Graduation project

Multi-Sensor Fusion for Soft Robot Pose Estimation Using Co-Training Method

ME
Xinhai Zhou

Delft University of Technology

TU Delft

Graduation project

Multi-Sensor Fusion for Soft Robot Pose Estimation Using Co-Training Method

by

Xinhai Zhou

Student Name	Student Number
Xinhai Zhou	5723876

Instructor:	Dr. Jovana Jovanova
Teaching Assistant:	Q.(Aaron) Chen
Project Duration:	28-June-2024 - 30-Jan-2025
Faculty:	Faculty of Mechanical Engineering , Delft
Report number:	2024.MME.9021

Cover: https://wall.alphacoders.com/big.php?i=351853 Discovered By: Abandoned

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Preface

In recent years, soft robotics has emerged as a transformative field due to its unique capabilities, such as adaptability and safety in interacting with delicate objects and dynamic environments. Unlike traditional rigid robots, soft robots are made from flexible materials, allowing them to perform tasks in complex and unstructured settings. However, one of the primary challenges in soft robotics is accurate pose estimation. The inherent deformability and non-linear behaviors of soft robots make traditional pose estimation methods unreliable, particularly in scenarios requiring high precision.

To tackle this issue, multi-sensor fusion has gained attention as a robust solution. By combining data from diverse sensor modalities, such as inertial measurement units (IMUs), cameras, and strain sensors, it becomes possible to mitigate the limitations of individual sensors and improve the accuracy and reliability of pose estimation. In this project, we employ a co-training method, a semi-supervised machine learning approach, to enhance the fusion process. This method leverages the complementary strengths of multiple sensor types, iteratively refining pose estimation through collaborative training, even with limited labeled data.

The goal of this graduation project is to develop a multi-sensor fusion framework using the co-training method to achieve precise and robust pose estimation for soft robots. The project focuses on integrating data from IMUs, stereo cameras, and strain sensors, designing an effective fusion algorithm, and validating the system through simulations and experiments. By addressing the challenges associated with soft robot pose estimation, this work aims to contribute to the advancement of soft robotics and its applications in fields such as healthcare, manufacturing, and exploration.

Xinhai Zhou
Delft, January 2025

Contents

1

Introduction

Posture recognition and control in soft robots present significant challenges due to their inherent properties, such as infinite degrees of freedom and nonlinear material responses, which complicate dynamic modeling and real-time control, making it difficult to accurately predict and control their posture and movements. While the infinite DOF provides exceptional flexibility, it also greatly increases the complexity of the control systems, requiring advanced modeling techniques like finite element methods or Cosserat rod theory to approximate their behavior, often involving trade-offs between computational cost and accuracy. Additionally, the nonlinear material responses of soft robots make their reactions to various stimuli unpredictable, necessitating the development of innovative and adaptive control systems. The challenges are further exacerbated by the underactuated and redundantly actuated nature of soft robots, which demand sophisticated control strategies capable of handling large elastic deformations and unknown material parameters. To address these issues, researchers commonly employ discretization techniques to reduce the infinite-dimensional configuration space into a finite set of functional shapes, aiding in the prediction of the robots' final states and dynamic behavior. Neural networks and data-driven approaches are also widely utilized to enhance posture recognition and control.

In this study, we have developed a novel soft robotic structure utilizing smart materials as actuators, aiming to harness their unique adaptability and responsiveness. The primary focus of this report is on the recognition component of the soft robot, which plays a critical role in accurately identifying and analyzing its posture and dynamic behavior. At the same time, it will provide a basis for the subsequent design of control methods for soft robots.

The designed soft robotic structure, as depicted in the illustration, features a cylindrical configuration. The top and bottom planes of the structure are supported by two precisely fabricated PMMA plates, cut using laser technology, which provide the necessary structural stability. These support plates are interconnected by a central cylindrical core made of shape memory polymer (SMP), chosen for its ability to adapt and respond to external stimuli. Surrounding the SMP cylinder are three shape memory alloy (SMA) springs, strategically positioned to actuate the structure.

SMP materials exhibit a high degree of temperature sensitivity, primarily reflected in the significant variation in their Young's modulus. At room temperature, SMP materials possess a high Young's modulus, ensuring structural rigidity. However, when the material is heated and reaches a critical threshold temperature, its Young's modulus decreases dramatically, resulting in increased flexibility. Similarly, SMA materials also display notable temperature-sensitive properties. They are typically designed with a predefined memory shape, which allows them to recover to this shape upon heating after being deformed at room temperature. The specific control methods for this structure will be discussed in detail in the following sections.

To achieve precise control of the structure, accurate posture recognition is essential. In this study, a combination of IMU sensors and stereo vision is utilized for posture recognition of the structure. Additionally, due to the lack of true data in practical applications, a co-training machine learning approach is employed to train the recognition algorithm.

1.1. Research question

Soft robots, with their inherent infinite degrees of freedom and nonlinear material behaviors, present significant challenges for posture recognition. These challenges are further compounded by the absence of true data in practical applications, which limits the development of accurate and reliable recognition systems. Addressing these complexities requires innovative approaches that can effectively utilize available sensor data and advanced computational methods.

Thus, the main research question is: how to develop a novel method combining multi-sensor data and machine learning techniques to achieve accurate and reliable posture recognition for soft robots?

The key questions this study seeks to address are:

- How can an efficient posture recognition framework be designed to accommodate the nonlinear and complex properties of soft robots?

- How can posture recognition accuracy be improved in the absence of true data in practical applications?

