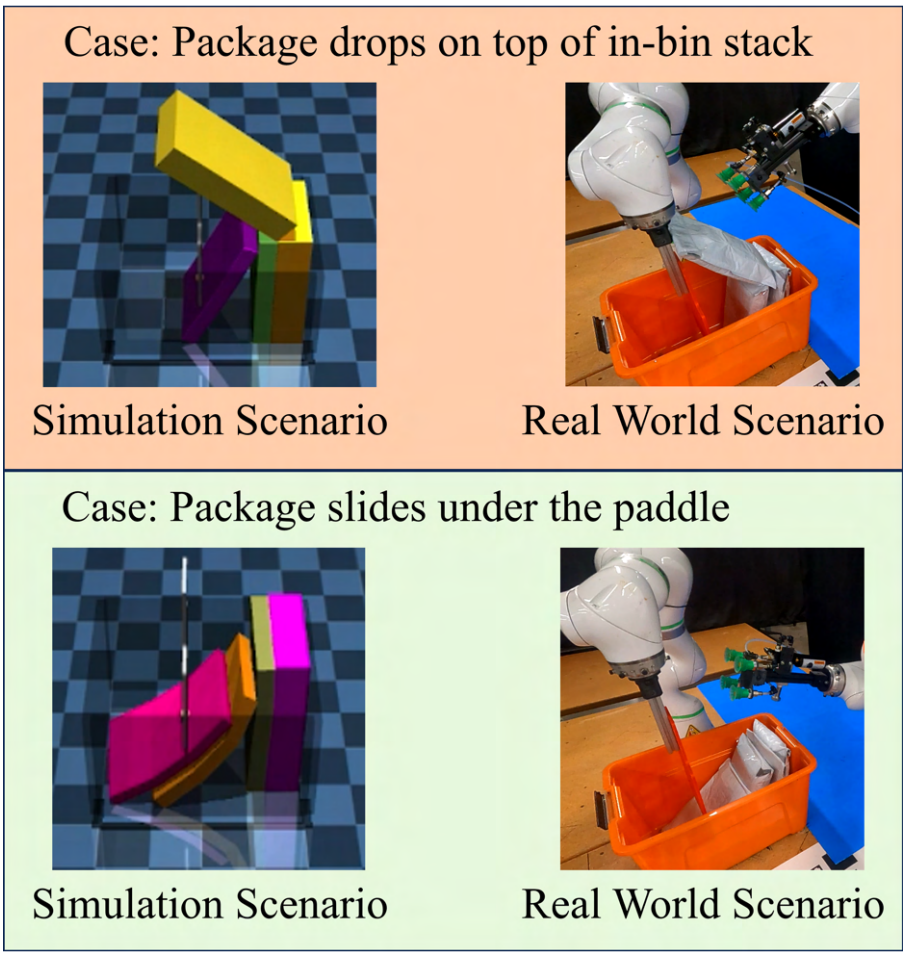


Figure 9.8: The comparison between the real world and the corresponding simulation scenario, depicting the effectiveness of simulation to capture the essence of package characteristics

Additionally, we simulate the real-world scenarios in our test dataset. We compute the corresponding values for both the packing scores and compare them with the true values. The low MSE values in the simulated vs. true packing scores error (Refer Table. 9.5) demonstrate that simulation produces data that can provide a good inductive bias for pre-training the model. The max errors were 0.33 for Score 1 and 0.32 for Score 2, observed in cases with lower packing scores (<0.5). However, in these cases, the values of packing scores in the simulation were lower than the actual observed ones, thus making the sim data reflective of failures.

No. of Datapoints	**MSE ↓**		**MAE ↓**	
	Score 1	**Score 2**	**Score 1**	**Score 2**
1000	0.014	0.013	0.096	0.09

Table 9.5: Performance of Sim vs Real in computing packing scores by recreating scenarios encountered in test data. The errors >0.2 occur in only 0.5% cases. However, all the failure cases are captured robustly with a low error.

9.8 Summary

This chapter introduced a compositional learning framework for addressing the emerging challenges of manipulating shell-like deformable packages in bin-packing operations, a problem increasingly relevant in robotic warehousing and fulfillment industries. By developing a bimanual robotic cell and an action prediction methodology that jointly reasons about coordinated grasping and sweeping actions, we demonstrated how robots can achieve optimized packing efficiency even when dealing with complex, flexible packages.

A key contribution of this work lies in the integration of simulated and real-world data within a self-supervised learning framework, enabling robust policy

training without requiring extensive manual labeling. Furthermore, the proposed framework incorporates physics-based priors in the form of the stiffness of the packages being handled. The developed simulation pipeline accurately captures the interaction dynamics between deformable shells within the bin, facilitating effective transfer to physical deployments. Our experimental results highlight the system's ability to generalize across a variety of packages and demonstrate strong real-world performance.

Overall, this work advances the frontier of robotic manipulation for complex deformable objects, providing a foundation for scalable, reliable, and efficient automation of shell-like package handling tasks.

Chapter 10

Anomaly and Failure Detection for Deformable Objects

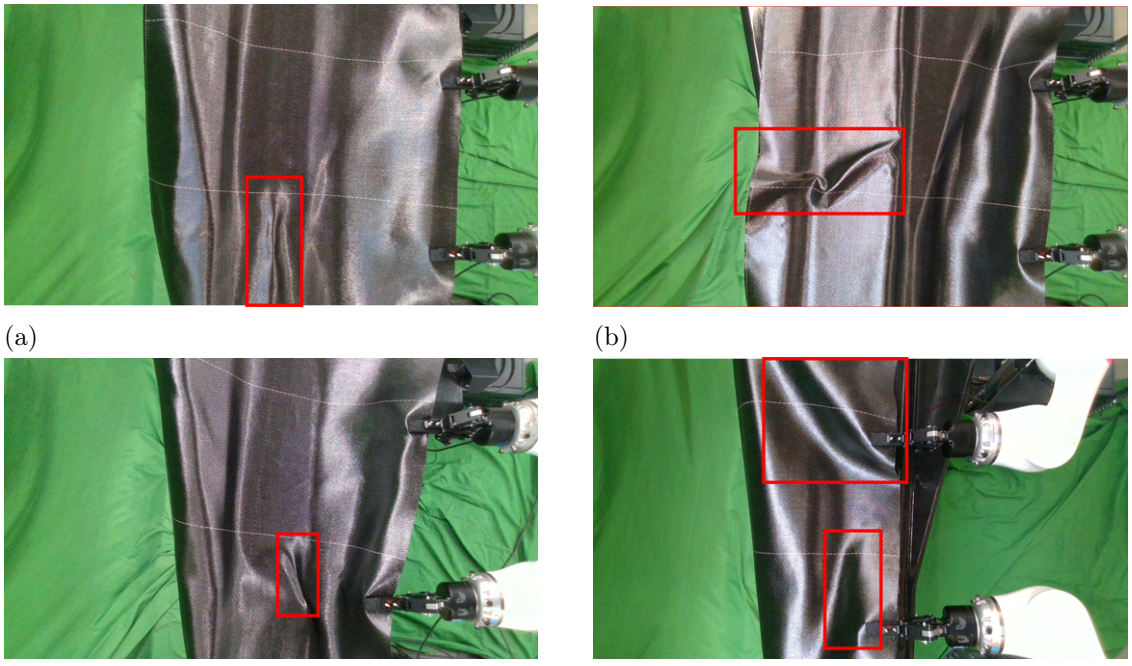
10.1 Introduction

In robotic manipulation tasks, failures are inevitable. Most of the learned policies we discussed in previous chapters are susceptible to failures due to modeling errors (sim-to-real gap), errors in controllers, perception errors, changes in the environment (lack of strong inductive bias), etc. In previous chapters, we studied how to detect and overcome perception errors, highlighting the potential of modern learning frameworks. Deep learning-based models can be powerful in detecting failure modes. Nonetheless, the reliance of these models on extensive datasets necessitates meticulous data collection and annotation procedures. Reliable failure detection is a cornerstone of any robust robotic manipulation system, but it is especially challenging when the manipulated objects themselves are deformable. Unlike rigid parts, where defects can often be characterized by simple geometric deviations, deformable materials exhibit a wide variety of irregular, context-dependent failure modes. In sheet-handling applications, such as composite layup or flexible packaging, defects like wrinkles and folds can appear in countless shapes, sizes, and orientations, making it difficult to define clear, universal criteria for what constitutes a "failure."

Detecting these defects early, before they propagate into more serious errors or lead to scrapped parts, is critical in high-performance industrial processes [248–252]. In high-precision manufacturing domains such as aerospace, even minor wrinkles during composite layup can undermine structural integrity, while in electronics or consumer goods assembly, fabric folds or surface defects can disrupt downstream operations. Early detection of defect onset—rather than relying on post-hoc identification—enables timely corrective actions, reducing material waste, rework, and production delays. As a result, robust defect detection has emerged as a critical enabler for the large-scale adoption of advanced robotic systems in modern manufacturing environments.

Several defect detection applications in manufacturing employ traditional vision-based methods [253–255]. In such applications, a high-resolution camera captures images of the part. These images are then processed with conventional image segmentation methods that are based on pixel filtering and image gradients. Such methods are highly sensitive to external factors such as lighting conditions, camera intrinsics, anisotropic interactions of the components, etc. These traditional methods are predominantly used for defect detection related to dimensional errors. However, another class of processes exists where components might be deformable (e.g., prepreg composite layup), where methods that depend on well-defined geometrical features of the part might be unable to detect defects. Moreover, these conventional techniques require fine-tuning of certain parameters to ensure robustness and repeatability in detection. Defects generated during processes such as composite prepreg layup have features that vary based on several external factors. Such salient features are mainly a result of the part's interaction with ambient light and the geometrical appearance of the defects.

To overcome these issues, recently, defect detection methods based on deep learning have been gaining a lot of momentum [213, 256–258]. A key aspect in



(a) (b)

(c) (d)

Figure 10.1: Variation in the wrinkles formed during prepreg composite layup

ensuring the resilience of robotic systems lies in generating high-fidelity data conducive to training robust models. Researchers have explored implementing deep learning in applications with extensive availability of process data, such as images or thermal signatures from sensors [259]. The major challenge with deep learning methods is their inherent dependency on the availability of huge amounts of data. Additionally, this data needs to be processed and prepared (e.g., annotated) for utilization in deep learning model training. Such pre-processed data is not readily available for several manufacturing applications. Furthermore, manually collecting and generating such data may not be feasible due to time and cost constraints.

In processes where collecting online process data becomes infeasible, synthetic data can play a critical role in the deployment of deep learning models. With advancements in generative models and computer graphics, the learning community has pivoted towards use of synthetic data.[260–264]. Synthetic data generation has emerged as a valuable avenue in the pursuit of solving the data issue. Synthetic data can be effective in learning segmentation models if the data exhibits realism not only in texture and appearance but also preserves the innate physics of the original data. With conventional generative adversarial methods, capturing real data's physics is difficult. Synthetic images that look photorealistic and manifest the physical features of the original data can solve the data generation problem for defect detection in complex manufacturing settings. Therefore, synthetic image generation has a huge potential for enabling the large-scale implementation of deep learning methods for defect detection in manufacturing applications.

Synthetic image generation, as discussed, presents its own set of challenges in the context of deep learning. However, the traditional methods employed for synthetic image generation may not be sufficient when it comes to specific applications that deal with deformable objects, e.g., composite prepreg

layup processes. Composite prepreg sheets are highly pliable and have a peculiar appearance dependent on the material type, weave pattern, etc. Due to the compliant nature of the sheets, it is challenging to synthesize images that emulate defects of the real-world process. Prepreg composite layup is characteristically a low-volume process with the defects exhibiting an irregular pattern, complex deformations of the material, and anisotropic reflections when captured using a 2D camera (Refer Fig. 10.1). Hence, we need a solution that can accurately model the sheet's physical interactions under external constraints to simulate the defects that occur in-process.

Figure 10.2: Comparing Synthetic Images with the Real Images. **Note**: The images shown here are for representative purposes only. They do not correlate in terms of visual appearance.

This chapter presents a physics-informed synthetic data pipeline for robust wrinkle and fold detection in deformable-sheet manufacturing. By marrying high-fidelity thin-shell simulation with advanced CGI rendering, we create a large-scale, photorealistic image corpus that captures the true physical and visual complexity of sheet defects. Although composite layup can produce

many failure modes—air gaps, bridging, fiber distortion, etc. [106]—our focus here is on wrinkles and folds, whose distinct 2D patterns lend themselves to image-based segmentation.

A systematic framework is introduced to generate defect-prone sheet configurations: first, a validated physics model simulates how boundary conditions and material properties give rise to wrinkles; then, a texture-mapping pipeline renders these deformations under realistic lighting and camera models (see Fig. 10.2). Using this approach, over 10,000 synthetic images encompassing diverse wrinkle geometries and appearances were produced.

To ground our model in reality, we also collected 1,000 real in-process images of wrinkles from an industrial prepreg layup cell. These were combined with the synthetic set to form a hybrid training corpus. A Mask R-CNN network [265] was then trained in two stages—first on synthetic data, then fine-tuned on the real images. The resulting model achieves a mean Average Precision (mAP) of 0.98 on a held-out set of 200 real images, accurately predicting both bounding boxes and pixel-precise defect masks (see Fig. 10.13).

This physics-informed, simulation-driven approach bridges the sim-to-real gap, overcomes data scarcity, and delivers an off-the-shelf solution for online defect detection in prepreg composite layup. To facilitate further research, the full synthetic and real-image datasets have been made open-source and publicly available.

10.2 Overview of Approach

In this section, we will briefly outline the components of our defect detection system. The entire system can be subdivided into four main components: Real Image Collection, Synthetic Image Generation, Data Preparation, and

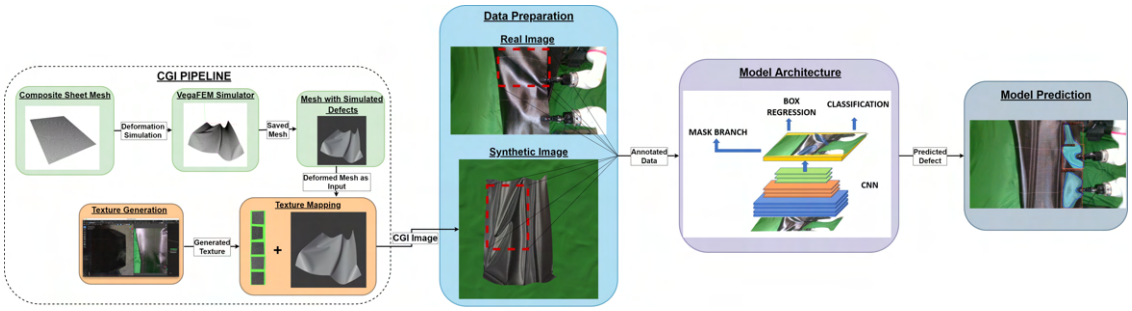


Figure 10.3: Process flow describing the system.

the Deep Learning Model. Fig. 10.3 gives an overview of these subsystems and the individual elements.

Real Image Collection: Although collecting real data for the layup process might be cumbersome, it is important that we capture the features embodying wrinkles that are not simulated. To achieve this, we perform a step-by-step layup and try to recreate scenarios that may lead to defect formation and capture the ones that lead to actual defects. We collect about 1,000 images using the approach described in Section 10.3.

Synthetic Image Generation: The synthetic imaging component consists of a FEM thin-shell simulator and the CGI module for computer graphics rendering. Our simulator here is based on the previous work by the authors [1] that builds an accurate mechanical model of the composite sheet. This simulator helps accurately predict sheet deformation under varied external constraints. We emulate the possible configurations of the sheet that can lead to the formation of anomalies on the sheet. Such anomalies signify the onset of a defect. Furthermore, we also simulate wrinkle defects for an already-conformed sheet on the composite tool. The simulator outputs a triangulated mesh of the sheet. This mesh then passes through a CGI pipeline that uses ray tracing to render a photo-realistic image. The rendering is improved by applying the custom texture we generate for the composite sheet. The entire process is detailed in Section 10.4.

Data Preparation: Once we have developed a hybrid dataset of real and synthetic images, we describe methods to annotate the data and formally define what constitutes a wrinkle or a defect. We describe the data preparation method in detail in Section 10.5.

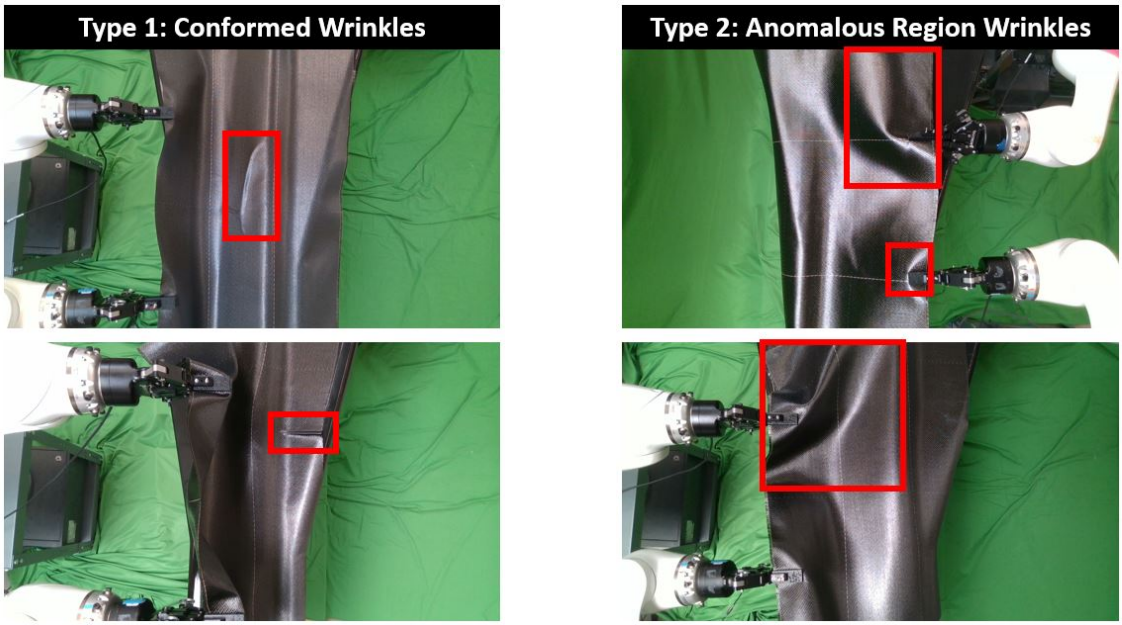
Deep Learning Model: After the data preparation and processing is completed, the properly annotated dataset can be used to train a deep learning model for predicting defects. As mentioned in Section 10.1, we use the Mask R-CNN architecture that outputs a mask of the predicted defect within a bounding box. The model architecture and framework are discussed precisely in Section 10.6.1.

Using the methodology presented in our work, we can design a robust system that can be deployed to detect defects in the composite layup process. Such a system can effectively detect wrinkles formed during composite layup and detect anomalous configurations of the sheet that may lead to defect formation. This obviates the need to remove the defect altogether. We will describe each of the aforementioned components comprehensively in the following sections.

10.3 Real Image Collection

To generate a dataset comprising of images of the defects formed during the actual process, we used an experimental setup with an industrial tool and two sheet grasping robots as shown in Fig. 10.1 and Fig. 10.2. We use a set of industrial grippers with custom 3D-printed attachments to grasp the sheet appropriately. The objective of the setup is to emulate a robotic composite layup process and capture various types of wrinkles and anomalies that may emerge during the process.

In this work, we have focused only on wrinkles as a class of defects for detection. Defining a wrinkle on a cross-section of the sheet that has already conformed



(a) (b)

Figure 10.4: The two types of wrinkles witnessed during composite layup. (a) The wrinkles formed on a conformed/draped portion of the sheet; (b) an anomalous region on the undraped portion of the sheet that signifies the onset of a wrinkle.

to the tool is straightforward. These wrinkles have clear, well-defined features, as can be seen in Fig. 10.4a. They are formed due to improper flexing of the sheet, leading to the formation of ridges.

A robust defect detection system should not only be able to detect already formed wrinkles but also predict if a certain configuration of the sheet is likely to form a wrinkle. This helps in avoiding defect formation altogether. To achieve this functionality, we try to flex and hold the composite sheet in configurations as depicted in Fig. 10.4b. These anomalous configurations are where, if a robot or an operator were to conform the sheet to the tool, a wrinkle would form. These configurations are commonly encountered and are the leading cause of wrinkle formation in the composite layup process. Hence, we capture features of such anomalous regions and classify them as defects.

Most parts in the composite industry have tubular (or tubular-like) cross-sections. This entails the tool being rotated about an axis to wrap the composite sheet around the tool's geometry. To incorporate this variation, we capture images of the wrinkles at different tool configurations (Refer to Fig. 10.4).

We used an overhead 2D camera with a 1920 x 1080 resolution to capture the images. There were slight variations in the lighting conditions during the data collection process. Furthermore, we also changed the position of the grasping robots to avoid introducing bias into the images. In this manner, we generate a dataset of 1,000 real images. This dataset consists of 800 images with a combination of Type 1 and Type 2 Defects and 200 images with no visible defects.

10.4 Synthetic Image Generation

The real image dataset generated using the methodology of Section 10.3 provides a good benchmark for describing a wrinkle. Since generating large amounts of real image data can be infeasible from the perspective of time and material costs, we need to rely on synthetic images to train a robust and accurate deep learning model. This section will describe the synthetic image generation pipeline in detail. The key feature of our proposed pipeline is the blend of accurate physics simulation that can emulate the Type 1 and Type 2 defects and a realistic texture of the composite prepreg material, capturing the anisotropic parameters to generate photo-realistic images. We use the sheet configurations that generated most of the defects as the basis for simulating and replicating the Type 1 and Type 2 defects. We also generated synthetic images with no perceivable defects.

We will describe every element of our synthetic image generation pipeline and present a methodology to generate accurate and photorealistic images of defects in the composite layup process.

10.4.1 Physics Based Simulator

To generate accurate synthetic images with realistic wrinkles, it is crucial to simulate the sheet precisely. Traditional FEM simulators are infeasible due to their slow convergence. In this work, we have employed a physics-based simulator that is based on the previous work done by authors in [1], built on top of the VegaFEM simulation library [29].

To generate synthetic imagery, we recreate the setup of Section 10.3 in the simulation environment. A triangulated mesh of the same size as the sheet is created and four constraints are added to replicate the sheet's grasping points. The aforementioned simulator can handle dynamic constraints. We move these

constraints so as to replicate the anomalous configurations experienced during the real image generation process. We discuss this process in detail in the next section.

10.4.2 Data sampling

Parameter	Distribution
P_1	$[0, U(0.7, 1.0)]$
P_2	$[1, U(0.7, 1.0)]$
P_3	$[0, U(0, 0.3)]$
P_4	$[1, U(0, 0.3)]$
τ_1	$[U(0.1, 1.4), U(0, -0.4), 1]$
τ_2	$[U(-2.5, -1.2), U(0, -0.4), 1]$
τ_3	$[U(0.1, 1.4), U(0, 0.4), 1]$
τ_4	$[U(-2.5, -1.2), U(0, 0.4), 1]$
V_1, V_2, V_3, V_4	$U(0.1, 0.3)$
n	$\mathcal{N}(0, 0.1)$

Table 10.1: Parameters and their distribution in data sampling.

In this section, we discuss our methodology to simulate the anomalous configurations. We generate uniform data by randomly sampling the thin-shell locations and world-coordinate trajectories of the four constraints located on the perimeter of the sheet mesh. The constrained vertices of the mesh are selected close to the perimeter of the rectangular sheet to replicate the scenario of Section 10.3. These constrained points are denoted by P_1, P_2, P_3, P_4. For each trajectory, the converged simulated shape of the sheet is only stored at the endpoint of the trajectory, regardless of the intermediate trajectory positions. Therefore, we simplify the trajectory of the constrained vertices into a straight line from the starting to the ending point. We represent the vectors joining starting and ending points by $\tau_1, \tau_2, \tau_3, \tau_4$, where each τ_i is sampled randomly as given in Table 10.1. The velocities of the constraint vertices are represented by V_1, V_2, V_3, V_4, and are also sampled randomly (Table 10.1). The

simulation timestep t is a constant fixed at 10 milliseconds in our sampling process. This enables the constrained vertices to reach the endpoints faster while still being able to span the desired search space.

A trajectory is discretized into multiple simulation timesteps. In each timestep, the four holding points move along their velocity vectors. To model the noise from the robot actuators, the planner, the sensors, etc., we applied a simple Gaussian noise n as a perturbation to the moving distance at each time step. The sampling parameters are in Table 10.1.

The data sampling process can be executed in parallel. After the completion of trajectory execution, we stop the simulator after 55 timesteps, which is sufficient for the simulator to converge and output the thin shell shape. With a simple multi-core implementation, the proposed pipeline can generate about 360 shapes per hour using a Core i7-9750H processor with six cores. By employing the proposed pipeline, we generated 10,000 mesh shapes, capturing varying wrinkling features. These shapes also capture some anomalous configurations we encountered during real image data collection.

10.4.3 CGI Pipeline

Our physics-based simulator and the data sampling methodology produce meshes that contain realistic wrinkles. However, to generate photo-realistic images that can be used for training the model, we must also render the produced sheet meshes realistically. To do so, a high-quality texture of the sheet is paramount. We employed CGI technology (described below) to generate a detailed procedural texture of the composite sheet used in the real data collection. This texture can then be applied to the 10,000 synthetically generated meshes to generate high-quality renders close to real images. In this work, initially, we experimented with generating an "Image Texture" by taking several

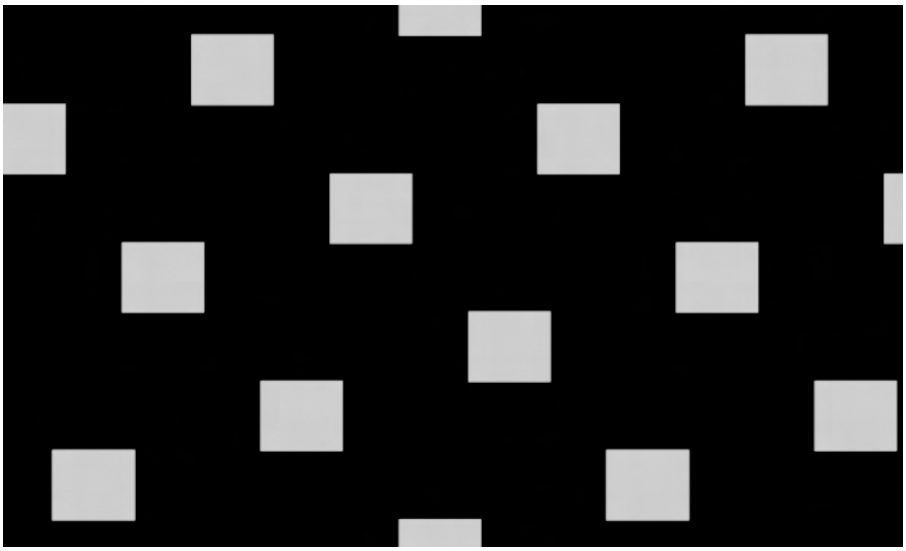


Figure 10.5: The black-and-white tiled matte indicates the fiber direction of the synthetic texture.

images of the carbon fiber sheet under varying lighting conditions and camera angles. However, the generated image texture could not precisely capture the light interactions of the sheet, rendering the images to look distinctly different from the actual image. Hence, we adopted a methodology to generate a procedural texture of the carbon fiber prepreg sheet rather than an image Texture.

We used Blender [266], an open-source animation and rendering software, to build the carbon fiber texture and render the images. A texture was constructed to imitate the real carbon fiber. We developed a custom CG texture in Blender using its node editing system. There were three main components to the texture: The first was mimicking the texture of the parallel-running fibers in the carbon fiber. This was accomplished by stretching a procedurally generated noise map in one dimension. The second step was to find a way to reorient these fibers at 90^o angles in certain areas, since the carbon fiber sheet is made of interwoven vertical and horizontal carbon fiber strips. This

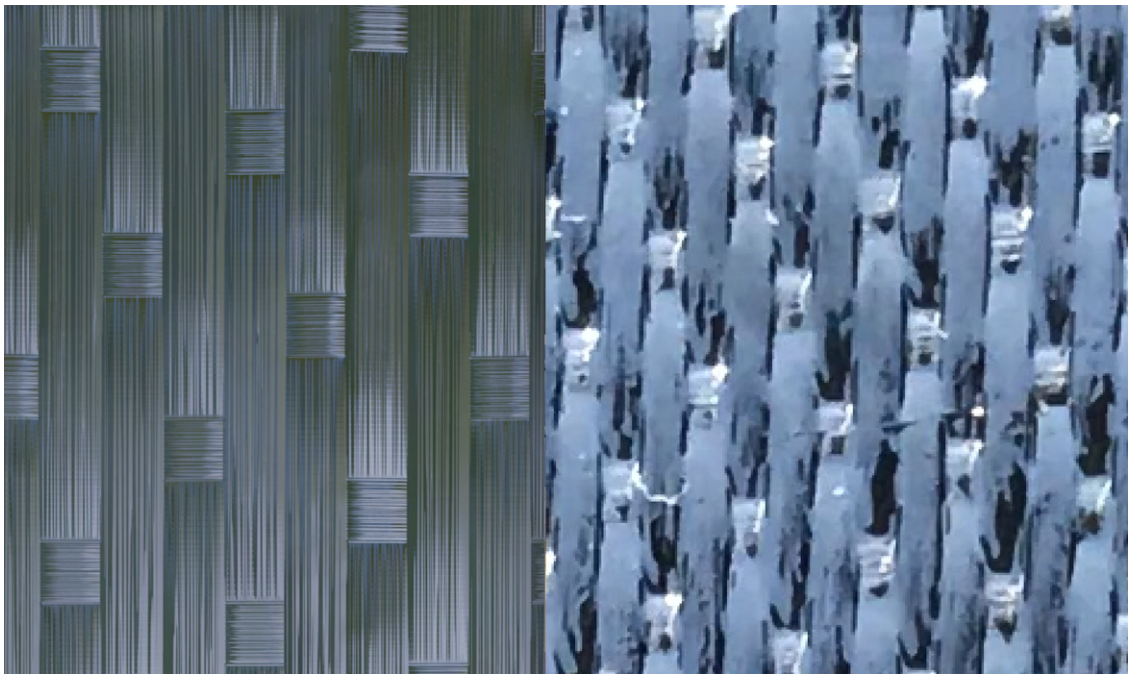


Figure 10.6: Left: zoom on the CG texture. Right: photo of real carbon fiber under a microscope (Visual Appearance may differ due to the scale).

was accomplished by procedurally generating an orientation map in different software (Matlab), which can be seen in Fig. 10.5. The Blender texture generator then oriented the virtual fibers of the texture horizontally, where the "orientation map" showed a pixel value of 0 (black), and vertically, where the "orientation map" showed a pixel value of 255 (white). The result is a close emulation of the interwoven appearance of the carbon fiber sheet. A close-up view of the CG texture is shown in Fig. 10.6. Although to the naked eye, the texture showcased in Fig. 10.6 might not look exactly similar in appearance on a macro level, the objective of the texture is to capture the salient features such as the shape of the wrinkle, appearance, etc. of the prepreg sheet. These features are sufficiently good for training the deep learning model.

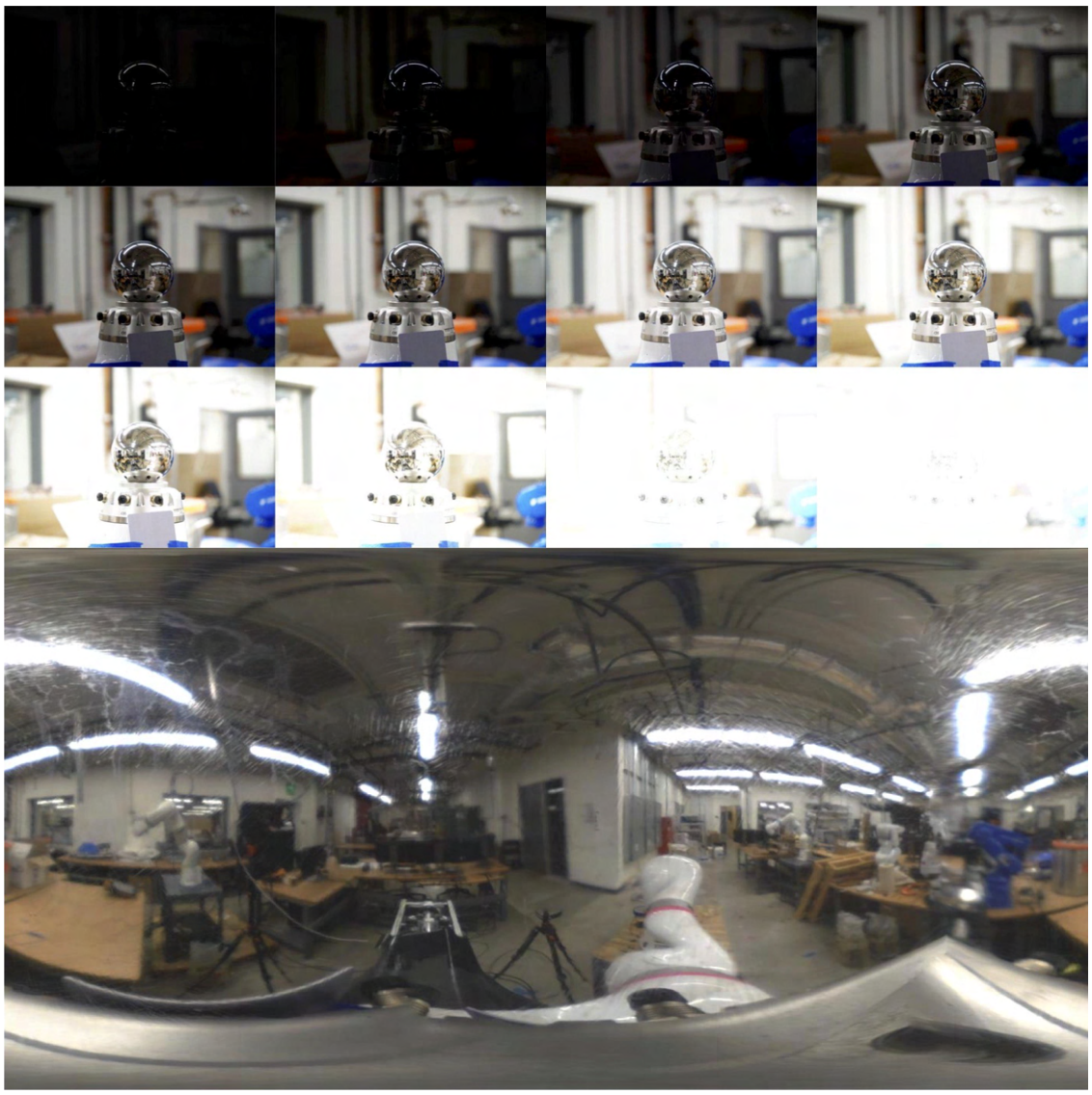


Figure 10.7: A collection of images at different exposures (top) was compiled to create a 360° panoramic HDRI (bottom).

The generated texture can then be applied to the 10,000 synthetically generated meshes. When virtually lit and rendered, these objects look close to photorealistic (see Fig. 10.2). We now describe this process.

First, we need to match the lighting of the virtual environment to the lighting of the real environment. This entails capturing a spherical panorama of our manufacturing space. We do this by photographing a mirror sphere and then wrapping this 2D image onto a 3D photo-sphere in the computer. We photographed a standard $3''$ chrome ball commonly used in CGI applications from two different angles and at several different levels of camera exposure to capture the entire spectrum of light present in the workroom. These images were then wrapped to create the 360° high dynamic range image (HDRI) shown in Fig. 10.7.