- How can multi-sensor data (IMU and stereo vision) be effectively integrated to enhance recognition performance?

- How can machine learning and co-training methods be utilized to address data scarcity and improve recognition precision?

1.2. Research Objectives

1. Build an appropriate machine learning model for posture detection of the soft robot, tailored to its unique characteristics and operational requirements. 2. Use simulation methods to create an initial dataset for training the posture detection model, addressing the challenge of limited real-world data. 3. Implement the developed approach to detect the posture of the soft robot and evaluate its effectiveness in practical scenarios.

1.3. Report Structure

As shown in figure 1.1, the research work of soft robot hand is mainly divided into the following parts, the left part is the structure design, the middle part is the design of attitude tracking algorithm while the right side is the design of control algorithm, the main research of this report is for the middle part, so it will be centered on the study of literature on soft material, tracking algorithm, sensor characteristics, and so on, and then the design of the corresponding algorithms.

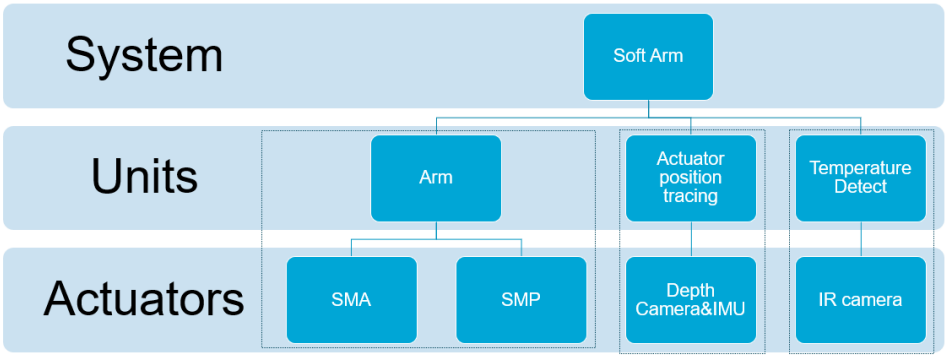

Figure 1.1: Research Roadmap

The structure of this report is as follows: Literature review part is focusing on existing methods and identifying research gaps to establish the foundation for this study. The hardware design and sensor installation section outline the overall structure of the soft robot and the integration of sensors. Data acquisition and processing discusses methods for collecting, synchronizing, and preparing data for analysis. The fusion methodology and algorithm implementation section introduces the approach for

combining multiple data sources and details the implementation of the proposed algorithm. Experimental validation describes the setup, testing scenarios, and result analysis to evaluate system performance. Finally, the conclusion summarizes the findings and contributions of the study while suggesting directions for future research.

2

Literature review

Smart materials have become a research focus in recent years. These advanced materials can respond to changes in their environment—such as temperature, pressure, humidity, electric fields, or magnetic fields—by altering their own properties accordingly. They find applications in various industries, including medicine, textiles, and transportation.

The primary characteristic of smart materials is their ability to change their physical properties in response to external stimuli. This feature makes them suitable for complex working environments. For instance, when exposed to changes in temperature, pressure, humidity, light, electric fields, or magnetic fields, smart materials can modify their properties, such as volume, density, and viscosity. This adaptability allows them to function as actuators, capable of operating in intricate and specialized environments. Examples include using them as actuators for underwater grippers or in the manufacturing of soft robotic actuators. Additionally, they can serve as sensors in specific situations, detecting variations in environmental factors.

The ability of these materials to respond to environmental changes opens up new possibilities for material development. It enables seamless interaction between manufactured objects and their surroundings. However, smart materials also possess complex characteristics with a high degree of freedom, making them challenging to control. Therefore, before using such materials, detailed modeling is typically required to determine the physical properties of the systems built with them. Based on this modeling, it is essential to develop appropriate control and sensing systems, along with designing effective control algorithms, to better manage and utilize these materials.

2.1. Smart Materials

2.1.1. Piezoelectric Materials

Piezoelectric materials are known for their ability to convert mechanical energy into electrical energy and vice versa, and are of interest due to their diverse applications in various fields. These materials are integral to the development of electronic devices such as piezoelectric filters, microdisplacements, actuators and sensors, which are vital in information and communication technologies, biomedicine and military defence [1]. The versatility of piezoelectric materials extends to energy harvesting, where they can convert mechanical energy from human activities and environmental vibrations into usable electrical energy, thus solving the energy crisis and reducing environmental pollution [2, 3].

Piezoelectric polymers, especially poly (vinylidene fluoride) based ferroelectric polymers, are emerging as promising alternatives to conventional chalcogenide ferroelectrics due to their flexibility, biocompatibility, and lower cost, making them suitable for applications in soft and flexible sensors, actuators, robotics, and energy harvesters [4].

The fast response, high precision and adaptability of piezoelectric materials make them ideal for microelectromechanical systems (MEMS), nanoelectromechanical systems (NEMS) and 3D printing, with significant advances in lead-based and lead-free piezoelectric materials for applications in catalysis,

biomedical engineering and additive manufacturing [5].

In the field of smart materials, piezoelectric actuators are embedded in adaptive structures, such as aeroplane wings, to improve overall efficiency, and are also used in industrial applications such as diesel fuel injectors, optical lenses and vibration damping [6]. The selection of suitable piezoelectric materials depends on factors such as type, Curie temperature and environmental stability, with a focus on their ferroelectric behaviour and characterisation through various measurement techniques [7].