Next, we apply the texture and the lighting setup to produce renders that closely replicate real-world images. The synthetic images generated using this framework can be juxtaposed against real images in Fig. 10.2. We rendered the 10,000 meshes to produce 10,000 images of the composite sheet in varying configurations. The rendering was performed on a core i7-11700 (8 core, 2.5 to 4.9 GHz) processor boosted with an NVIDIA GeForce GTX 1660 Ti 6GB GPU. With this configuration, the average time to render a single image was about 5 seconds. Using these methods, a digital rendering environment can then be created with optical materials and lighting conditions that match the real manufacturing environment; see Fig. 10.8. We achieved this by importing the HDRI images into blender and consequently performing ray tracing.

The proposed CGI pipeline helped us in augmenting our real dataset of 1,000 images. This hybrid dataset with 11,000 images is sufficient for feature extraction and is employed for training the deep learning model proposed in Section 10.6. Fig. 10.9 summarizes the synthetic image generation approach.

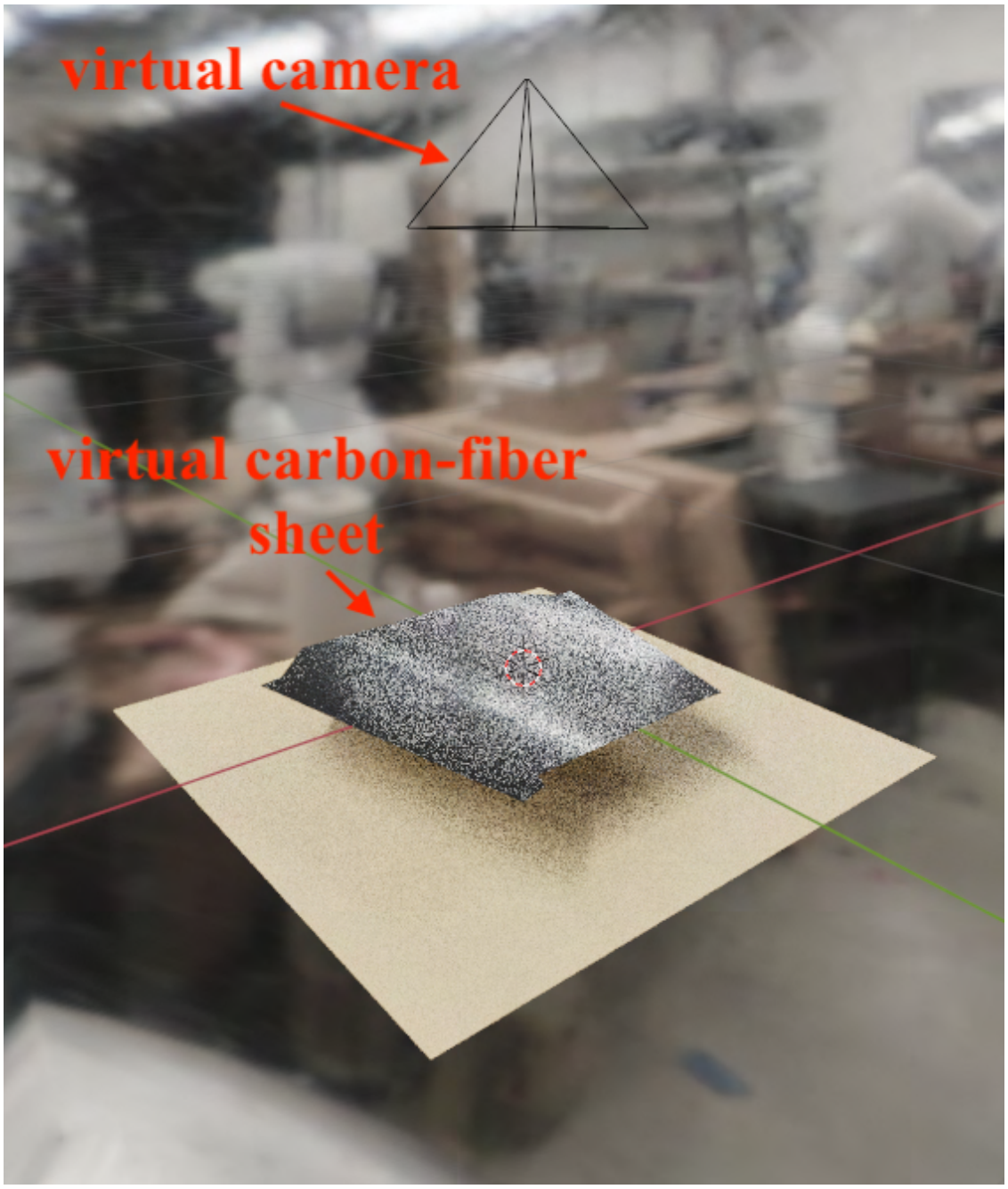
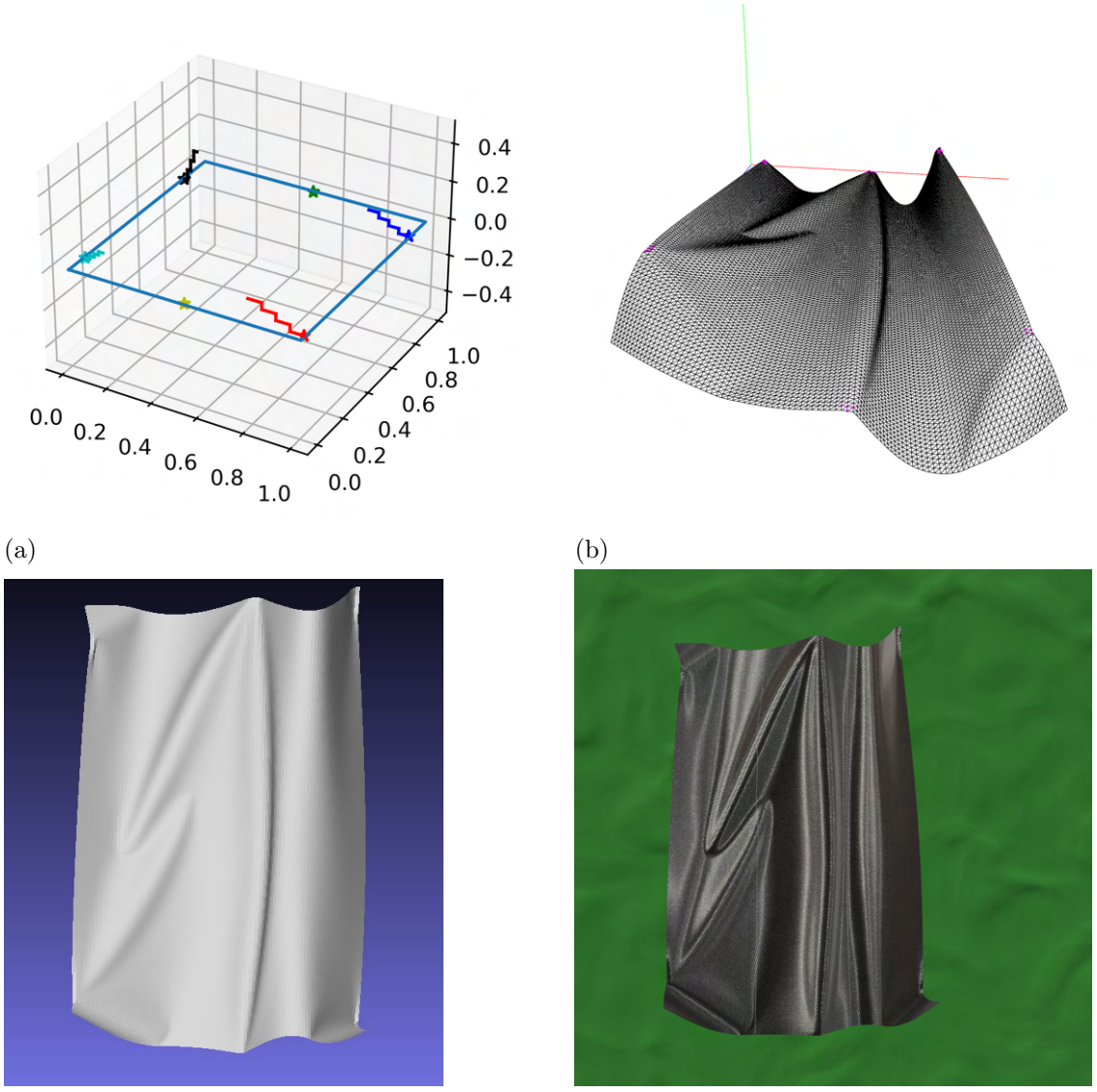


Figure 10.8: The virtual environment.



(a) (b)

(c) (d)

Figure 10.9: (a) Trajectory of the four holding points; (b) the shape after executing the trajectory in our physical-based simulator; (c) output mesh before rendering; (d) the CGI-rendered mesh.

10.5 Data Preparation

The hybrid dataset generated using the methodology described in Sections 10.3 and 10.4 needs to be appropriately annotated for model training. We described the key features of wrinkles in Section 10.3 and categorized them into Type 1 and Type 2 defects. We used the same description to generate the annotations for the dataset. We classify both Type 1 and Type 2 defects as one class called "wrinkle". Fig. 10.10 shows example annotations on a real and a synthetic image.

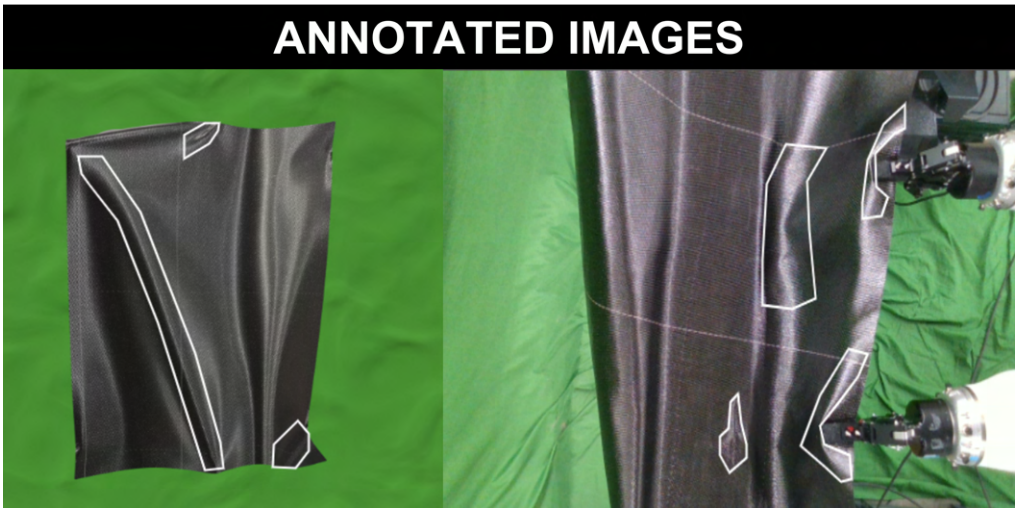


Figure 10.10: Annotations depicting the Type 1 and Type 2 defects. Kindly note that both Type 1 and Type 2 Defects are annotated as one single class "Wrinkle"

We used an online open-source tool called makesense.ai to perform the annotations. This tool helped us generate polygon annotations that engulf the image's defected region ("wrinkle") and can be saved as JSON files. All the images were annotated using only one class per annotation called "wrinkle". Images with no visible defects received no annotation. Once we complete the data preparation phase, we can then use the annotated data to train our proposed deep learning network.

10.6 Model Description

Our modeling methodology is based on the image segmentation deep learning model Mask R-CNN [265]. We use a two-stage training procedure to train the image segmentation model to detect defects in the composite sheet. The first stage is a pre-training stage that uses a large synthesized dataset. The second stage is a fine-tuning stage that utilizes a small real-world dataset. We further introduce a scaled mixing technique to the fine-tuning stage so that the synthesized dataset is also employed in the fine-tuning stage; this avoids catastrophic forgetting. Our deep learning model architecture and training details will be described in Sections 10.6.1 and 10.6.2, respectively.

10.6.1 Model Architecture and Settings

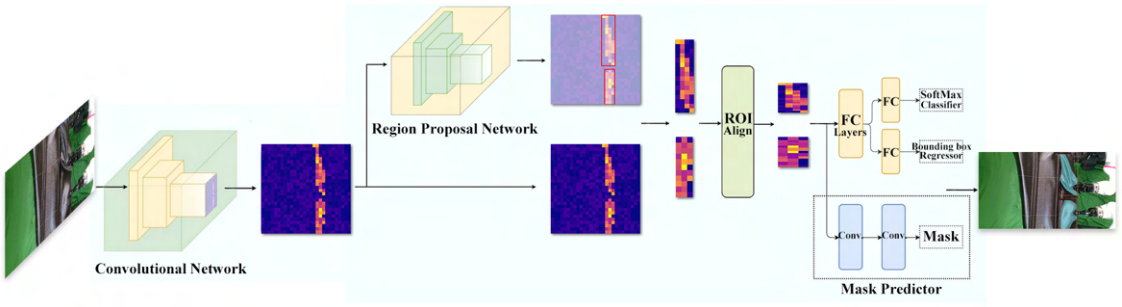


Figure 10.11: Mask R-CNN architecture.

We applied the Mask R-CNN shown in Fig. 10.11 as our base model for the image segmentation task. After trying typical backbones, including VGG [267], Inception [268], and DenseNet [269], We select a ResNet-50 for its best performance among them as the backbone network which extracts features from the input image. Feature Pyramid Network (FPN) [270] is used as the neck and backbone network. This architecture better utilizes the extracted features in a multi-scale manner.

Ultimately, we use two heads applied as the output of the model, one for outputting the bounding box and another for predicting the mask. The two heads share a Region-of-Interest (RoI) network that proposes the potential RoIs on top of the neck network.

The Mask R-CNN we use has been pre-trained on the Microsoft Common Objects in Context (MS COCO) [271] dataset. This dataset consists of 1.5 million labeled images from 80 categories. Our pre-trained model learns rich common visual patterns of images after performing transfer learning by iteratively adjusting modeling parameters through in-domain images of the carbon fiber sheet. The proposed model will be able to detect task-specific patterns (i.e., defects) quickly. Compared to training a model from scratch, we need less redundant data to let the model start from learning the basic common patterns.

Our implementation is based on the MMDetection framework [272]. The model is trained on a single NVIDIA GeForce RTX 2070 GPU. For the model that uses ResNet-50 as the backbone, the entire training process takes about 12.5 hours. The memory use is approximately 4.4 GB, and the inference runs at about 14.5 fps. We also explored variants of the model architecture that improve the model performance; they will be discussed in Section 10.7.3.

10.6.2 Training

Despite the high quality of our synthetic images, the sim2real gap [273, 274] still exists. For example, the real carbon fiber sheet may have a slightly worn surface, which causes the reflectance to be uneven. The physical specifications of a real sheet may also change over time. There may be temperature variations. The complex lighting conditions in our production environment are also difficult to reconstruct perfectly. The movements of the real robots

are computationally expensive to be completely reproduced in the simulator. Finally, the error from the physical simulator itself and the imperfection of the rendering engine should also be considered. These minor errors accumulate and cause a gap between the images of the real and virtual sheets. The time and processing cost of pursuing a higher degree of realism in the synthetic data increases exponentially, even higher than the cost of collecting real data. Thus, a better solution is to compensate for the gap through real images.

However, directly mixing the synthesized data with real data and then using the mixed dataset to train the model would cause an inductive bias. This might cause the model to overly rely on the features learned from virtual images while ignoring real images. We desire to learn the physics of the sheet from the synthetic data, but then use the real data to compensate for the details in the real production environment. The problem occurs due to the data imbalance, given that synthetic images outnumber real ones by 10 to 1 in our dataset.

One approach to this problem is using a **2-stage training** method. Sajjan et al. [275] proposed to use a two-stage training procedure that first pre-trains the network on a large out-of-domain real-world dataset, then fine-tune it on a smaller in-domain synthetic dataset for the robotic grasp of transparent objects. Inspired by their method of learning basic patterns in the first stage and then transferring knowledge into the target real domain in the second stage, we applied a two-stage training method that first pre-trains on a large synthetic dataset and then fine-tunes the model on the smaller real dataset.

In the pre-training stage, the model learns typical patterns of the defects formed in the composite sheet and the deformation of the sheet caused by the movement of robots. By using our physics-informed simulator to simulate the deformation of the composite sheet, we implicitly introduce the physical

knowledge as a bias into a deep learning model. This helps the model understand the generating process of a defect. In the fine-tuning stage, the model learns task-specific knowledge of how the defects look in the real layup, which includes additional noise originating from lighting conditions, robot actuation, occlusions, etc.

Another way to mitigate the gap between the real and synthesized datasets is to regard it as a multi-task problem where the model has to learn both the physics-based behavior through the synthetic dataset and the expert knowledge about the defects in the real world through the real dataset. We use **scaled mixing** that entails training with each real image several times, whereas training only once for each synthesized image. This is a technique that has succeeded in training multilingual language models where there is a huge disparity between the data set sizes of larger languages vs smaller languages; each language is a subtask to learn. Instead of directly mixing real and synthetic data, we replicate the real data k times before mixing (we use $k = 5$). Furthermore, we combine this approach with a 2-stage training by applying scaled mixing in the second stage. The performance and analysis of these training techniques in our experiments will be discussed in Section 10.7.1.

10.7 Results

We evaluated our method on images acquired from a robotic composite layup cell. Section 10.7.1 will discuss the experimental setting. We will also discuss techniques to enhance our model (Section 10.7.2). The experimental results are analyzed in Section 10.7.3. Finally, we also discuss failure cases where our model failed to produce a successful prediction (Section 10.7.4).