In addition, the coupled physics of piezoelectric materials, including piezoelectric photonics and piezoelectric electronics, is being explored for optical and optoelectronic micro- and nanodevices, highlighting their potential for future research and practical applications [8]. Continuing demands in the electrical, energy and biomedical fields have prompted researchers to explore new combinations of materials and devices, while engineers have endeavoured to enhance existing technologies, highlighting the broader impact and importance of piezoelectric materials in precision and acoustic engineering [9].

Overall, advances in piezoelectric materials, from traditional ceramics to innovative polymers and composites, have demonstrated their critical role in modern technology and their potential to address future challenges in energy, healthcare and industrial applications.

2.1.2. Shape Memory Alloys(SMA)

Shape Memory Alloys (SMAs) are a unique class of metallic materials that can return to a pre-defined shape or size when subjected to appropriate thermal or mechanical stimuli, a phenomenon known as the shape memory effect.

These alloys typically consist of mixtures of martensites and austenite, which can transform into one another through thermal or mechanical actions, resulting in their distinctive properties [10, 11]. The most well-known SMAs include nickel-titanium (Ni-Ti) alloys, often referred to as Nitinol, and copper-based alloys such as Cu-Zn-Al and Cu-Al-Ni, which exhibit significant strain recovery and force generation upon phase transformation [12].

SMAs can undergo reversible phase transitions due to changes in temperature, pressure, or stress, exhibiting pseudoelasticity and, under certain conditions, linear superelasticity, allowing for large, recoverable strains [13, 14, 15]. The composition of these alloys can vary, with some including elements like aluminum, zirconium, cobalt, chromium, and iron to enhance their properties [16]. For instance, a specific SMA composition might include 2 to 15% aluminum, 0.01 to 3% beryllium, and the balance being copper, with possible additions of zinc to allow for cold work [17].

The mechanisms behind the shape memory effect and superelasticity in Ni-Ti alloys are based on thermally or stress-induced martensitic phase transformations, which are crucial for their thermomechanical properties, corrosion resistance, and biocompatibility, making them suitable for medical devices [18]. SMAs are utilized in a wide range of applications, including tubes and valves in piping systems for power plants, ships, and the petroleum industry; explosive bolts in aerospace and construction; packaging devices for electronic materials; and dental materials, prosthetics, and other biomedical devices [13, 14, 15]. They are also found in sensors in automobiles, consumer products, and generally in smart materials and adaptive structures, thanks to their ability to produce significant force and recover substantial amounts of strain [12, 15]. The remarkable behaviors of SMAs are attributed to their atomic-level and microstructural mechanisms, which enable novel manufacturing techniques, deployable mechanisms, and adaptive structures, particularly in the aerospace field [19].

Despite their impressive capabilities, SMAs can present challenges due to some less known and unusual phenomena in their thermo-mechanical response, which can create pitfalls for casual users [19]. Overall, the unique properties and diverse applications of SMAs make them a critical material in various advanced technological fields.

2.1.3. Shape Memory Polymers(SMP)

Shape Memory Polymers (SMPs) are a class of smart materials that can return from a deformed state to their original shape upon exposure to an external stimulus such as temperature change, light, or moisture. These polymers are typically composed of a combination of hard and soft segments, which contribute to their unique properties. For instance, SMPs can be synthesized using a variety of monomers

and crosslinkers, such as linear chain acrylates and multi-functional acrylate cross-linkers, which exhibit shape memory effects at specific transition temperatures [20].

The shape memory effect in polymers is primarily due to their ability to undergo a phase transition, which can be tailored by adjusting the polymer composition and structure. For example, a shape memory polymer can be formed by esterification and polycondensation of aromatic diacids, linear aliphatic diacids, and linear aliphatic diols, resulting in a polymer with a glass transition temperature (Tg) of 30°C to 100°C and a melting point (Tm) of 170°C to 250°C [21]. Additionally, SMPs can be designed to have high shape recovery rates and recovery forces, making them suitable for various applications, including medical devices and actuators [22].

The versatility of SMPs is further enhanced by their ability to hold multiple shapes in memory, which can be achieved by incorporating hard and soft segments with different transition temperatures. Moreover, SMPs can be synthesized using different chemical compositions, such as cyclooctene and multicyclic dienes, to achieve specific mechanical properties and functionalities [23]. The development of SMPs has also led to the creation of materials with unique properties, such as water- and solvent-driven shape memory effects, which are particularly useful for in vivo applications [24].

Furthermore, SMPs can be engineered to have antibacterial properties and biocompatibility, making them ideal for medical and engineering applications [25]. The history and commercialization of SMPs have seen significant advancements, with various models proposed to describe their behavior and potential for tailored properties [26]. SMPs can also be used in innovative applications such as contact lens molds, where their ability to return to a predetermined shape is highly beneficial [27].

The ongoing research and development in the field of SMPs continue to expand their potential applications, including in 3D printing, where they offer numerous design possibilities and functional architectures [24]. Overall, SMPs represent a rapidly evolving field with a wide range of applications, driven by their unique ability to respond to external stimuli and return to their original shape [28].

2.1.4. Hydrogels

Hydrogels are advanced materials characterized by their ability to respond to various external stimuli, making them highly suitable for biomedical and nanotechnological applications. These hydrogels are formed by crosslinking polymeric networks that can imbibe water and remain insoluble, thanks to their hydrophilic functional groups such as -OH, -CONH2, -SO3H, -CONH, and -COOR, which enable them to act as super absorbents and controlled release systems for drugs [29].