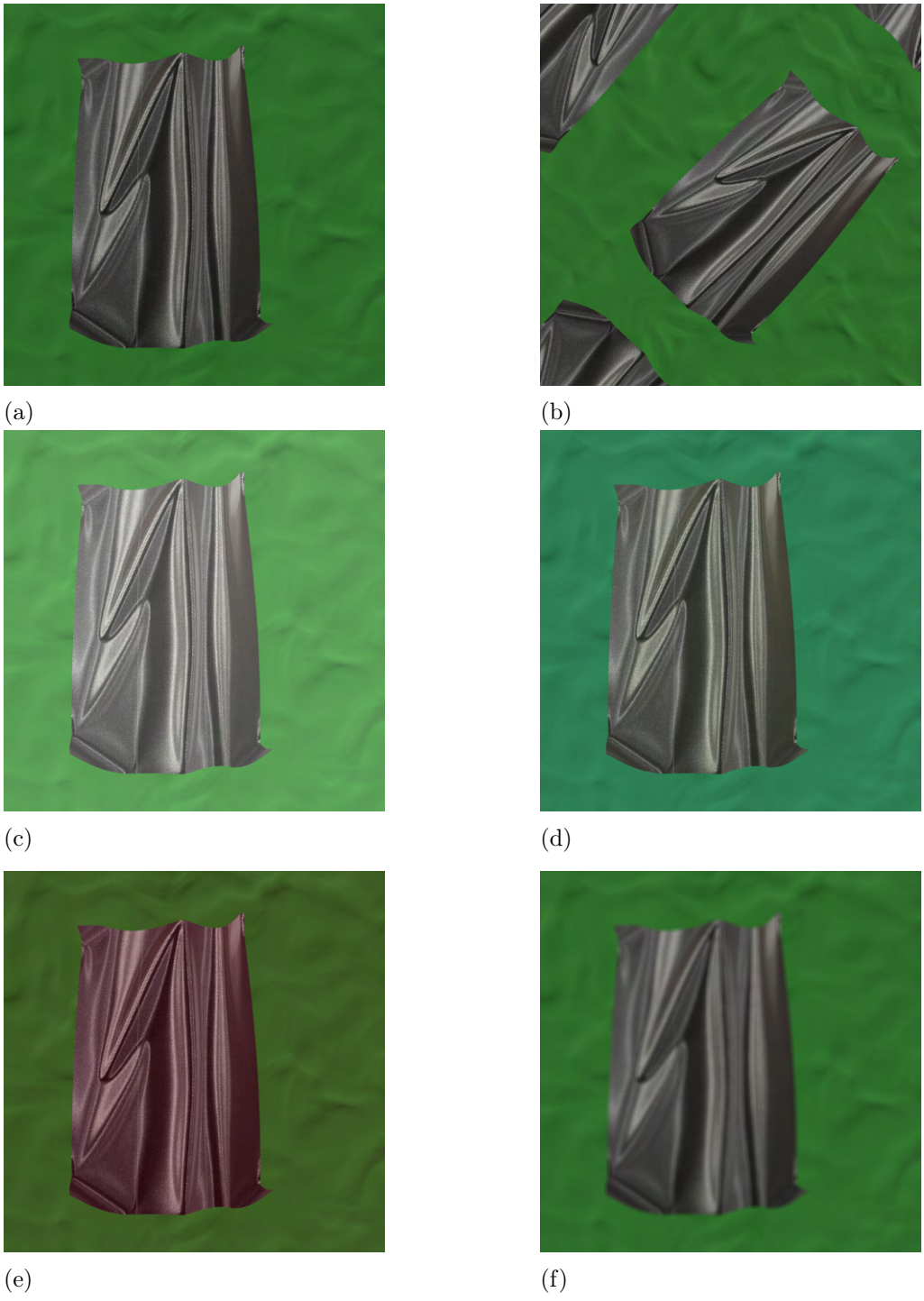


Figure 10.12: Data augmentation methods. (a) Original image, (b) random shift, rotation, or scale, (c) random brightness and contrast, (d) random hue, saturation, value, (e) randomly shifted RGB values, and (f) random blur.

10.7.1 Training settings

The real dataset consists of 1,000 images, and the synthetic dataset has 10,000 images. We split the real dataset to 6:2:2 for training, validation, and test sets. For models trained in 2 stages, we use the entire synthetic dataset for training in the first stage, whereas we use the validation set of the real dataset for model selection.

In both stages, we apply data augmentation techniques comprising random shifts, rotations, and scalings of the images. Hence, before an image enters our training pipeline, it will initially pass through the data augmentation pipeline. With a probability of 0.5, the image is randomly transformed by shifting it by a pixel amount uniformly sampled from $U[-0.0625, 0.0625]$ for the x and y axes, respectively. Subsequently, the image is rotated by an angle uniformly sampled from $[-45^o, 45^o]$, with a probability of 0.5. Finally, with a 0.5 probability, we scale the image by a factor uniformly sampled from $[0.9, 1.1]$. These data augmentation methods largely expand the dataset size and introduce an inductive bias whereby the patterns of the defects are invariant to the transformations from the camera. This, in turn, helps the model learn more robust representations. While our data augmentation pipeline is randomized, fixing the random seed in our experiment ensures the reproducibility of the images when re-running the experiment.

We employ the Intersection-over-Unification (IoU) area metric to evaluate the accuracy of our deep network. This metric divides the area of intersection of the predicted defect region and the ground truth region by the area of the unified region. To define whether the output is considered accurate, we use the common IoU threshold of 0.5, based on work done in the PASCAL Visual Object Classes (VOC) challenge [276]. This means that a predicted defect

with IoU larger than the threshold of 0.5 is considered positive, whereas an IoU below 0.5 is regarded as negative.

We implemented our deep learning pipeline using the PyTorch framework [277]. We trained 12 epochs for each stage and applied Stochastic Gradient Descent (SGD) with a momentum of 0.9 and weight decay of $1e-4$ to optimize our model. The learning rate is $2e-3$. A step learning rate scheduler is employed; hence, in the 8th and 11th epochs, the learning rate will be scaled down by $\times 0.1$. We also applied a linear learning rate warm-up for 5000 training steps with a ratio of $1e-5$.

10.7.2 Model enhancement

	Test mAP					Train mAP							
	Def.	**Pre.**	**Und.**	**Avg.**	**Gain**	**Def.**	**Pre.**	**Und.**	**Avg.**	**Gain**	**Mem.**	**Inf.**	**Train.**
Base	0.938	0.832	0.9	0.89	-	0.938	0.821	0.875	0.878	-	4.4 GB	14.5 fps	0.5 hrs
+ Syn. data	0.944	0.865	0.925	0.911	2.4%	0.938	0.844	0.9	0.894	1.82%	4.4 GB	14.5 fps	5.5 hrs
+ 2-Stages	0.956	0.898	0.95	0.935	5.02%	0.949	0.868	0.917	0.911	3.80%	4.4 GB	14.5 fps	5.5 hrs
+ Scaled	0.977	0.932	0.975	0.961	8.02%	0.956	0.892	0.945	0.931	6.04%	4.4 GB	14.5 fps	7.5 hrs
+ Scaled 2-S.	0.987	0.945	1.0	0.977	9.81%	0.961	0.903	0.958	0.941	7.14%	4.4 GB	14.5 fps	12.5 hrs
More aug.	0.940	0.838	0.925	0.901	1.24%	0.939	0.830	0.942	0.904	2.92%	4.4 GB	14.5 fps	0.5 hrs
+ Scaled 2-S.	0.989	0.948	1.0	0.979	10.0%	0.966	0.912	0.967	0.948	8.01%	4.4 GB	14.5 fps	12.5 hrs
More layers	0.936	0.838	0.9	0.891	0.15%	0.942	0.833	0.875	0.883	1.01%	6.4 GB	11.7 fps	0.72 hrs
+ Scaled 2-S.	0.980	0.939	0.975	0.965	8.39%	0.960	0.9	0.942	0.934	6.38%	6.4 GB	11.7 fps	18 hrs
Cascade	0.942	0.847	0.925	0.905	1.65%	0.947	0.848	0.95	0.915	4.21%	6.0 GB	9.6 fps	0.6 hrs
+ Scaled 2-S.	0.990	0.947	1.0	0.979	10.0%	0.963	0.918	0.967	0.949	8.13%	6.0 GB	9.6 fps	15 hrs
ResNeSt	0.947	0.841	0.95	0.913	2.55%	0.950	0.845	0.95	0.915	4.21%	5.5 GB	13.3 fps	1 hrs
+ Scaled 2-S.	0.989	0.950	1.0	0.98	10.08%	0.965	0.913	0.975	0.951	8.31%	5.5 GB	13.3 fps	25 hrs

Table 10.2: Results of experiments. All gains are compared to the "Base" model. Detailed interpretation is available in Section 10.7.3.

Apart from the experiments that explore the effectiveness of the training techniques and the use of synthetic data, we performed additional experiments to enhance our model. This helped us devise effective auxiliary schemes for our application, as described next.

Advanced backbone network: We explored the use of the state-of-the-art backbone network ResNeSt-50 [278], which is an advanced version of the

ResNet backbone we used as the base model. ResNeSt-50 introduces channel-wise attention to the different network branches of ResNet to better capture cross-feature interactions and learn diverse representations. To analyze the trade-off between the cost and accuracy improvement, we replace the ResNet-50 network with ResNeSt-50.

Deeper model: We use a deeper backbone network, ResNet-101 to replace the ResNet-50 network in this experiment. This helped us examine the cost vs accuracy trade-off of using a deeper model.

Advanced model architecture: We apply a Cascade Mask R-CNN [279], which improves the Mask R-CNN architecture by introducing a multi-stage object detection architecture. This consists of a sequence of detectors trained with increasing IoU thresholds to replace our basic Mask R-CNN architecture. This helped us explore the trade-off between a more complex and advanced architecture and accuracy improvement.

Additional data augmentation: We introduce additional data augmentation techniques into the pipeline to explore the influence on model performance due to this subsequent inductive bias. We include randomly changing brightness and contrast with a factor uniformly sampled from $[-0.2, 0.2]$ with a probability of 0.2; randomly shifting RGB channel by pixels uniformly sampled from $[-10, 10]$ for each channel with a probability of 0.1; randomly changing the hue, saturation and value of the input image with a quantity uniformly sampled from $[-20, 20]$, $[-30, 30]$, and $[-20, 20]$, respectively, by a probability of 0.1; randomly shuffling the RGB channels by a probability of 0.1; and randomly imposing blur with the kernel size uniformly sampled from $[3, 7]$ with a probability of 0.1. These data augmentation methods introduce inductive biases such that the patterns of defects are invariant to the color of the carbon

fiber sheet and the camera parameters. The data augmentation methods are shown in Fig. 10.12.

10.7.3 Analysis of results

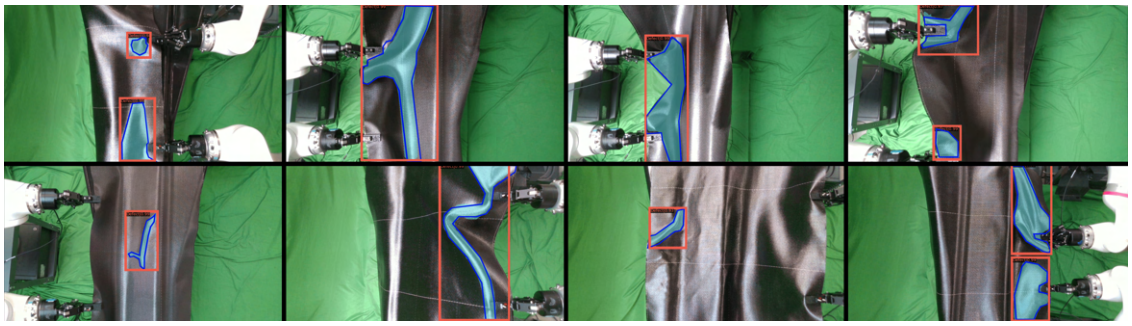


Figure 10.13: Model prediction examples. The orange rectangle marks the predicted bounding box, the transparent cyan region represents the predicted mask, and dark blue polygons denote the ground truth.

The results of our experiments are shown in Table 10.2. We compare the performance of the mask R-CNN-based base model and the four model enhancements discussed in Section 10.7.3.

For each of the five variants, we first train them without synthetic data and then demonstrate the improvement with synthetic data based on the best-observed training techniques. The best training technique combines Scaled mixing and 2-Stage training, denoted as "+ Scaled 2-S.". The results show that for all variants, the synthetic data gives a gain of about 8 to 10%.

In the first variant, we experimented with the base model denoted as "Base" and performed ablation studies for training techniques. "+ Syn. data" denotes directly mixing the real and synthetic data set. It provides only a nominal increment of 2.4% compared to the model trained without synthetic data. By introducing "2-Stages" training, the data imbalance problem is moderated. However, this approach is still limited by the "catastrophic forgetting" problem. This is reflected in the model trained with "Scaled" mixing, where the gain

increases from 5.02% to 8.02%. Finally, by combining the two approaches, the model achieved an overall gain of 9.81%.

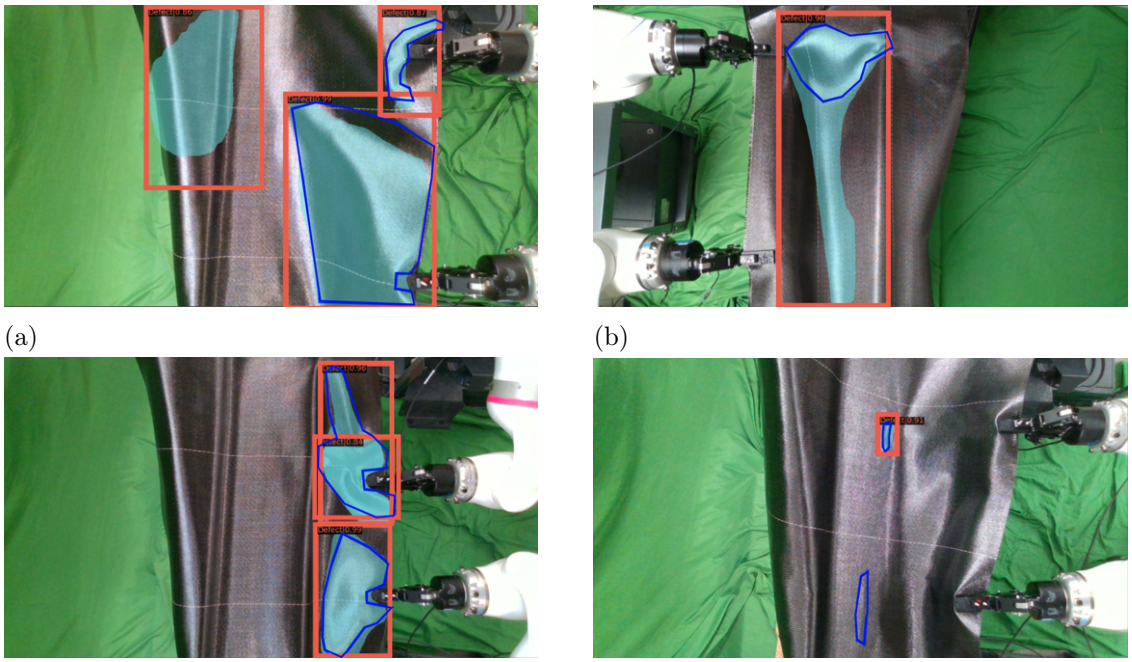
The advanced data augmentation ("More aug.") increased average mAP by 1.24% and slightly improved the trained model on synthesized data. Advanced backbone model ResNeSt-50 ("ResNeSt") achieved the highest 10.08% improvement, but the memory use increased to 5.5 GB while the inference speed decreased to 13.3 fps, and training time increased to 25 hours. Cascade architecture ("Cascade") also achieved a higher mAP but increased the memory use to 6.0 GB. The inference speed decreased to 9.6 fps while the training time increased to 15 hours. However, using this deeper model, ResNet-101 with 101 layers ("More layers"), the average test mAP gain for the model with synthetic data decreased from 9.81% to 8.39%. This may be due to the over-complexity of the model for our application. Memory usage increased to 6.4 GB, inference speed decreased to 11.7 fps, and training time increased to 18 hours.

The results show that our training techniques are effective. The augmentation methods do improve the performance of model architectures such as the Cascade architecture or the advanced backbone model. Additionally, advanced data augmentation helps the model to generalize the dataset better. That said, an upgrade to the model architecture incurs the cost of lowering the computational efficiency. Due to our problem's relatively lower complexity compared to a general image segmentation model, which may contain millions of samples, a highly complex model with too many layers eventually deteriorates the performance.

Overall, our method achieved the highest average test mAP of 0.98. A proper choice of model complexity and the trade-off between higher performance due

to advanced model architecture vs efficiency is important in real-world applications. Moreover, additional data and training "tricks" also require more training time.

10.7.4 Analysis of failure cases



(a) (b) (c) (d)

Figure 10.14: Failure cases. The orange rectangle and transparent cyan regions mark the bounding box and mask predicted by the model, respectively; the dark blue polygons denote human annotation. (a) False positive (Unexpected region), (b) Confused region, (c) Redundant prediction, (d) Annotation mistakes.

We further analyzed the test samples where the model failed to make satisfactory predictions in the test set. We summarize these instances in the following four cases, and recommend possible solutions.

Failure case 1: False Positives The model sometimes predicts an unexpected region; this happens very rarely. An example of this case is shown in Fig. 10.14a, where the region on the left side predicted by the model is

unexpected. This can be avoided simply by using model ensembling [280], i.e., training multiple models and making predictions by voting. Although deep learning models always have the risk of behaving unexpectedly, these behaviors usually vary between models. Hence, by ensembling the results, the weight of each unexpected prediction will be largely reduced, and thus, the risk of predicting unexpected regions will be largely avoided.

Failure case 2: Confused Region The edge of the composite tool overlaps with the wrinkle, making it hard to distinguish the defect and defect-free region as shown in Fig. 10.14b. Such local features of the tool sometimes make it difficult to locate the wrinkle when the defect lies in the vicinity of these features. This is a less obvious case where the boundary between defective and defect-free regions is overlapping. This could be attributed to the high sensitivity of the model, which would try to catch all potential defects. However, such sensitivity is preferred in our intended application, where a false negative that potentially causes layup failure is more detrimental than a false positive. The model may implicitly be imposed with such inductive biases by the data annotation process. A high sensitivity may also be a possible cause of the failure in case 2. Like case 1, model ensemble methods could also solve this case.