The versatility of smart hydrogels is further enhanced by their tunable electrical and mechanical properties, biocompatibility, and multi-stimulus sensitivity, making them ideal for wearable health monitoring devices that track physiological data continuously [30]. Recent advancements have led to the development of semi-interpenetrating polymeric networks (semi-IPN) based on collagen-polyurethane-alginate, which exhibit improved swelling capacity, storage modulus, and resistance to degradation, along with the ability to stimulate or inhibit cellular activities depending on the application, such as wound healing or anticancer therapies [31].

These hydrogels can respond to a variety of stimuli, including pH, temperature, light, enzyme activity, redox agents, and electric or magnetic fields, offering significant potential for drug delivery, tumor therapy, tissue engineering, and biodevices [32]. The structural and phase transitions triggered by these stimuli allow for precise control over the release of therapeutic agents, making them highly effective in clinical settings [29].

The ongoing research and development in this field are focused on overcoming the challenges of translating these materials from academic research to clinical applications, with a promising future for their use in various therapeutic and diagnostic applications [33]. Overall, smart material hydrogels represent a rapidly evolving field with immense potential to revolutionize healthcare and nanotechnology.

2.1.5. Electrochromic Materials

Electrochromic materials, which exhibit reversible changes in their optical properties (such as color, transparency, and reflectivity) under an applied electric field, have garnered significant attention due to their wide range of applications in smart windows, wearable electronics, displays, adaptive camouflage,

and more [34] [35] [36]. These materials can be broadly categorized into organic and inorganic types, each with distinct advantages and challenges.

Organic electrochromic materials, for instance, offer benefits like easy molecular modification, rich color variations, and fast switching speeds, making them suitable for applications requiring rapid response times and diverse color options [36]. However, the development of stable cathodic coloring organic materials remains challenging due to the inherent instability of electron-accepting compounds, although recent advancements in phthalimide derivatives have shown promise in enhancing stability and performance [37].

Inorganic electrochromic materials, such as transition-metal oxides, are favored for their low power requirements, high coloration efficiency, and durability. Nickel oxide (NiO), for example, is a p-type semiconductor known for its high color contrast ratio and chemical stability, making it a popular choice for flexible and soft electrochromic devices [38]. Recent innovations have also explored the use of the Burstein–Moss effect in degenerate semiconductors to achieve multicolor tunability, which could simplify device structures while offering selective modulation of visible and near-infrared light [39]. Additionally, the integration of optical resonators has been shown to enhance the color contrast and enable multicolor tuning by converting small refractive index changes into high-contrast color variations, thus expanding the potential applications of electrochromic materials in multichromatic displays and adaptive camouflage [35].

The development of new anodic coloring materials, such as IrO2-doped NiO films, has further improved the performance of complementary energy storage electrochromic devices (ESECDs), achieving rapid switching times and high cycling durability [40]. Moreover, the extension of electrochromic technology into the infrared region has opened new avenues for applications in infrared stealth, thermal control of spacecraft, and smart windows, with research focusing on optimizing materials like metal oxides, plasma nanocrystals, and carbon nanomaterials to enhance performance [41].

The advent of 2D materials, including covalent organic frameworks and MXenes, has also shown potential in addressing the limitations of conventional electrochromic materials by offering improved optical contrast, response time, and flexibility [42]. Educational methods have even been developed to demonstrate electrochromism using household materials, highlighting the accessibility and educational value of this technology [43].

2.2. Structure Modelling of Smart Materials

Modeling of smart materials is essential for several key reasons. Firstly, it enables a deeper understanding of their unique properties and behaviors, which is crucial for predicting and controlling these materials effectively. Engineers can design more efficient systems by comprehending the underlying mechanisms and behaviors of smart materials through precise models [44]. Secondly, modeling allows for performance optimization of smart material systems before the creation of physical prototypes. By simulating the behavior of these materials and their interactions with the environment, designers can determine the best configurations, sizes, and control strategies to meet desired performance goals.

Moreover, modeling helps save time and costs. Physical prototypes of smart material systems can be expensive and time-consuming to produce. By using models, engineers can quickly evaluate different design options and iterate virtually, reducing the need for multiple physical prototypes and associated costs, thereby accelerating the development process. This approach can significantly shorten the development cycle and lower expenses.

Additionally, modeling reduces risks associated with the complex and often nonlinear responses of smart materials. Engineers can analyze and predict material behavior under various conditions and stimuli, identifying potential risks like failure points or undesirable behaviors and mitigating them during the design phase to enhance overall system reliability.

Furthermore, modeling facilitates the exploration of design trade-offs. Engineers can simulate and analyze different design choices and their impacts on performance metrics such as energy consumption, response time, or durability, allowing for informed decision-making and optimal balance between conflicting design goals.

Finally, models provide scalability and versatility. Smart materials are used in a wide range of applications and environments. Reliable models capturing the fundamental behaviors of smart materials can be applied to various systems and scenarios, serving as a foundation for designing diverse smart material-based systems and saving time and effort for each new application.

In conclusion, modeling enables engineers to understand, optimize, and control the behavior of smart materials, leading to the development of more efficient and reliable systems while reducing development time and costs. Smart material modeling is a multidisciplinary process that integrates knowledge of material properties, physics, numerical methods, and simulation technologies. The specific modeling methods vary depending on the type of smart material and its intended application, often employing advanced software tools and specialized modeling techniques.

2.3. Sensors for soft robotics

Sensoring soft robotics would enable them to detect changes in their environment and respond accordingly. Smart materials are designed to react to stimuli such as temperature, pressure, moisture, or chemical changes, and the role of sensoring is to provide the necessary data for these materials to function effectively.

We will begin by discussing commonly used sensors that can measure stress, strain, temperature, and posture in soft robotics. These sensors play a critical role in providing the necessary data for monitoring and controlling soft robots, enabling them to operate effectively in dynamic and complex environments. By examining their characteristics and compatibility with soft robotic systems, we can better understand their potential applications and limitations in practical scenarios.

2.3.1. Stress Sensors

Piezoelectric Stress Sensor

When a piezoelectric material is subjected to mechanical stress, it produces an electric charge proportional to the amount of stress [45]. This charge can be measured and analyzed to determine the magnitude and nature of the applied stress. Piezoelectric sensors offer several advantages: they exhibit high sensitivity, detecting very small changes in stress; they have a fast response, providing immediate feedback to changes in stress, making them suitable for dynamic measurements; they do not require external power as they generate their own electrical signal in response to mechanical stress; and they are robust, functioning in harsh environments, including high temperatures and pressures. These characteristics make piezoelectric sensors highly suitable for applications in smart materials. Smart materials often need to monitor and respond to changes in their environment, and the high sensitivity and fast response of piezoelectric sensors enable precise and real-time monitoring of stress and strain.

Kathleen Coleman et al. dedicated to advancing the application of piezoelectric sensors in the realm of the Internet of Things (IoT) and wearable electronics, focusing on flexible electronic stress monitoring [46]. To achieve this, they opted to use 120 µm thick Polyvinylidene Fluoride (PVDF) films. PVDF was chosen due to its high piezoelectric coefficient (g33) of 0.287 Vm/N and its compatibility with flexible electronic applications.

In their experimentation, the researchers integrated these piezoelectric sensors into a system using a Texas Instruments MSP430 Micro-Controller Unit (MCU) to capture and react to variations in stress levels. They designed an impedance-matching circuit to optimize the sensor's signal for digital processing, utilizing SPICE software to model a Thevenin equivalent circuit that included the piezoelectric source and pull-up resistors with varying impedances. The final system was programmed such that an LED would light up when the stress exceeded a threshold, corresponding to a critical radius of curvature of 10 mm, a point at which the flexible PCB was deemed to fail.

The outcome of their research highlighted a novel application for piezoelectric sensors within flexible hybrid electronics (FHE), demonstrating the crucial role of proper impedance matching and sophisticated circuit design to effectively integrate piezoelectric films with external circuits and hardware. The successful activation of the LED alert system upon reaching the predefined stress threshold showcased a viable method for real-time stress monitoring in flexible electronics, providing a practical tool for enhancing the reliability and performance of IoT and wearable devices.

This research underscored the effectiveness of piezoelectric sensors for monitoring mechanical stress in flexible electronics, a key factor for the durability and functionality of IoT and wearable technologies. It also stressed the importance of careful circuit design, particularly impedance matching, to ensure precise signal digitization and optimal sensor performance.

However, there are areas identified for potential improvement. Exploring alternative materials or different thicknesses of the piezoelectric films might yield better performance or facilitate easier integration with flexible electronics. Further development of more sophisticated algorithms for the MCU could enable more detailed stress analysis and even predict potential failures before they occur. Additionally, evaluating the durability and performance of the piezoelectric sensors under varied environmental conditions would be beneficial to confirm their reliability in practical applications.

Piezoresistive Stress Sensor

When a piezoresistive material is subjected to mechanical stress, its electrical resistance changes proportionally to the amount of stress [47]. This change in resistance can be precisely measured using a Wheatstone bridge circuit or other resistance-measuring techniques, allowing for the determination of the applied stress. Piezoresistive sensors offer several advantages: they exhibit high sensitivity, detecting very small changes in stress; they have a fast response, providing immediate feedback to changes in stress, making them suitable for dynamic measurements; they have a simple and compact construction; and they can be easily integrated with electronic circuits for signal processing and data acquisition. These characteristics make piezoresistive sensors highly suitable for applications in smart materials. Smart materials often need to monitor and respond to changes in their environment, and the high sensitivity and fast response of piezoresistive sensors enable precise and real-time monitoring of stress and strain. Additionally, their simple construction and ease of integration with electronic systems ensure compatibility with various smart material applications, including flexible electronics, wearable devices, and structural health monitoring systems.

Minliang Li and his team aiming to innovate in the field of sensor technology by developing a new type of piezoresistive sensor with distinctive characteristics. Their methodology involved a novel approach: doping reduced graphene oxide (rGO) powder into a graphene oxide (GO) film. This unique combination was engineered to create a flexible piezoresistive sensor that not only responds to changes in pressure by altering its resistance but also possesses rectification properties, enabling it to control the direction of electrical current flow.

The fruits of their labor were impressive. The newly developed sensor demonstrated remarkable performance metrics, including a high peak sensitivity of 9.65 kPa−1, a swift response time of just 72 milliseconds, and a quick recovery time of 26 milliseconds. Notably, the sensor maintained its stability even after undergoing 5500 cycles of pressure testing, underscoring its durability.

The study identified areas ripe for further enhancement: While the sensor's performance is already impressive, advancing its sensitivity beyond the current peak could broaden its applicability, particularly for detecting subtler pressure variations. Enhancing the sensor's response and recovery times could augment its utility in scenarios that demand rapid detection and action. Future research might explore alternative materials or novel combinations thereof to further improve the sensor's rectification properties and overall performance.