Failure case 3: Redundant Prediction In this case, the same wrinkle region is predicted multiple times, as shown in Fig. 10.14c. The upper wrinkle has been marked by the model twice. Although reducing the test accuracy, this case does not influence real-world applications, and we can merge the predictions.

Failure case 4: Annotation mistakes Fig. 10.14d shows two examples of such mistakes. The top annotation in the image is very small. Therefore, a small perturbation of the model prediction will cause the IoU metric to

recommend it as a failed prediction. However, the prediction is completely acceptable in the actual application. The bottom annotation in the image is a weak wrinkle missed by the model. It is an unexpected defect due to reusing the same composite sheet several times during the real image collection process. Furthermore, there might be some ambiguity amongst human annotators as to whether to annotate such weak wrinkles as a defect or not. An improved annotation and data collection procedure could easily handle these annotation errors.

10.8 Summary

In this chapter, a physics-informed, deep learning-based framework for detecting wrinkles and folds in sheet-like deformable objects using photo-realistic synthetic data is presented. We demonstrated how combining a high-fidelity physics-based simulation with advanced CGI rendering techniques enables the generation of realistic training datasets for defect segmentation, which is critical for domains where large-scale real data collection is impractical.

This chapter presented a complete pipeline: simulating physically plausible wrinkle formations, rendering them into high-quality images, and training a Mask R-CNN model for instance-level segmentation of defects. By accurately localizing every visible wrinkle or fold, the trained model empowers robotic systems to sense the onset of defects and adapt their manipulation strategies accordingly, enabling online failure recovery and enhancing process resilience.

The experiments validate the effectiveness of hybrid training, first pretraining on synthetic images, then fine-tuning with a limited set of real-world process data. This two-stage approach achieved a mean Average Precision (mAP) of 0.98 on real defect images, demonstrating that synthetic data, when generated

with strong physical priors, can significantly boost model performance and generalization.

Beyond achieving high segmentation accuracy, this chapter highlights several broader contributions: (1) A physics-informed synthetic data generation framework that captures realistic deformation behaviors, (2) A detailed CGI-based methodology for generating photo-realistic textures and defect appearances, (3) An ablation study identifying the most effective modeling and training strategies for defect detection, and (4) A rapid, scalable process for dataset generation and model deployment in industrial settings.

The system developed here can be deployed online in production environments, enabling real-time monitoring of composite layup or other sheet manipulation processes. Such a robust and adaptive defect detection capability is essential for improving process quality, reducing scrap rates, and ensuring the scalability of robotic automation in high-performance manufacturing tasks. Beyond the specific application of defect detection in composite layup, this work exemplifies a broader principle that runs throughout this dissertation: the fusion of physics-based modeling with modern AI methods can enable scalable, robust, and explainable solutions for deformable object manipulation. By grounding deep learning in physical realism—through simulation, structured data generation, and task-informed model design—we can overcome the challenges of data scarcity, uncertainty, and generalization that often limit the deployment of learning-based robotics in industrial environments.

Chapter 11

Conclusions

11.1 Intellectual Contributions

In this dissertation, we introduced a novel physics-informed learning paradigm with the motivation to advance the field of robotic manipulation, specifically for deformable objects in high-precision industrial environments. We introduced a class of three deformable objects that are understudied, yet frequently encountered in high-performance industrial environments: (1) a large-scale sheet (2D), (2) a compliant tool (2D), and (3) a shell-like deformable package (3D). We discussed how each of these objects demands a fresh perspective for enabling robots to operate effectively in semi-structured and unstructured environments, where deformability introduces significant complexity. To meet these challenges, this dissertation proposes a structured framework that integrates physics-based priors into the key components of robotic intelligence: simulation, planning, learning, and anomaly detection.

First, this dissertation demonstrates how simulation-based learning, with physics-informed object models, enables accurate planning and robust anomaly detection for complex deformable objects. Second, it incorporates advanced learning frameworks to capture how tool compliance introduces highly non-linear object dynamics—exemplified in industrial tasks like robotic screwdriving,

thereby crafting models that are data-driven as well as physically explainable. Third, the concept of task-informed physics constraints is introduced, which illustrates how planning for large deformable sheets can explicitly account for manufacturing process requirements. Lastly, this dissertation presents compositional learning frameworks that decompose complex robotic tasks—such as bi-manual manipulation and multi-step task sequencing—into modular, learnable subtasks, promoting generalization, interpretability, and scalable policy design.

Together, these ingredients form a blueprint for crafting robotic agents that move closer to achieving human-level dexterity in deformable object manipulation, while meeting the rigorous demands of industrial applications.

To summarize, the specific contributions of this dissertation are:

- Physics-informed Simulation for Complex Deformable Objects: Development of structured simulation frameworks for large sheet-like objects and closed-form packages, integrating real-world parameter estimation and physics-based modeling.
- Task Sequencing Policies from Expert Knowledge: Introduction of methods to capture and model human expert task planning for deformable object processes, incorporating both performance and effort-based preferences through inverse reinforcement learning.
- Simulation-based Manipulation Planning: Creation of grasp planning and manipulation strategies for precision tasks using simulation-guided search, balancing task objectives and physical constraints.
- Compliance-aware Modeling for Manipulation Under Uncertainty: Formulation of physics-informed models for screwdriving under uncertainty, demonstrating how active and passive compliance influences dynamic behavior and success rates.

- Physics-driven Synthetic Data for Anomaly Detection: Design of a hybrid simulation and CGI framework to generate photo-realistic defect datasets for training deep learning models for wrinkle and fold detection in sheet-like objects.

Collectively, these contributions lay the foundation for a new paradigm in robotic manipulation — one that bridges structured physical knowledge with flexible learning models to achieve robustness, scalability, and safety in the manipulation of complex deformable objects.

11.2 Anticipated Benefits

The cornerstone of this dissertation rests on a fundamental belief: robots will profoundly transform human life, reshaping how we work, create, and live on this planet. The ideas and methods proposed in this dissertation represent a small but critical step toward this future, one where robots serve as collaborators, extending human capability across a wide spectrum of domains. Deformable object manipulation is just one essential building block among many that must be developed to realize this vision.

From both technical and societal perspectives, the anticipated benefits of this work include:

- *Explainability and Interpretability:* The physics-informed learning paradigm proposed here is built upon the principle that robotic decision-making must be understandable and explainable. By grounding manipulation policies in physical principles, this work ensures that robots can reason about their actions in ways that are transparent and verifiable — an essential requirement for building trust and ensuring safety at scale, particularly in industrial deployments.

- *Economic upliftment with improved quality of life*: Industries across manufacturing, logistics, and assembly are experiencing an acute shortage of labor willing to perform repetitive, strenuous, and hazardous tasks. Many of these tasks involve deformable object manipulation under uncertainty, where manual dexterity has historically been irreplaceable. This dissertation lays a foundation for deploying robots to take over these physically demanding tasks, fostering economic growth while allowing humans to pursue more intellectually engaging, creative, and safer forms of work, ultimately leading to a higher quality of life.

- *Faster Deployment and Greater Adaptability:* A major bottleneck in traditional robotics has been the rigidity of preprogrammed systems, which lack flexibility for high-mix, low-volume (HMLV) industrial environments. Meanwhile, purely data-driven learning systems have struggled with generalization and robustness. By combining structured physical knowledge with learning, this dissertation proposes a solution that accelerates robot deployment timelines and extends their adaptability across a wide range of tasks, especially those involving complex, variable deformable objects where conventional automation was previously uneconomical.

- *Technological Advancement Toward Human-like Learning:* The physics-informed learning framework introduced in this dissertation moves robotic systems closer to human-like reasoning and modular learning. Much like the human brain, where specialized modules interact to perform complex tasks, the proposed compositional, physics-grounded architectures enable robots to operate safely, learn effectively from limited data, generalize across conditions, and recover from unexpected failures.

Overall, the anticipated benefits of this research advance the broader field of robotics toward a future where intelligent machines can adapt, collaborate, and scale across real-world industrial applications, driving profound impacts in automation, economic sustainability, and human well-being.

11.3 Future Directions

This dissertation represents only a small but meaningful step toward enabling robots to handle the complexity of deformable object manipulation at human-level skill and adaptability. While the contributions here lay important groundwork, many exciting avenues for future research remain, offering opportunities to further advance toward the broader goal of building intelligent, resilient, and explainable robotic systems.

Several key directions for future work include:

- *Toward Unified World Models:* This work lays a foundation for the development of unified, compositional world models for robotics—models that can seamlessly generalize across a wide variety of deformable materials and their complex hybrid interactions. By embedding physics-based priors, such as energy conservation principles, accurate contact dynamics, and explicit material parameters, into structured graph representations, it becomes possible for a single model architecture to efficiently represent diverse deformable object behaviors simply through adjustments in node-level properties such as stiffness, damping, and elasticity. High-fidelity FEM simulations, as described in Chapter 4, provide a practical means for generating vast amounts of realistic synthetic trajectories, systematically covering the complete range of contact scenarios, deformation modes, and inertial responses. This synthetic data can pre-train a Graph Neural Dynamics backbone (Chapter 5), enabling it to internalize correct

inductive biases before being fine-tuned with limited amounts of real-world sensor data. Moreover, generative modeling frameworks grounded in these physics-based priors offer an additional powerful avenue: they can rapidly synthesize physically plausible scenarios or anticipate unseen object behaviors and interactions. By conditioning generative models on physically meaningful latent parameters, such as material stiffness or friction coefficients, robots can efficiently explore hypothetical scenarios, enabling anticipatory control and informed decision-making in dynamic environments. Crucially, the integration of an online parameter estimation mechanism allows the model to continually refine these embedded material properties during operation, facilitating rapid adaptation to novel objects and evolving task conditions without requiring expensive retraining. Collectively, these strategies offer a robust, scalable pathway toward flexible, accurate, and adaptive robotic manipulation of deformable and hybrid objects.

- *Learning Beyond Human Demonstrations:* In industrial and real-world applications, the ultimate potential of robotic manipulation lies not merely in replicating human skills but in surpassing human limitations in speed, precision, and reliability. The approaches introduced in this dissertation provide stepping stones toward frameworks that not only capture essential human expertise through demonstration but also leverage structured physical priors and constraints to autonomously optimize performance beyond what humans typically achieve. Specifically, the inverse reinforcement learning-based sequencing method (Chapter 6) offers an initial pathway for encoding and interpreting human preferences and rationales, enabling robots to generalize sequencing behavior effectively across diverse tasks and geometries. By integrating these learned strategies with accurate forward models of the object being manipulated (Chapter 3),

one can improve on this initial policy to better suit the robot's embodiment so that we can achieve optimal performance and generalize across variations in objects and operating conditions.

- *Learning Safe and Resilient Manipulation:* In complex manipulation tasks involving deformable and rigid objects, encountering unforeseen scenarios is inevitable. Therefore, robotic systems must autonomously recognize (as detailed in Chapter 10), diagnose, and recover from failures to ensure reliability and resilience (Chapter 8). This dissertation demonstrates that physics-informed dynamics modeling frameworks (Chapters 4 and 5), coupled with task policies learned through inverse reinforcement learning and expert demonstrations (Chapter 6), offer powerful representations for anticipating and managing manipulation failures. Failures such as grip loss due to excessive deformation or misalignment at contact interfaces are fundamentally tied to the physical state of the system. For instance, suction-based failures predominantly arise from deformation of the object at the suction–object interface. By utilizing the latent space of dynamics models, which captures deformation patterns, force distributions, and is grounded on material properties, robots can reliably predict imminent failures. Furthermore, generative models incorporating physics-based priors can systematically synthesize diverse failure scenarios, enriching the training dataset with realistic, challenging cases. Policies trained in this manner can proactively anticipate and mitigate potential failures, rather than merely react to them. Integrating latent-space predictions with physics-informed generative modeling thus enables robotic systems to autonomously detect early indicators of failure and execute timely recovery actions. This unified approach provides a robust framework for continual policy refinement, significantly enhancing the

safety, reliability, and effectiveness of robotic manipulation in dynamic, real-world environments.

- *Expanding to a Broader Class of Deformable Objects:* While this dissertation addressed several complex and industrially relevant classes of deformable objects, many deformable categories remain unexplored. Future work could extend these methods to include irregular-shaped sheets (e.g., automotive seat covers), materials with anisotropic and viscoelastic properties, multi-layered composites, and even biological tissues, each introducing new modeling, planning, and control challenges.

By addressing these future challenges, physics-informed learning could evolve into a cornerstone technology, enabling the next generation of intelligent robotic systems that operate safely, efficiently, and robustly in the dynamic, uncertain worlds that define real industrial and everyday environments.

References

1. Chen, Y.-W., Joseph, R. J., Kanyuck, A., Khan, S., Malhan, R. K., Manyar, O. M., *et al.* A Digital Twin for Automated Layup of Prepreg Composite Sheets. *Journal of Manufacturing Science and Engineering* **144,** 041010. doi:10.1115/1.4052132 (2022).
2. Manyar, O. M., McNulty, Z., Nikolaidis, S. & Gupta, S. K. *Inverse reinforcement learning framework for transferring task sequencing policies from humans to robots in manufacturing applications* in *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), 849–856.
3. Manyar, O. M., Desai, J., Deogaonkar, N., Joesph, R. J., Malhan, R., McNulty, Z., *et al. A Simulation-Based Grasp Planner for Enabling Robotic Grasping during Composite Sheet Layup* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), 930–937. doi:10.1109/ICRA48506.2021.9560939.
4. Manyar, O. M., Narayan, S. V., Lengade, R. & Gupta, S. K. *Physics-Informed Learning to Enable Robotic Screw-Driving Under Hole Pose Uncertainties* in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2023), 2993–3000. doi:10.1109/IROS55552.2023.10342151.
5. Manyar, O. M., Ye, H., Sagare, M., Mayya, S., Wang, F. & Gupta, S. K. *Simulation-Assisted Learning for Efficient Bin-Packing of Deformable Packages in a Bimanual Robotic Cell* in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2024), 5211–5218.
6. Manyar, O. M., Cheng, J., Levine, R., Krishnan, V., Barbič, J. & Gupta, S. K. Physics informed synthetic image generation for deep learning-based detection of wrinkles and folds. *Journal of Computing and Information Science in Engineering* **23,** 030903 (2023).
7. Kroemer, O., Niekum, S. & Konidaris, G. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research* **22,** 1–82 (2021).
8. Mazumdar, S., Pichler, D., Benevento, M., Seneviratine, W., Liang, R. & Witten, E. *American Composites Manufacturing Association 2020 State of the Industry Report* 2020.

9. Fleischer, J., Teti, R., Lanza, G., Mativenga, P., Möhring, H.-C. & Caggiano, A. Composite materials parts manufacturing. *CIRP Annals - Manufacturing Technology* (2018).
10. P. Volino, N. M.-T. & Faure, F. Mechanical comparison of new composite materials for aerospace applications. *ACM Transactions on Graphics* (2009).
11. Sin, F. S., Schroeder, D. & Barbič, J. *Vega FEM Library.*
12. Aono, M., Breen, D. E. & Wozny, M. J. Fitting a woven-cloth model to a curved surface: mapping algorithms. *Computer-Aided Design* **26,** 278–292 (1994).
13. Wang, J., Paton, R. & Page, J. The draping of woven fabric preforms and prepregs for production of polymer composite components. *Composites Part A: Applied Science and Manufacturing* **30,** 757–765 (1999).
14. Aono, M. in *CG International'90* 95–115 (Springer, 1990).
15. Collier, J. R., Collier, B. J., O'Toole, G. & Sargand, S. Drape prediction by means of finite-element analysis. *Journal of the Textile Institute* **82,** 96–107 (1991).
16. Krogh, C., Glud, J. A. & Jakobsen, J. Modeling the robotic manipulation of woven carbon fiber prepreg plies onto double curved molds: A path-dependent problem. *Journal of Composite Materials* **53,** 2149–2164. doi:10.1177/0021998318822722 (2019).
17. Volino, P., Magnenat-Thalmann, N. & Faure, F. A simple approach to nonlinear tensile stiffness for accurate cloth simulation (2009).
18. Tamstorf, R. & Grinspun, E. Discrete bending forces and their Jacobians. *Graphical models* **75,** 362–370 (2013).
19. Ng, H. N. & Grimsdale, R. L. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications* **16,** 28–41 (1996).
20. Elkington, M., Ward, C. & Sarkytbayev, A. *Automated composite draping: a review* in *SAMPE 2017* (SAMPE North America, 2017).
21. Sharma, S. B. & Sutcliffe, M. P. F. Draping of woven fabrics: Progressive drape model. *Plastics, Rubber and Composites* **32,** 57–64. doi:10.1179/146580103225009149 (2003).
22. Hancock, S. G. & Potter, K. D. The use of kinematic drape modelling to inform the hand lay-up of complex composite components using woven reinforcements. *Composites Part A: Applied Science and Manufacturing* **37,** 413–422. doi:http://dx.doi.org/10.1016/j.compositesa.2005.05.044 (2006).
23. Breen, D. E., House, D. H. & Getto, P. H. A physically-based particle model of woven cloth. *The Visual Computer* **8,** 264–277 (1992).
24. Breen, D. E., House, D. H. & Wozny, M. J. A particle-based model for simulating the draping behavior of woven cloth. *Textile Research Journal* **64,** 663–685 (1994).
25. Provot, X. *et al. Deformation constraints in a mass-spring model to describe rigid cloth behaviour* in *Graphics interface* (1995), 147–147.