In detail, the researchers undertook the following steps in their study: They prepared the innovative material by doping rGO powder into a GO film, thus creating a composite material with unique piezoresistive and rectification properties. They then fabricated the sensor by incorporating this composite into a flexible substrate, enabling it to maintain functionality even when bent or flexed. Comprehensive performance testing was conducted, which included evaluating the sensor's pressure sensitivity, response and recovery times, and long-term durability under repeated use. Additionally, the sensor's capability to rectify low-frequency AC signals was specifically tested, highlighting its unique ability to control current flow direction—a feature not typically seen in standard piezoresistive sensors.

2.3.2. Temperature Sensors

Infrared Temperature Sensor

Infrared (IR) temperature sensors operate based on the principle of detecting infrared radiation emitted by objects. All objects emit infrared radiation as a function of their temperature, according to Planck's

law of black-body radiation. The sensor measures the intensity of this radiation and converts it into an electrical signal, which is then processed to determine the temperature of the object. This non-contact method allows for temperature measurement from a distance, making it ideal for applications where direct contact is impractical or impossible [48].

The characteristics of infrared temperature sensors make them highly suitable for smart material applications. Smart materials often require precise temperature monitoring to maintain their functional properties. The non-contact nature of IR sensors ensures that the measurement process does not interfere with the material's properties or performance.

Ruowei Li et al. focused on enhancing the accuracy of non-contact body temperature measurements, essential during the COVID-19 pandemic, using the ZTP-135SR thermopile infrared temperature sensor known for its non-contact measurement capabilities [49]. Researchers aimed to refine these measurements to reduce the risk of infection transmission. To improve the accuracy of the temperature data from the sensor, they extracted the raw temperature readings and applied post-amplification and filtering techniques to strengthen the signals and eliminate noise. Additionally, they incorporated environmental compensation through polynomial fitting, which adjusted the temperature readings based on the environmental conditions to yield more accurate results.

The successful enhancement of the infrared temperature measurement system provided quick, non-invasive, and accurate body temperature assessments. Such advancements are crucial for monitoring health and controlling the spread of infectious diseases like COVID-19. The study highlighted the importance of sophisticated processing techniques such as post-amplification, filtering, and polynomial fitting for environmental compensation in ensuring the reliability of temperature measurements in healthcare settings.

While the system showed significant improvements, there are opportunities for further enhancements. Future research could explore integrating advanced algorithms or machine learning techniques to elevate the accuracy and reliability of the temperature measurement system further. Additionally, examining the long-term stability and durability of the sensor system under varied environmental conditions would help ensure consistent performance over time. Questions about the specific data involved in polynomial fitting, the practical implementation of post-amplification, and the environmental conditions tested for compensation would provide deeper insights into the system's robustness and application potential in different geographical areas.

Thermal Imaging Sensor

Thermal imaging sensors, also known as infrared cameras, operate by detecting the infrared radiation emitted by objects and converting it into an electronic signal to create a visual representation of temperature distribution. These sensors use an array of infrared detectors to capture the emitted radiation, which is then processed to form an image where different temperatures are displayed in varying colors. This allows for the visualization of temperature variations across a surface, providing detailed thermal profiles without the need for direct contact [50].

Similar to IR, the main advantages of the Thermal Imaging Sensor are that it is non-contact and accurate, and it can also generate a thermal map, which makes it is quite good for observing the heat distribution of the device.

Shengcheng Li,Tao Lu and their team focused on enhancing the detection of insulation defects in transformer bushings through the application of thermal imaging technology [51]. To achieve this, they utilized an infrared thermal imaging detection device equipped with a 640×512 uncooled infrared focal plane array and an ADV7390 video coding chip. This setup enabled the conversion of infrared light into electrical signals that could be analyzed to assess the condition of the insulation.

The researchers embarked on designing a thermal imaging detection system that was not only miniaturized and power-efficient but also boasted high resolution. They meticulously optimized the device for low power consumption, incorporating a reference source, digital potentiometer, and operational amplifier to maintain the output analog bias within a specified voltage range. Furthermore, they refined the signal processing method by implementing a scheme that divided the voltage and formed a differential signal with a reference voltage (VREF), which was then processed by the ADC chip.

Their efforts culminated in the creation of a device that successfully met the expected standards, with signal voltages maintaining around 2V, lower than the fluctuation range of the signal. This setup proved effective in detecting insulation defects in transformer bushings, with the capability to capture the infrared feature distribution of these defects and continuously track the infrared spectrum on the exterior of the bushing. By doing so, the device could pinpoint abnormal temperature positions indicative of potential issues.

From this research, it was evident that infrared thermal imaging technology could serve as a powerful tool for detecting insulation defects in transformer bushings. The integration of a miniaturized, low-power, high-resolution thermal imaging detection device facilitated accurate and reliable defect detection. The advancements in signal processing and hardware optimization were crucial in enhancing the device's effectiveness at identifying critical temperature discrepancies that signal insulation defects.

However, there are opportunities for further advancements in this technology. Enhancing the resolution and sensitivity of the thermal imaging detection device could allow for the detection of even smaller defects, which would be beneficial for early diagnosis and maintenance planning. Incorporating more sophisticated algorithms could improve feature point matching and defect analysis, increasing the accuracy of the detection process. Additionally, further reductions in power consumption and device size could make the technology more practical for field applications, enabling more widespread and routine use in the maintenance and monitoring of transformer bushings. These improvements could significantly enhance the operational reliability and efficiency of power distribution systems, particularly in monitoring and preventing failures that could lead to service interruptions.

2.3.3. Motion Capture

Motion capture (mocap) technology involves recording and processing the movements of people or objects to generate corresponding virtual animations, and it has become integral in various fields such as video games, film production, sports analysis, and healthcare [52]. The technology can be implemented using different systems, including marker-based, vision-based, and volumetric capture systems. Marker-based systems, while accurate, are often inconvenient due to the need to attach markers to the body.

Ziyi Xin undertook a comprehensive review of motion capture technology, aiming to enrich understanding of its origins, methodologies, and broad applications across various fields [53]. Utilizing a detailed review method, they examined existing literature and studies on motion capture technology to provide an in-depth overview of its development, current state, and future prospects. This encompassed a historical exploration of the technology's origins, a look at how it has been realized over time, and an analysis of the mainstream methodologies used to capture motion. The researchers also delved into the various applications of motion capture technology in fields such as entertainment, sports analysis, human biomechanics, automotive, and virtual reality, showcasing its wide-ranging impact and potential.

The result of her investigation is a detailed paper that not only educates readers about the critical aspects and potential of motion capture technology but also highlights its significant market growth, with the 3D Motion Capture Market projected to reach $270.9 million by 2026. The paper offers insights into future research directions and underscores the increasing demand and investment in motion capture technology across various industries.

From this study, readers can understand the historical development and technological evolution of motion capture, which provides a solid foundation for appreciating the technical advancements in this field. The paper details how motion capture technology is applied in creating virtual characters in movies, enhancing athletic performance, and understanding human movement in biomechanics, among other uses, illustrating its versatility. The projected market growth further indicates the rising importance and potential of motion capture technology in numerous industries.

However, there are areas for potential improvement and further research. Enhancing the accuracy and efficiency of motion capture methods could lead to more precise and reliable data for various applications, and exploring new applications in emerging fields could open up new possibilities and expand the technology's use and impact. Additionally, making the device more compact and reducing its power consumption could benefit practical applications in the field.

Qingfeng Shi and Yunkun Cui focused on enhancing motion capture systems for motor skills through the use of computer vision technology [54]. They implemented a method that combined inertial conduction with wireless network transmission, aiming to capture and control motion more effectively. By strategically placing sensors on 17 crucial joints of the human body, including the shoulders, elbows, and knees, they were able to collect comprehensive movement data. This data was then transmitted in real-time to 3D modeling software for accurate action control.

The outcomes of their research demonstrated a significant improvement in the motion capture system, addressing previous limitations such as limited communication distance, low data accuracy, and narrow application range. The system showcased enhanced performance in several key areas: it provided improved accuracy due to the use of strapdown inertial navigation technology, enabled real-time control of actions via wireless transmission, and offered advanced visualization through 3D modeling software, making it easier for users to analyze and understand captured movements.

Despite these advancements, the study suggested potential areas for further development. Expanding the application range of the motion capture system to include fields like medical rehabilitation or virtual reality could enhance its versatility and impact. Optimizing sensor placement and quantity could further improve the accuracy and reliability of data collection. Additionally, enhancing the integration of the motion capture system with various 3D modeling software could make the system more user-friendly and adaptable to different uses.

2.4. Posture estimation algorithm

In the first part, we discussed the properties of various smart materials and introduced some modeling and analysis methods for these materials. Many smart materials are used in the manufacture of active actuators or robots that need to move precisely according to human intentions. Therefore, after analyzing and modeling these materials, we need to design appropriate control methods.

2.4.1. Proportional-Integral-Derivative Control (PID Control)

Proportional-Integral-Derivative (PID) control is a widely used control loop feedback mechanism in industrial and engineering applications due to its simplicity and reliable performance [55]. The PID controller adjusts the control input to a system based on three terms: the proportional term (P), which depends on the current error; the integral term (I), which accounts for the accumulation of past errors; and the derivative term (D), which predicts future errors based on the current rate of change. This combination allows the PID controller to correct errors quickly and efficiently, ensuring system stability and performance.

The output u(t) of the PID controller can be represented as [56]:

$$u(t) = Kp * e(t) + Ki * e(t)dt + Kd * de(t)/dt \tag{2.1}$$

The output $u(t)$ of the PID controller consists of the following three parts:

Proportional (P) part:

$$P = Kp * e(t) \tag{2.2}$$

Where Kp is the proportional coefficient, and $e(t)$ is the error at the current moment (target value - actual value).

Integral (I) part:

$$I = Ki * e(t)dt \tag{2.3}$$

Where Ki is the integral coefficient, and $\int e(t)dt$ is the integral of the error from the initial moment to the current moment.

Derivative (D) part:

$$D = Kd * de(t)/dt \tag{2.4}$$

Where Kd is the derivative coefficient, and $de(t)/dt$ is the derivative of the error at the current moment.

This is the basic working principle of the PID controller. By adjusting the three parameters Kp, Ki, and Kd, you can achieve precise control of the system output.

The proportional part can quickly correct the error, the integral part can eliminate the steady-state error, and the derivative part can improve the system's response speed and stability. With the coordinated cooperation of the three, you can achieve good control of the system.

In order to meet more usage scenarios and requirements, the PID controller also has some evolved and upgraded versions, mainly including the following:

Adaptive PID controller: Can automatically adjust the values of Kp, Ki and Kd according to the system status to adapt to changes in system parameters. By online identification of the system model, dynamically adjusting the PID parameters to improve control performance.

Fuzzy PID controlle: Combining fuzzy logic reasoning, using expert experience to perform fuzzy adjustment of PID parameters. Can better handle non-linear, highly uncertain complex systems.

Neural network PID controller: Using neural networks for self-learning and self-optimization of PID parameters. Can adapt to more complex non-linear systems and improve control accuracy.