26. House, D. & Breen, D. *Cloth modeling and animation* (CRC Press, 2000).
27. Kawabatra, S. The Standardization and Analysis of Hand Evaluation. *The Textile Machinery Society Japan* (1980).
28. Baraff, D. & Witkin, A. P. Large Steps in Cloth Simulation. *In Proc. of ACM SIGGRAPH 98,* 43–54 (1998).
29. Sin, F. S., Schroeder, D. & Barbič, J. Vega: Nonlinear FEM Deformable Object Simulator. *Computer Graphics* **32,** 38–50 (2013).
30. Johnson, S. G. *The NLopt nonlinear-optimization package.*
31. Runarsson, T. P. & Yao, X. Search biases in constrained evolutionary optimization. *IEEE Trans. on Systems, Man, and Cybernetics Part C: Applications and Reviews* **35,** 233–243 (2005).
32. Da, F. & Cohen-Steiner, D. *Advancing Front Surface Reconstruction.* 5.0.2 (CGAL Editorial Board, 2020).
33. Digne, J., Morel, J. M., Souzani, C. M. & Lartigue, C. Scale space meshing of raw data point sets. *Computer Graphics Forum* **30,** 1630–1642 (2011).
34. Macklin, M. *Warp: A High-performance Python Framework for GPU Simulation and Graphics* `https://github.com/nvidia/warp`. NVIDIA GPU Technology Conference (GTC). 2022.
35. Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J. & Durand, F. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* **38,** 201 (2019).
36. Liu, F., Su, E., Lu, J., Li, M. & Yip, M. C. Robotic Manipulation of Deformable Rope-Like Objects Using Differentiable Compliant Position-Based Dynamics. *IEEE Robotics and Automation Letters* **8,** 3964–3971. doi:`10.1109/LRA.2023.3264766` (2023).
37. Liang, J., Lin, M. & Koltun, V. *Differentiable Cloth Simulation for Inverse Problems* in *Advances in Neural Information Processing Systems* **32** (Curran Associates, Inc., 2019).
38. Li, Y., Du, T., Wu, K., Xu, J. & Matusik, W. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. *ACM Trans. Graph.* **42.** doi:`10.1145/3527660` (2022).
39. Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B., *et al. PlasticineLab: A Soft-Body Manipulation Benchmark with Differentiable Physics* in *International Conference on Learning Representations* (2021).
40. Heiden, E., Macklin, M., Narang, Y. S., Fox, D., Garg, A. & Ramos, F. *DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting* in *Proceedings of Robotics: Science and Systems* (Virtual, 2021). doi:`10.15607/RSS.2021.XVII.067`.
41. Lin, X., Wang, Y., Olkin, J. & Held, D. *SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation* in *Conference on Robot Learning* (2020).

42. Chen, S., Xu, Y., Yu, C., Li, L., Ma, X., Xu, Z., *et al. DaxBench: Benchmarking Deformable Object Manipulation with Differentiable Physics* in *The Eleventh International Conference on Learning Representations* (2023).
43. Liu, M., Yang, G., Luo, S. & Shao, L. *SoftMAC: Differentiable Soft Body Simulation with Forecast-based Contact Model and Two-way Coupling with Articulated Rigid Bodies and Clothes* in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2024), 12008–12015. doi:10.1109/IROS58592.2024.10801308.
44. Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., *et al.* A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* **37,** 1–14 (2018).
45. Macklin, M., Müller, M. & Chentanez, N. *XPBD: position-based simulation of compliant constrained dynamics* in *Proceedings of the 9th International Conference on Motion in Games* (Association for Computing Machinery, Burlingame, California, 2016), 49–54. doi:10.1145/2994258.2994272.
46. Murthy, J. K., Macklin, M., Golemo, F., Voleti, V., Petrini, L., Weiss, M., *et al. gradSim: Differentiable simulation for system identification and visuomotor control* in *International Conference on Learning Representations* (2021).
47. Qiao, Y.-L., Liang, J., Koltun, V. & Lin, M. *Scalable Differentiable Physics for Learning and Control* in *Proceedings of the 37th International Conference on Machine Learning* **119** (PMLR, 2020), 7847–7856.
48. Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., *et al. DiffTaichi: Differentiable Programming for Physical Simulation* in *International Conference on Learning Representations* (2020).
49. Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I. & Bachem, O. *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation* in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)* (2021).
50. Sundaresan, P., Antonova, R. & Bohgl, J. *DiffCloud: Real-to-Sim from Point Clouds with Differentiable Simulation and Rendering of Deformable Objects* in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2022), 10828–10835. doi:10.1109/IROS47612.2022.9981101.
51. Blanco-Mulero, D., Barbany, O., Alcan, G., Colomé, A., Torras, C. & Kyrki, V. Benchmarking the Sim-to-Real Gap in Cloth Manipulation. *IEEE Robotics and Automation Letters* **9,** 2981–2988. doi:10.1109/LRA.2024.3360814 (2024).
52. Yan, W., Vangipuram, A., Abbeel, P. & Pinto, L. *Learning predictive representations for deformable objects using contrastive estimation* in *Conference on Robot Learning* (2021), 564–574.

53. Seita, D., Florence, P., Tompson, J., Coumans, E., Sindhwani, V., Goldberg, K., *et al. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), 4568–4575. doi:10.1109/ICRA48506.2021.9561391.
54. Chen, S., Liu, Y., Yao, S. W., Li, J., Fan, T. & Pan, J. DiffSRL: Learning Dynamical State Representation for Deformable Object Manipulation With Differentiable Simulation. *IEEE Robotics and Automation Letters* **7,** 9533–9540. doi:10.1109/LRA.2022.3192209 (2022).
55. Qi, C., Lin, X. & Held, D. Learning Closed-Loop Dough Manipulation Using a Differentiable Reset Module. *IEEE Robotics and Automation Letters* **7,** 9857–9864. doi:10.1109/LRA.2022.3191239 (2022).
56. Shi, H., Xu, H., Huang, Z., Li, Y. & Wu, J. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks. *The International Journal of Robotics Research* **43,** 533–549. doi:10.1177/02783649231219020 (2024).
57. Kuroki, S., Guo, J., Matsushima, T., Okubo, T., Kobayashi, M., Ikeda, Y., *et al. GenDOM: Generalizable One-shot Deformable Object Manipulation with Parameter-Aware Policy* in *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024), 14792–14799. doi:10.1109/ICRA57147.2024.10611378.
58. Zhang, Y., Liu, F., Liang, X. & Yip, M. *Achieving Autonomous Cloth Manipulation with Optimal Control via Differentiable Physics-Aware Regularization and Safety Constraints* in *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024), 9931–9938. doi:10.1109/ICRA57147.2024.10611111.
59. Shukla, R., Yu, Z., Moode, S., Manyar, O. M., Wang, F., Mayya, S., *et al. Performing Efficient and Safe Deformable Package Transport Operations Using Suction Cups* in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2024), 11835–11842. doi:10.1109/IROS58592.2024.10802323.
60. Bridson, R., Fedkiw, R. & Anderson, J. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (TOG)* **21,** 594–603 (2002).
61. Smith, B., Goes, F. D. & Kim, T. Stable Neo-Hookean Flesh Simulation. *ACM Transactions on Graphics (TOG)* **37.** doi:10.1145/3180491 (2018).
62. Deepmind. *MuJoCo XLA (MJX)* https://mujoco.readthedocs.io/en/stable/mjx.html. [Accessed 03-03-2025]. 2023.
63. Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H. & Deng, S.-H. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *Journal of Electronic Science and Technology* **17,** 26–40. doi:https://doi.org/10.11989/JEST.1674-862X.80904120 (2019).

64. Srinivasan, K., Heiden, E., Ng, I., Bohg, J. & Garg, A. *DexMOTS: Learning Contact-Rich Dexterous Manipulation in an Object-Centric Task Space with Differentiable Simulation* in *International Symposium of Robotics Research (ISRR)* (2024).
65. Makoviichuk, D. & Makoviychuk, V. *rl-games: A High-performance Framework for Reinforcement Learning* `https://github.com/Denys88/rl_games`. 2021.
66. Nogueira, F. *Bayesian Optimization: Open source constrained global optimization tool for Python* 2014.
67. Wu, T., Pan, L., Zhang, J., Wang, T., Liu, Z. & Lin, D. *Density-aware chamfer distance as a comprehensive metric for point cloud completion* in *Proceedings of the 35th International Conference on Neural Information Processing Systems* (2021), 29088–29100.
68. Zhang, K., Li, B., Hauser, K. & Li, Y. *AdaptiGraph: Material-Adaptive Graph-Based Neural Dynamics for Robotic Manipulation* in *Proceedings of Robotics: Science and Systems* (Delft, Netherlands, 2024). doi:`10.15607/RSS.2024.XX.010`.
69. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE transactions on neural networks* **20,** 61–80 (2008).
70. Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B. & Torralba, A. *Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids* in *International Conference on Learning Representations* (2019).
71. Liu, Z., Zhou, G., He, J., Marcucci, T., Li, F.-F., Wu, J., *et al.* Model-based control with sparse neural dynamics. *Advances in Neural Information Processing Systems* **36,** 6280–6296 (2023).
72. Shi, H., Xu, H., Clarke, S., Li, Y. & Wu, J. *RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools* in *Conference on Robot Learning* (2023), 642–660.
73. Shi, H., Xu, H., Huang, Z., Li, Y. & Wu, J. Robocraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *The International Journal of Robotics Research* **43,** 533–549 (2024).
74. Lin, X., Huang, Z., Li, Y., Tenenbaum, J. B., Held, D. & Gan, C. *DiffSkill: Skill Abstraction from Differentiable Physics for Deformable Object Manipulations with Tools* in *International Conference on Learning Representations (ICLR)* (2022).
75. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. & Yu, P. S. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* **32,** 4–24 (2020).
76. Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., *et al.* Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).
77. Wellner, P., Reyes, V., Ashton, H. & Moutray, C. *Creating pathways for tomorrow's workforce today. Beyond reskilling in manufacturing* `https:`

//www2.deloitte.com/us/en/insights/industry/manufacturing/manufacturing-industry-diversity.html (2021).
78. Manyar, O. M., Cheng, J., Levine, R., Krishnan, V., Barbic, J. & Gupta, S. K. *A Synthetic Image Assisted Deep Learning Framework for Detecting Defects during Composite Sheet Layup* in *42nd Computers and Information in Engineering Conference (CIE)* (2022).
79. Malhan, R. K., Jomy Joseph, R., Shembekar, A. V., Kabir, A. M., Bhatt, P. M. & Gupta, S. K. *Online Grasp Plan Refinement for Reducing Defects During Robotic Layup of Composite Prepreg Sheets* in *IEEE International Conference on Robotics and Automation (ICRA)* (Paris, France, 2020).
80. Atkeson, C. G. & Schaal, S. *Robot Learning From Demonstration* in *ICML* (1997).
81. Arora, S. & Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence* **297,** 103500 (2021).
82. Ziebart, B. D., Maas, A., Bagnell, J. A. & Dey, A. K. *Maximum Entropy Inverse Reinforcement Learning* in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3* (AAAI Press, Chicago, Illinois, 2008), 1433–1438.
83. Ramachandran, D. & Amir, E. *Bayesian Inverse Reinforcement Learning* in *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (Morgan Kaufmann Publishers Inc., Hyderabad, India, 2007), 2586–2591.
84. Levine, S., Popovic, Z. & Koltun, V. *Nonlinear Inverse Reinforcement Learning with Gaussian Processes* in *Advances in Neural Information Processing Systems* (eds Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F. & Weinberger, K.) **24** (Curran Associates, Inc., 2011).
85. Ratliff, N. D., Bagnell, J. A. & Zinkevich, M. A. *Maximum Margin Planning* in *Proceedings of the 23rd International Conference on Machine Learning* (Association for Computing Machinery, Pittsburgh, Pennsylvania, USA, 2006), 729–736. doi:10.1145/1143844.1143936.
86. Bagnell, J., Chestnutt, J., Bradley, D. & Ratliff, N. *Boosting Structured Prediction for Imitation Learning* in *Advances in Neural Information Processing Systems* (eds Schölkopf, B., Platt, J. & Hoffman, T.) **19** (MIT Press, 2006).
87. Sadigh, D., Dragan, A. D., Sastry, S. S. & Seshia, S. A. *Active Preference-Based Learning of Reward Functions* in *Robotics: Science and Systems* (2017).
88. Basu, C., Biyik, E., He, Z., Singhal, M. & Sadigh, D. *Active Learning of Reward Dynamics from Hierarchical Queries* in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019). doi:10.1109/IROS40897.2019.8968522.
89. Biyik, E., Losey, D. P., Palan, M., Landolfi, N. C., Shevchuk, G. & Sadigh, D. Learning Reward Functions from Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences.

The International Journal of Robotics Research (IJRR). doi:10.1177/02783649211041652 (2021).

90. Nikolaidis, S., Ramakrishnan, R., Gu, K. & Shah, J. *Efficient model learning from joint-action demonstrations for human-robot collaborative tasks* in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2015).
91. Nemlekar, H., Modi, J., Gupta, S. K. & Nikolaidis, S. *Two-Stage Clustering of Human Preferences for Action Prediction in Assembly Tasks* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), 3487–3494. doi:10.1109/ICRA48506.2021.9561649.
92. Nemlekar, H., Guan, R., Luo, G., Gupta, S. K. & Nikolaidis, S. *Towards Transferring Human Preferences from Canonical to Actual Assembly Tasks* doi:10.48550/ARXIV.2111.06454.
93. Wu, Z., Lian, W., Unhelkar, V., Tomizuka, M. & Schaal, S. *Learning Dense Rewards for Contact-Rich Manipulation Tasks* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), 6214–6221. doi:10.1109/ICRA48506.2021.9561891.
94. Zhang, X., Sun, L., Kuang, Z. & Tomizuka, M. Learning Variable Impedance Control via Inverse Reinforcement Learning for Force-Related Tasks . *IEEE Robotics and Automation Letters* **6,** 2225–2232. doi:10.1109/LRA.2021.3061374 (2021).
95. Sugisawa, Y., Takasugi, K. & Asakawa, N. Machining sequence learning via inverse reinforcement learning. *Precision Engineering* **73,** 477–487. doi:https://doi.org/10.1016/j.precisioneng.2021.09.017 (2022).
96. Peternel, L., Petrič, T. & Babič, J. *Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface* in *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), 1497–1502. doi:10.1109/ICRA.2015.7139387.
97. Ng, C. W. X., Chan, K. H. K., Teo, W. K. & Chen, I.-M. *A method for capturing the tacit knowledge in the surface finishing skill by demonstration for programming a robot* in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), 1374–1379. doi:10.1109/ICRA.2014.6907031.
98. Ng, W. X., Chan, H. K., Teo, W. K. & Chen, I.-M. Programming a Robot for Conformance Grinding of Complex Shapes by Capturing the Tacit Knowledge of a Skilled Operator. *IEEE Transactions on Automation Science and Engineering* **14,** 1020–1030. doi:10.1109/TASE.2015.2474708 (2017).
99. Hu, J., Kabir, A. M., Hartford, S. M., Gupta, S. K. & Pagilla, P. R. *Robotic deburring and chamfering of complex geometries in high-mix/low-volume production applications* in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)* (2020), 1155–1160. doi:10.1109/CASE48305.2020.9217042.

100. Abbeel, P. & Ng, A. Y. *Apprenticeship Learning via Inverse Reinforcement Learning* in *Proceedings of the Twenty-First International Conference on Machine Learning* (Association for Computing Machinery, Banff, Alberta, Canada, 2004), 1. doi:10.1145/1015330.1015430.
101. Myers, R. H. & Montgomery, D. C. *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments* 1st (John Wiley & Sons, Inc., USA, 1995).
102. Taskar, B., Chatalbashev, V., Koller, D. & Guestrin, C. *Learning Structured Prediction Models: A Large Margin Approach* in *Proceedings of the 22nd International Conference on Machine Learning* (Association for Computing Machinery, Bonn, Germany, 2005), 896–903. doi:10.1145/1102351.1102464.
103. Bertsekas, D. P. *Convex Optimization Algorithms. The Science of Microfabrication* (Athena Scientific, Belmont, Massachusetts, 2016).
104. Amin, K., Jiang, N. & Singh, S. *Repeated Inverse Reinforcement Learning* in *Advances in Neural Information Processing Systems* (eds Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., *et al.*) **30** (Curran Associates, Inc., 2017).
105. Attene, M., Falcidieno, B. & Spagnuolo, M. Hierarchical Mesh Segmentation Based on Fitting Primitives. *Vis. Comput.* **22,** 181–193. doi:10.1007/s00371-006-0375-x (2006).
106. Malhan, R. K., Shembekar, A. V., Kabir, A. M., Bhatt, P. M., Shah, B., Zanio, S., *et al.* Automated planning for robotic layup of composite prepreg. *Robotics and Computer-Integrated Manufacturing* **67,** 102020.
107. Newell, G. & Khodabandehloo, K. Modelling flexible sheets for automatic handling and lay-up of composite components. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **209,** 423–432 (1995).
108. Krogh, C., Glud, J. A. & Jakobsen, J. Modeling the robotic manipulation of woven carbon fiber prepreg plies onto double curved molds: A path-dependent problem. *Journal of Composite Materials* **53,** 2149–2164 (2019).
109. Elkington, M., Bloom, D., Ward, C., Chatzimichali, A. & Potter, K. Hand layup: understanding the manual process. *Advanced Manufacturing: Polymer & Composites Science* **1,** 138–151 (2015).
110. Buckingham, R. O. & Newell, G. C. Automating the manufacture of composite broadgoods. *Composites Part A: Applied Science and Manufacturing* **27,** 191–200. doi:http://dx.doi.org/10.1016/1359-835X(96)80001-9 (1996).
111. Molfino, R., Zoppi, M., Cepolina, F., Yousef, J. & Cepolina, E. E. Design of a Hyper-flexible cell for handling 3D Carbon fiber fabric. *Recent advances in mechanical engineering and mechanics* **165** (2014).
112. Elkington, M., Ward, C. & Potter, K. D. *Automated layup of sheet prepregs on complex moulds* in *SAMPE Long Beach Conference* (2016).

113. Seliger, G., Szimmat, F., Niemeier, J. & Stephan, J. Automated handling of non-rigid parts. *CIRP Annals* **52,** 21–24 (2003).
114. Fantoni, G., Santochi, M., Dini, G., Tracht, K., Scholz-Reiter, B., Fleischer, J., *et al.* Grasping devices and methods in automated production processes. *CIRP Annals* **63,** 679–701 (2014).
115. Manyar, O. M., Kanyuck, A., Deshkulkarni, B. & Gupta, S. K. *Visual servo based trajectory planning for fast and accurate sheet pick and place operations* in *International Manufacturing Science and Engineering Conference* **85802** (2022), V001T04A019.
116. Björnsson, A., Jonsson, M. & Johansen, K. *Automated material handling in composite manufacturing using pick-and-place systems – a review* 2018.
117. Elkington., M., Ward, C. & Sarkytbayev, A. *Automated composite draping: a review* in *SAMPE* (SAMPE North America, 2017).
118. Peng, Z. & Yuanchun, L. *Position/force control of two manipulators handling a flexible payload based on finite-element model* in *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (2007), 2178–2182. doi:10.1109/ROBIO.2007.4522507.
119. Zhang, P. & c. Li, Y. *Simulations and Trajectory Tracking of Two Manipulators Manipulating a Flexible Payload* in *IEEE Conference on Robotics, Automation and Mechatronics* (2008), 72–77. doi:10.1109/RAMECH.2008.4681323.
120. Tzeranis, D., Ishijima, Y. & Dubowsky, S. Manipulation of Large Flexible Structural Modules By Space Robots Mounted on Flexible Structures. *Proc. Int. Sym. on Artificial Intelligence, Robotics and Automation in Space* (2005).
121. Das, J. & Sarkar, N. Autonomous Shape Control of a Deformable Object by Multiple Manipulators. *Journal of Intelligent & Robotic Systems* **62,** 3–27. doi:10.1007/s10846-010-9436-5 (2011).
122. Henrich, D. & Wörn, H. *Robot Manipulation of Deformable Objects* (Springer Science & Business Media, 2012).
123. Ding, F., Huang, J., Wang, Y., Matsuno, T. & Fukuda, T. Vibration damping in manipulation of deformable linear objects using sliding mode control. *Advanced Robotics* **28,** 157–172. doi:10.1080/01691864.2013.861769 (2014).
124. Alonso-Mora, J., Knepper, R., Siegwart, R. & Rus, D. *Local motion planning for collaborative multi-robot manipulation of deformable objects* in *IEEE International Conference on Robotics and Automation (ICRA)* (2015), 5495–5502.
125. Kruse, D., Radke, R. J. & Wen, J. T. *Human-robot collaborative handling of highly deformable materials* in *American Control Conference (ACC)* (2017), 1511–1516. doi:10.23919/ACC.2017.7963167.
126. Tsarouchi, P., Spiliotopoulos, J., Michalos, G., Koukas, S., Athanasatos, A., Makris, S., *et al.* A decision making framework for human robot collaborative workplace generation. *Procedia CIRP* **44,** 228–232 (2016).

127. Gunnarsson, G. G., Nielsen, O. W., Schlette, C. & Petersen, H. G. *Fast and simple interacting models of drape tool and ply material for handling free hanging, pre-impregnated carbon fibre material* in *International Conference on Informatics in Control, Automation and Robotics* (2018), 1–25.
128. Khalil, F. F. & Payeur, P. in *Robot Manipulators Trends and Development* (InTech, 2010).
129. Jiménez, P. Survey on model-based manipulation planning of deformable objects. *Robotics and computer-integrated manufacturing* **28,** 154–163 (2012).
130. Moll, M. & Kavraki, L. E. Path planning for deformable linear objects. *IEEE Transactions on Robotics* **22,** 625–636 (2006).
131. Saha, M. & Isto, P. Manipulation Planning for Deformable Linear Objects. *IEEE Transactions on Robotics* **23,** 1141–1150. doi:10.1109/TRO.2007.907486 (2007).
132. Roussel, O., Borum, A., Taix, M. & Bretl, T. *Manipulation planning with contacts for an extensible elastic rod by sampling on the submanifold of static equilibrium configurations* in *IEEE International Conference on Robotics and Automation (ICRA)* (2015), 3116–3121.
133. Papacharalampopoulos, A., Makris, S., Bitzios, A. & Chryssolouris, G. Prediction of cabling shape during robotic manipulation. *The International Journal of Advanced Manufacturing Technology* **82,** 123–132 (2016).
134. Saha, M. & Isto, P. *Motion planning for robotic manipulation of deformable linear objects* in *IEEE International Conference on Robotics and Automation* (2006), 2478–2484. doi:10.1109/ROBOT.2006.1642074.
135. Wakamatsu, H., Arai, E. & Hirai, S. Knotting/Unknotting Manipulation of Deformable Linear Objects. *The International Journal of Robotics Research* **25,** 371–395. doi:10.1177/0278364906064819 (2006).
136. Ladd, A. M. & Kavraki, L. E. Using Motion Planning for Knot Untangling. *The International Journal of Robotics Research* **23,** 797–808. doi:10.1177/0278364904045469 (2004).
137. Matsuno, T., Fukuda, T. & Arai, F. *Flexible rope manipulation by dual manipulator system using vision sensor* in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No.01TH8556)* **2** (2001), 677–682 vol.2. doi:10.1109/AIM.2001.936748.
138. Berenson, D. *Manipulation of deformable objects without modeling and simulating deformation* in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013), 4525–4532. doi:10.1109/IROS.2013.6697007.
139. Li, Y., Yue, Y., Xu, D., Grinspun, E. & Allen, P. K. *Folding deformable objects using predictive simulation and trajectory optimization* in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), 6000–6006.

140. Doumanoglou, A., Stria, J., Peleka, G., Mariolis, I., Petrík, V., Kargakos, A., *et al.* Folding Clothes Autonomously: A Complete Pipeline. *IEEE Transactions on Robotics* **32,** 1461–1478. doi:10.1109/TRO.2016.2602376 (2016).
141. Maitin-Shepard, J., Cusumano-Towner, M., Lei, J. & Abbeel, P. *Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding* in *IEEE International Conference on Robotics and Automation (ICRA)* (2010), 2308–2315.
142. Van Den Berg, J., Miller, S., Goldberg, K. & Abbeel, P. in *Algorithmic Foundations of Robotics IX* 409–424 (Springer, 2010).
143. "Koustoumpardis, P. N., Chatzilygeroudis, K. I., Synodinos, A. I. & Aspragathos, N. A. *"Human Robot Collaboration for Folding Fabrics Based on Force/RGB-D Feedback"* in *"Advances in Robot Design and Intelligent Control"* ("Springer International Publishing", 2016), "235–243".
144. Miller, S., Fritz, M., Darrell, T. & Abbeel, P. *Parametrized shape models for clothing* in *IEEE International Conference on Robotics and Automation (ICRA)* (2011), 4861–4868.
145. Miller, S., van den Berg, J., Fritz, M., Darrell, T., Goldberg, K. & Abbeel, P. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research* **31,** 249–267. doi:10.1177/0278364911430417 (2012).
146. Bai, Y., Yu, W. & Liu, C. K. Dexterous manipulation of cloth. *Computer Graphics Forum* **35,** 523–532 (2016).
147. Schulman, J., Ho, J., Lee, C. & Abbeel, P. in *Robotics Research: The 16th International Symposium ISRR* 339–354 (Springer International Publishing, Cham, 2016). doi:10.1007/978-3-319-28872-7_20.
148. McConachie, D. & Berenson, D. Bandit-based model selection for deformable object manipulation. *arXiv preprint arXiv:1703.10254* (2017).
149. McConachie, D., Ruan, M. & Berenson, D. *Interleaving planning and control for deformable object manipulation* in *International Symposium on Robotics Research (ISRR)* (2017).
150. Koganti, N., Tamei, T., Ikeda, K. & Shibata, T. Bayesian Nonparametric Learning of Cloth Models for Real-Time State Estimation. *IEEE Transactions on Robotics* **33,** 916–931. doi:10.1109/TRO.2017.2691721 (2017).
151. Flixeder, S., Glück, T. & Kugi, A. Modeling and force control for the collaborative manipulation of deformable strip-like materials. *IFAC-PapersOnLine* **49,** 95–102 (2016).
152. Kruse, D., Radke, R. J. & Wen, J. T. *Collaborative human-robot manipulation of highly deformable materials* in *IEEE International Conference on Robotics and Automation (ICRA)* (2015), 3782–3787. doi:10.1109/ICRA.2015.7139725.

153. Hu, Z., Sun, P. & Pan, J. Three-dimensional deformable object manipulation using fast online gaussian process regression. *Robotics and Automation Letters* **3,** 979–986 (2018).
154. Yang, P.-C., Sasaki, K., Suzuki, K., Kase, K., Sugano, S. & Ogata, T. Repeatable folding task by humanoid robot worker using deep learning. *Robotics and Automation Letters* **2,** 397–403 (2017).
155. Zienkiewicz, O. C. *The Finite Element Method* (McGraw-Hill Book Company (UK) Limited, Maidenhead, Berkshire, England, 1977).
156. Wang, H., O'Brien, J. F. & Ramamoorthi, R. Data-driven elastic models for cloth: modeling and measurement. *ACM Transactions on Graphics (SIGGRAPH 2011)* **30,** 71:1–71:12 (2011).
157. Lee, A. X., Huang, S. H., Hadfield-Menell, D., Tzeng, E. & Abbeel, P. *Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects* in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), 4402–4407. doi:10.1109/IROS.2014.6943185.
158. Lee, A. X., Lu, H., Gupta, A., Levine, S. & Abbeel, P. *Learning force-based manipulation of deformable objects from multiple demonstrations* in *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), 177–184. doi:10.1109/ICRA.2015.7138997.
159. Huang, S. H., Pan, J., Mulcaire, G. & Abbeel, P. *Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects* in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), 878–885. doi:10.1109/IROS.2015.7353475.
160. Malhan, R. K., Kabir, A. M., Shah, B., Centea, T. & Gupta, S. K. Hybrid cells for multi-layer prepreg composite sheet layup. *IEEE International Conference on Automation Science and Engineering (CASE). Munich, Germany.* (2018).
161. Malhan, R., Kabir, A., Shah, B., Centea, T. & Gupta, S. *Automated prepreg sheet placement using collaborative robotics* in *North America Society for the Advancement of Material and Process Engineering (SAMPE) Long beach conference* (2018).
162. Yoshikawa, T. Manipulability of Robotic Mechanisms. *The International Journal of Robotics Research* **4,** 3–9. doi:10.1177/027836498500400201 (1985).
163. Pan, J., Chitta, S. & Manocha, D. *FCL: A general purpose library for collision and proximity queries* in *2012 IEEE International Conference on Robotics and Automation* (2012), 3859–3866. doi:10.1109/ICRA.2012.6225337.
164. Chung, M. K., Lee, I. & Yeo, Y. S. Physiological workload evaluation of screw driving tasks in automobile assembly jobs. *International Journal of Industrial Ergonomics* **28.** 5th Pan-Pacific Conference on Occupational Ergonomics, 181–188. doi:https://doi.org/10.1016/S0169-8141(01)00031-2 (2001).

165. Zhang, D., Leng, J., Xie, M., Yan, H. & Liu, Q. Digital twin enabled optimal reconfiguration of the semi-automatic electronic assembly line with frequent changeovers. *Robotics and Computer-Integrated Manufacturing* **77,** 102343. doi:https://doi.org/10.1016/j.rcim.2022.102343 (2022).
166. Jia, Z., Bhatia, A., Aronson, R. M., Bourne, D. & Mason, M. T. A Survey of Automated Threaded Fastening. *IEEE Transactions on Automation Science and Engineering* **16,** 298–310. doi:10.1109/TASE.2018.2835382 (2019).
167. Lara, B., Althoefer, K. & Seneviratne, L. D. *Automated robot-based screw insertion system* in *IECON'98. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 98CH36200)* **4** (1998), 2440–2445.
168. Warnecke, H., Abele, E., Walther, J. & Fischer, G. Investigations of the Screw Driving Process with Sensor-Controlled Industrial Robots. *CIRP Annals* **34,** 41–44. doi:https://doi.org/10.1016/S0007-8506(07)61719-3 (1985).
169. Nicolson, E. & Fearing, R. *Dynamic modeling of a part mating problem: threaded fastener insertion* in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91* (1991), 30–37 vol.1. doi:10.1109/IROS.1991.174422.
170. Han, S., Choi, M.-S., Shin, Y.-W., Jang, G.-R., Lee, D.-H., Cho, J., *et al.* Screwdriving Gripper That Mimics Human Two-Handed Assembly Tasks. *Robotics* **11.** doi:10.3390/robotics11010018 (2022).
171. Hu, Z., Wan, W., Koyama, K. & Harada, K. A Mechanical Screwing Tool for Parallel Grippers—Design, Optimization, and Manipulation Policies. *IEEE Transactions on Robotics* **38,** 1139–1159. doi:10.1109/TRO.2021.3091282 (2022).
172. Hwang, J.-Y., Jung, D.-H., Roh, Y.-J., Nam, K.-J. & Hwang, D.-Y. *Low-cost automatic screw machine using a commercial electric screwdriver* in *2012 12th International Conference on Control, Automation and Systems* (2012), 1055–1060.
173. Watson, J., Miller, A. & Correll, N. Autonomous industrial assembly using force, torque, and RGB-D sensing. *Advanced Robotics* **34,** 546–559. doi:10.1080/01691864.2020.1715254 (2020).
174. Pitipong, S., Pornjit, P. & Watcharin, P. *An automated four-DOF robot screw fastening using visual servo* in *2010 IEEE/SICE International Symposium on System Integration* (2010), 379–383. doi:10.1109/SII.2010.5708355.
175. Liu, Z., Xie, W., Xiao, M., Zhang, Z., Jin, X. & Qin, Y. Servo control technology for automatic screw-tightening process of robotic arm based on multi-vision sensor. *Journal of Physics: Conference Series* **2819,** 012031. doi:10.1088/1742-6596/2819/1/012031 (2024).
176. Claudia Carvalho, A., Isabel Carvalho, A. & Correia, R. *Robot With Vision Guidance System for Unscrewing Operation* in *2024 International*

Conference on Decision Aid Sciences and Applications (DASA) (2024), 1–5. doi:10.1109/DASA63652.2024.10836165.

177. Tang, L. & Jia, Y.-B. *Robotic Fastening with a Manual Screwdriver* in *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), 5269–5275. doi:10.1109/ICRA48891.2023.10161139.
178. Chen, C.-H. & Chong, W.-d. *Fastening torque control for robotic screw driver under uncertain environment* in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)* (2014), 814–818. doi:10.1109/ICCAS.2014.6987891.
179. Dharmara, K., Monfared, R. P., Ogun, P. S. & Jackson, M. R. Robotic assembly of threaded fasteners in a non-structured environment. *The International Journal of Advanced Manufacturing Technology* **98,** 2093–2107. doi:10.1007/s00170-018-2363-5 (2018).
180. Wang, X., Shi, L. & Katupitiya, J. Robust control of a dual-arm space robot for in-orbit screw-driving operation. *Acta Astronautica* **200,** 139–148. doi:https://doi.org/10.1016/j.actaastro.2022.07.048 (2022).
181. Tao, R., Jing, F., Hou, J., Xing, S., Fu, Y., Fan, J., *et al.* APTMRS: Autonomous Prism Target Maintenance Robotic System for FAST. *IEEE Transactions on Automation Science and Engineering,* 1–17. doi:10.1109/TASE.2024.3406772 (2024).
182. Nicolson, E. J. *Grasp stiffness solutions for threaded insertion* (University of California, Berkeley, 1990).
183. Nicolson, E. & Fearing, R. *Compliant control of threaded fastener insertion* in *[1993] Proceedings IEEE International Conference on Robotics and Automation* (1993), 484–490 vol.1. doi:10.1109/ROBOT.1993.292026.
184. Ang, M. & Andeen, G. Specifying and achieving passive compliance based on manipulator structure. *IEEE Transactions on Robotics and Automation* **11,** 504–515. doi:10.1109/70.406934 (1995).
185. Tang, L., Jia, Y.-B. & Xue, Y. *Robotic Manipulation of Hand Tools: The Case of Screwdriving* in *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024), 13883–13890. doi:10.1109/ICRA57147.2024.10610831.
186. Klingajay, M. & Giannoccaro, N. *Comparison between least square & Newton Raphson for estimation parameters of an autonomous threaded fastenings* in *IEEE International Conference on Industrial Technology, 2003* **1** (2003), 163–168 Vol.1. doi:10.1109/ICIT.2003.1290261.
187. Matsuno, T., Huang, J. & Fukuda, T. *Fault detection algorithm for external thread fastening by robotic manipulator using linear support vector machine classifier* in *2013 IEEE International Conference on Robotics and Automation* (2013), 3443–3450. doi:10.1109/ICRA.2013.6631058.
188. Wilson, W. C., Rogge, M. D., Fisher, B. H., Malocha, D. C. & Atkinson, G. M. Fastener Failure Detection Using a Surface Acoustic Wave Strain

Sensor. *IEEE Sensors Journal* **12,** 1993–2000. doi:10.1109/JSEN.2011.2181160 (2012).

189. Iosifidis, A. *Detecting Faults During Automatic Screwdriving: A Dataset and Use Case of Anomaly Detection for Automatic Screwdriving* in *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems: Proceedings of the 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021), Aalborg, Denmark, October/November 2021* (2021), 224.
190. Althoefer, K., Lara, B., Zweiri, Y. H. & Seneviratne, L. D. Automated failure classification for assembly with self-tapping threaded fastenings using artificial neural networks. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* **222,** 1081–1095. doi:10.1243/09544062jmes546 (2008).
191. Moreira, G. R., Lahr, G. J. G., Boaventura, T., Savazzi, J. O. & Caurin, G. A. P. *Online prediction of threading task failure using Convolutional Neural Networks* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 2056–2061. doi:10.1109/IROS.2018.8594501.
192. Vučićević, O., Li, C. & Zakeriharandi, M. *Time-Series Anomaly Detection for Industrial Screwdriving Task with Machine Learning Algorithms* MA thesis (Aalborg University, 2023).
193. Yang, Z., Yazdi, P. G. & Thiede, S. *Machine-learning-enabled Decision Support for Screwdriving Process* in *2024 IEEE 22nd International Conference on Industrial Informatics (INDIN)* (2024), 1–6. doi:10.1109/INDIN58382.2024.10774335.
194. Wende, M., Bender, M., Frye, M., Grunert, D. & Schmitt, R. H. ML-Pipeline for the Quality Assessment of Screwdriving Processes. *Procedia CIRP* **126.** 17th CIRP Conference on Intelligent Computation in Manufacturing Engineering (CIRP ICME '23), 951–956. doi:https://doi.org/10.1016/j.procir.2024.08.362 (2024).
195. Ribeiro, D., Matos, L. M., Moreira, G., Pilastri, A. & Cortez, P. Isolation forests and deep autoencoders for industrial screw tightening anomaly detection. *Computers* **11,** 54 (2022).
196. Ferhat, M., Ritou, M., Leray, P. & Le Du, N. Incremental discovery of new defects: application to screwing process monitoring. *CIRP Annals* **70,** 369–372. doi:https://doi.org/10.1016/j.cirp.2021.04.026 (2021).
197. Cheng, X., Jia, Z. & Mason, M. T. *Data-Efficient Process Monitoring and Failure Detection for Robust Robotic Screwdriving* in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)* (2019), 1705–1711. doi:10.1109/COASE.2019.8842854.

198. Shiue, Y.-R. Development of two-level decision tree-based real-time scheduling system under product mix variety environment . *Robotics and Computer-Integrated Manufacturing* **25,** 709–720. doi:https://doi.org/10.1016/j.rcim.2008.06.002 (2009).
199. Cheng, X., Jia, Z., Bhatia, A., Aronson, R. M. & Mason, M. T. *Sensor selection and stage & result classifications for automated miniature screwdriving* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 6078–6085.
200. Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J. & Battaglia, P. *Learning to simulate complex physics with graph networks* in *International conference on machine learning* (2020), 8459–8468.
201. Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., *et al.* Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems* **29** (2016).
202. Nagabandi, A., Konolige, K., Levine, S. & Kumar, V. *Deep Dynamics Models for Learning Dexterous Manipulation* in *Proceedings of the Conference on Robot Learning* **100** (PMLR, 2020), 1101–1112.
203. Lusch, B., Kutz, J. N. & Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications* **9,** 4950. doi:10.1038/s41467-018-07210-0 (2018).
204. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences* **113,** 3932–3937. doi:10.1073/pnas.1517384113 (2016).
205. Zhao, Z., Li, Y., Liu, C. & Liu, X. Predicting part deformation based on deformation force data using Physics-informed Latent Variable Model. *Robotics and Computer-Integrated Manufacturing* **72,** 102204. doi:https://doi.org/10.1016/j.rcim.2021.102204 (2021).
206. Raissi, M., Perdikaris, P. & Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378,** 686–707. doi:https://doi.org/10.1016/j.jcp.2018.10.045 (2019).
207. Djeumou, F., Neary, C., Goubault, E., Putot, S. & Topcu, U. *Neural Networks with Physics-Informed Architectures and Constraints for Dynamical Systems Modeling* in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference* **168** (PMLR, 2022), 263–277.
208. Kaiser, E., Kutz, J. N. & Brunton, S. L. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **474,** 20180335. doi:10.1098/rspa.2018.0335 (2018).
209. Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S. & Yang, L. Physics-informed machine learning. *Nature Reviews Physics* **3,** 422–440. doi:10.1038/s42254-021-00314-5 (2021).

210. Champion, K., Lusch, B., Kutz, J. N. & Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences* **116,** 22445–22451. doi:10.1073/pnas.1906995116 (2019).
211. Brunton, S. L. & Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* 2nd ed. doi:10.1017/9781009089517 (Cambridge University Press, 2022).
212. Giannoccaro, N. & Klingajay, M. *Identification of threaded fastening parameters based on least square method* in *SICE 2004 Annual Conference* **3** (2004), 2592–2597 vol. 3.
213. Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., Thakar, S., Yoon, Y. J., *et al.* Image-Based Surface Defect Detection Using Deep Learning: A Review. *Journal of Computing and Information Science in Engineering* **21.** 040801. doi:10.1115/1.4049535 (2021).
214. Wu, Y., Sicard, B. & Gadsden, S. A. Physics-informed machine learning: a comprehensive review on applications in anomaly detection and condition monitoring. *Expert Systems with Applications,* 124678 (2024).
215. Sucan, I. A., Moll, M. & Kavraki, L. E. The open motion planning library. *IEEE Robotics & Automation Magazine* **19,** 72–82 (2012).
216. Huber, M., Mower, C. E., Ourselin, S., Vercauteren, T. & Bergeles, C. LBR-Stack: ROS 2 and Python Integration of KUKA FRI for Med and IIWA Robots. *Journal of Open Source Software* **9,** 6138. doi:10.21105/joss.06138 (2024).
217. Macenski, S., Foote, T., Gerkey, B., Lalancette, C. & Woodall, W. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics* **7,** eabm6074 (2022).
218. Wang, J., Li, Y., Gao, R. X. & Zhang, F. Hybrid physics-based and data-driven models for smart manufacturing: Modelling, simulation, and explainability. *Journal of Manufacturing Systems* **63,** 381–391 (2022).
219. Boothroyd, G. *Assembly Automation and Product Design* doi:10.1201/9781420027358 (CRC Press, 2005).
220. Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
221. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).
222. Shome, R., Tang, W. N., Song, C., Mitash, C., Kourtev, H., Yu, J., *et al. Towards Robust Product Packing with a Minimalistic End-Effector* in *2019 International Conference on Robotics and Automation (ICRA)* (2019), 9007–9013. doi:10.1109/ICRA.2019.8793966.
223. Agarwal, M., Biswas, S., Sarkar, C., Paul, S. & Paul, H. S. Jampacker: An Efficient and Reliable Robotic Bin Packing System for Cuboid Objects. *IEEE Robotics and Automation Letters* **6,** 319–326. doi:10.1109/LRA.2020.3043168 (2021).

224. Yang, Z., Yang, S., Song, S., Zhang, W., Song, R., Cheng, J., *et al. PackerBot: Variable-Sized Product Packing with Heuristic Deep Reinforcement Learning* in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), 5002–5008. doi:10.1109/IROS51168.2021.9635914.
225. Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., *et al.* Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters* **5,** 6670–6677. doi:10.1109/LRA.2020.3013848 (2020).
226. Chen, H., Niu, Y., Hong, K., Liu, S., Wang, Y., Li, Y., *et al. Predicting Object Interactions with Behavior Primitives: An Application in Stowing Tasks* in *7th Annual Conference on Robot Learning* (2023).
227. Shen, B., Jiang, Z., Choy, C., Savarese, S., Guibas, L. J., Anandkumar, A., *et al.* Action-conditional implicit visual dynamics for deformable object manipulation. *The International Journal of Robotics Research* **43,** 437–455. doi:10.1177/02783649231191222 (2024).
228. Liu, F., Su, E., Lu, J., Li, M. & Yip, M. C. Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. *IEEE Robotics and Automation Letters* (2023).
229. Thakar, S., Kabir, A., Bhatt, P. M., Malhan, R. K., Rajendran, P., Shah, B. C., *et al. Task assignment and motion planning for bi-manual mobile manipulation* in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)* (2019), 910–915.
230. Avigal, Y., Berscheid, L., Asfour, T., Kröger, T. & Goldberg, K. *SpeedFolding: Learning Efficient Bimanual Folding of Garments* in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2022), 1–8. doi:10.1109/IROS47612.2022.9981402.
231. Chiu, Z.-Y., Richter, F., Funk, E. K., Orosco, R. K. & Yip, M. C. *Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), 7737–7743.
232. Chitnis, R., Tulsiani, S., Gupta, S. & Gupta, A. *Efficient bimanual manipulation using learned task schemas* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), 1149–1155.
233. Shi, J. & Koonjul, G. S. Real-time grasping planning for robotic bin-picking and kitting applications. *IEEE Transactions on Automation Science and Engineering* **14,** 809–819 (2017).
234. Kagerer, F., Beinhofer, M., Stricker, S. & Nüchter, A. BED-BPP: Benchmarking dataset for robotic bin packing problems. *The International Journal of Robotics Research* **42,** 1007–1014. doi:\allowbreak10.1177/02783649231193048\allowbreak (2023).
235. Weng, C.-Y., Yin, W., Lim, Z. J. & Chen, I.-M. *A Framework for Robotic Bin Packing with a Dual-Arm Configuration* in *Advances in Mechanism and Machine Science* (ed Uhl, T.) (Springer International Publishing, Cham, 2019), 2799–2808.

236. Wang, F. & Hauser, K. *Stable Bin Packing of Non-convex 3D Objects with a Robot Manipulator* in *2019 International Conference on Robotics and Automation (ICRA)* (2019), 8698–8704. doi:10.1109/ICRA.2019.8794049.
237. Yang, S., Song, S., Chu, S., Song, R., Cheng, J., Li, Y., *et al.* Heuristics Integrated Deep Reinforcement Learning for Online 3D Bin Packing. *IEEE Transactions on Automation Science and Engineering* **21,** 939–950. doi:10.1109/TASE.2023.3235742 (2024).
238. Zuo, Q., Liu, X., Xu, L., Xiao, L., Xu, C., Liu, J., *et al. The Three-dimensional Bin Packing Problem for Deformable Items* in *2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (2022), 0911–0918. doi:10.1109/IEEM55944.2022.9989600.
239. Ma, W., Zhang, B., Han, L., Huo, S., Wang, H. & Navarro-Alarcon, D. Action Planning for Packing Long Linear Elastic Objects into Compact Boxes with Bimanual Robotic Manipulation. *IEEE/ASME Transactions on Mechatronics* (2022).
240. Li, S., Keipour, A., Jamieson, K., Hudson, N., Swan, C. & Bekris, K. *Demonstrating Large-Scale Package Manipulation via Learned Metrics of Pick Success* in *Proceedings of Robotics: Science and Systems* (Daegu, Republic of Korea, 2023). doi:10.15607/RSS.2023.XIX.023.
241. Bahety, A., Jain, S., Ha, H., Hager, N., Burchfiel, B., Cousineau, E., *et al. Bag All You Need: Learning a Generalizable Bagging Strategy for Heterogeneous Objects* in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2023), 960–967. doi:10.1109/IROS55552.2023.10341841.
242. Chen, L. Y., Shi, B., Seita, D., Cheng, R., Kollar, T., Held, D., *et al. AutoBag: Learning to Open Plastic Bags and Insert Objects* in *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), 3918–3925. doi:10.1109/ICRA48891.2023.10161402.
243. Charles, R. Q., Su, H., Kaichun, M. & Guibas, L. J. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation* in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 77–85. doi:10.1109/CVPR.2017.16.
244. Heiden, E., Millard, D., Coumans, E., Sheng, Y. & Sukhatme, G. S. *NeuralSim: Augmenting Differentiable Simulators with Neural Networks* in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (2021), 9474–9481. doi:10.1109/ICRA48506.2021.9560935.
245. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. *A density-based algorithm for discovering clusters in large spatial databases with noise* in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (AAAI Press, Portland, Oregon, 1996), 226–231. doi:10.5555/3001460.3001507.
246. Araújo, A. M. C. & Oliveira, M. M. A robust statistics approach for plane detection in unorganized point clouds. *Pattern Recognition* **100,**

107115. doi:https://doi.org/10.1016/j.patcog.2019.107115 (2020).

247. Todorov, E., Erez, T. & Tassa, Y. *MuJoCo: A physics engine for model-based control* in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), 5026–5033. doi:10.1109/IROS.2012.6386109.

248. Jie, L., Siwei, L., Qingyong, L., Hanqing, Z. & Shengwei, R. *Real-time rail head surface defect detection: A geometrical approach* in *IEEE International Symposium on Industrial Electronics* (2009), 769–774. doi:10.1109/ISIE.2009.5214088.

249. Cui, Y., Jin, J. S., Luo, S., Park, M. & Au, S. S. *Automated Pattern Recognition and Defect Inspection System* in *Fifth International Conference on Image and Graphics* (2009). doi:10.1109/ICIG.2009.144.

250. Iivarinen, J. *Surface defect detection with histogram-based texture features* in *SPIE Optics East* (2000).

251. Xue-wu, Z., Yan-qiong, D., Yan-yun, L., Ai-ye, S. & Rui-yu, L. A Vision Inspection System for the Surface Defects of Strongly Reflected Metal Based on Multi-Class SVM. *Expert Syst. Appl.* **38,** 5930–5939. doi:10.1016/j.eswa.2010.11.030 (2011).

252. Masci, J., Meier, U., Ciresan, D., Schmidhuber, J. & Fricout, G. *Steel defect classification with Max-Pooling Convolutional Neural Networks* in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (2012), 1–6. doi:10.1109/IJCNN.2012.6252468.

253. Yun, J. P., Choi, S., Jeon, Y.-j., Choi, D.-c. & Kim, S. W. *Detection of line defects in steel billets using undecimated wavelet transform* in *2008 International Conference on Control, Automation and Systems* (2008), 1725–1728. doi:10.1109/ICCAS.2008.4694506.

254. Aziz, M. A., Haggag, A. S. & Sayed, M. S. *Fabric defect detection algorithm using morphological processing and DCT* in *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)* (2013), 1–4. doi:10.1109/ICCSPA.2013.6487269.

255. Mei, H., Jiang, H., Yin, F., Wang, L. & Farzaneh, M. Terahertz Imaging Method for Composite Insulator Defects Based on Edge Detection Algorithm. *IEEE Transactions on Instrumentation and Measurement* **70,** 1–10. doi:10.1109/TIM.2021.3075031 (2021).

256. He, Y., Song, K., Meng, Q. & Yan, Y. An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features. *IEEE Transactions on Instrumentation and Measurement* **69,** 1493–1504. doi:10.1109/TIM.2019.2915404 (2020).

257. Li, F., Li, F. & Xi, Q. DefectNet: Toward Fast and Effective Defect Detection. *IEEE Transactions on Instrumentation and Measurement* **70,** 1–9. doi:10.1109/TIM.2021.3067221 (2021).

258. Xu, L., Lv, S., Deng, Y. & Li, X. A Weakly Supervised Surface Defect Detection Based on Convolutional Neural Network. *IEEE Access* **8,** 42285–42296. doi:10.1109/ACCESS.2020.2977821 (2020).

259. Sreedhar, U., Krishnamurthy, C., Balasubramaniam, K., Raghupathy, V. & Ravisankar, S. Automatic defect identification using thermal image analysis for online weld quality monitoring. *Journal of Materials Processing Technology* **212,** 1557–1566. doi:https://doi.org/10.1016/j.jmatprotec.2012.03.002 (2012).
260. Li, Y., Cheng, Y., Gan, Z., Yu, L., Wang, L. & Liu, J. *BachGAN: High-Resolution Image Synthesis From Salient Object Layout* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).
261. Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., *et al. Training deep networks with synthetic data: Bridging the reality gap by domain randomization* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2018), 969–977.
262. Lin, J., Zhang, R., Ganz, F., Han, S. & Zhu, J.-Y. *Anycost GANs for Interactive Image Synthesis and Editing* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 14986–14996.
263. Schraml, D. *Physically based synthetic image generation for machine learning: a review of pertinent literature* in *Photonics and Education in Measurement Science 2019* (eds Zagar, B., Mazurek, P., Rosenberger, M. & Dittrich, P.-G.) (SPIE, 2019). doi:10.1117/12.2533485.
264. Figueira, A. & Vaz, B. Survey on Synthetic DataGeneration, Evaluation Methods and GANs. *Mathematics* **10.** doi:10.3390/math10152733 (2022).
265. He, K., Gkioxari, G., Dollár, P. & Girshick, R. *Mask R-CNN* in *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2980–2988. doi:10.1109/ICCV.2017.322.
266. Community, B. O. *Blender - a 3D modelling and rendering package* Blender Foundation (Stichting Blender Foundation, Amsterdam, 2018).
267. Simonyan, K. & Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition* in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (eds Bengio, Y. & LeCun, Y.) (2015).
268. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., *et al. Going deeper with convolutions* in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 1–9. doi:10.1109/CVPR.2015.7298594.
269. Huang, G., Liu, Z., van der Maaten, L. & Weinberger, K. Q. *Densely Connected Convolutional Networks* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).
270. Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. & Belongie, S. *Feature Pyramid Networks for Object Detection* in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 936–944. doi:10.1109/CVPR.2017.106.

271. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., *et al. Microsoft COCO: Common Objects in Context* in *Computer Vision – ECCV 2014* (Springer International Publishing, Cham, 2014), 740–755.
272. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., *et al.* MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155* (2019).
273. Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., *et al. Adapting deep visuomotor representations with weak pairwise constraints* in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics* (2020), 688–703.
274. Pashevich, A., Strudel, R., Kalevatykh, I., Laptev, I. & Schmid, C. Learning to Augment Synthetic Images for Sim2Real Policy Transfer. *CoRR* **abs/1903.07740** (2019).
275. Sajjan, S., Moore, M., Pan, M., Nagaraja, G., Lee, J., Zeng, A., *et al. Clear grasp: 3d shape estimation of transparent objects for manipulation* in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), 3634–3642.
276. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. & Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* **88,** 303–338 (2010).
277. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., *et al.* in *Advances in Neural Information Processing Systems 32* (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R.) 8024–8035 (Curran Associates, Inc., 2019).
278. Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., *et al.* ResNeSt: Split-Attention Networks. *arXiv preprint arXiv:2004.08955* (2020).
279. Cai, Z. & Vasconcelos, N. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43,** 1483–1498. doi:10.1109/TPAMI.2019.2956516 (2021).
280. Ganaie, M. A., Hu, M., Tanveer, M. & Suganthan, P. N. Ensemble deep learning: A review. *CoRR* **abs/2104.02395** (2021).