Predictive PID controller: Combining system model to predict future states and make control decisions in advance. Can improve the system's response speed and anti-interference ability.

Hybrid PID controller: Combining PID control with other advanced control algorithms (such as LQR, H∞, etc.). Can fully utilize the advantages of various control algorithms to improve control performance. These evolved versions of the PID controller are designed for different application scenarios and system characteristics, and can better meet actual needs. They have been widely used in industrial automation, aerospace, new energy and other fields.

For example, Horst Meier et al. explored the use of shape memory alloys (SMAs) as actuators in various systems [57], focusing on the development of a smart control system to enhance their efficiency, dynamics, and reliability. The researchers used resistance feedback control systems to manage the energy input at specific key points, which are directly before the transformation into the high-temperature phase and shortly before the retransformation into the low-temperature phase (martensite start temperature. They conducted experiments and developed a mechatronic demonstrator system to validate the effectiveness of the resistance-controlled systems in improving the performance of SMA actuators.

They used a PID controller to optimize the control system for SMA actuators by detecting the electrical resistance of the actuator, which helps in maintaining precise control over the heating process. The PID controller adjusts the current supplied to the SMA actuator based on real-time feedback from the electrical resistance, ensuring that the actuator reaches the desired temperature for transformation without overheating.By using the PID controller, the system can quickly respond to changes in the actuator's state, minimizing delays and improving the overall efficiency and reliability of the transformation cycles.

The PID control algorithm is highly effective in regulating variables such as temperature, light intensity, and magnetic field strength. Its notable benefits include simplicity and low computational requirements. However, the process of tuning the parameters can be intricate and labor-intensive.

This hybrid approach allows the system to leverage the simplicity of PID control while benefiting from the adaptive learning capabilities of deep learning, especially in complex, nonlinear systems.

2.4.2. Model Predictive Control (MPC) with Deep Learning for Position Prediction

Model Predictive Control (MPC) is a sophisticated control strategy that uses a dynamic model of the system to predict and optimize future behavior over a finite horizon [58]. MPC can handle constraints on inputs and outputs, making it ideal for applications requiring precise position control.

The objective function typically includes terms for tracking error and control effort. A common form is [59]:

$$J = \sum_{k=0}^{N_p-1} \left[(y_{t+k} - y_{t+k}^{\text{ref}})^T Q (y_{t+k} - y_{t+k}^{\text{ref}}) + \Delta u_{t+k}^T R \Delta u_{t+k} \right] \tag{2.5}$$

where:

- y_{t+k} is the predicted output at time $t+k$.
- y_{t+k}^{ref} is the reference or desired output at time $t+k$.
- Δu_{t+k} is the change in control input.
- Q and R are weighting matrices for tracking error and control effort, respectively.
- N_p is the prediction horizon.

The predicted outputs are calculated using the system model. For a linear time-invariant system, the model can be represented as:

$$x_{t+k+1} = Ax_{t+k} + Bu_{t+k} \tag{2.6}$$

$$y_{t+k} = Cx_{t+k} + Du_{t+k} \tag{2.7}$$

where x is the state vector, u is the control input, and A, B, C, and D are matrices that describe the system dynamics.

The control actions must satisfy constraints, which can include:

- Input constraints: $u_{\min} \leq u_{t+k} \leq u_{\max}$
- State constraints: $x_{\min} \leq x_{t+k} \leq x_{\max}$
- Output constraints: $y_{\min} \leq y_{t+k} \leq y_{\max}$

The overall MPC problem can be written as a constrained optimization problem:

$$\begin{aligned}
\min_{u_t, u_{t+1}, \ldots, u_{t+N_c-1}} \quad & J \\
\text{subject to} \quad & x_{t+k+1} = Ax_{t+k} + Bu_{t+k}, \quad k = 0, 1, \ldots, N_p - 1 \\
& y_{t+k} = Cx_{t+k} + Du_{t+k}, \quad k = 0, 1, \ldots, N_p - 1 \\
& u_{\min} \leq u_{t+k} \leq u_{\max}, \quad k = 0, 1, \ldots, N_c - 1 \\
& x_{\min} \leq x_{t+k} \leq x_{\max}, \quad k = 0, 1, \ldots, N_p - 1 \\
& y_{\min} \leq y_{t+k} \leq y_{\max}, \quad k = 0, 1, \ldots, N_p - 1
\end{aligned}$$

where N_c is the control horizon.

Somasundar Kannan et al. used an adaptive predictive controller to manage the behavior of SMA linear actuators [60], which are devices that change shape in response to temperature changes and are used in various applications like robotics and medical devices. They successfully implemented MPC by employing a truncated linear combination of Laguerre filters to simplify the representation of dynamic systems, with real-time online updates. They thoroughly studied and proved the stability of the controller, showing that the tracking error is asymptotically stable under certain conditions related to modeling error, and that it converges to zero for step references despite potential inaccuracies in the identified model. Experimental results with two types of actuators validated the robustness of the control method, demonstrating its capability to handle input constraints without performance loss.

MPC offers several advantages, including strong predictive capabilities, the ability to handle multi-variable systems, optimization performance, and explicit handling of input and output constraints, making it highly effective in practical applications. Additionally, MPC exhibits good robustness in dealing with system uncertainties and external disturbances. However, MPC also has some limitations, such as high computational complexity, strong dependence on the accuracy of the system model, implementation complexity, high real-time requirements, and complex parameter tuning. Despite these challenges, the advantages of MPC make it a valuable choice for many complex control problems [61, 62].

(3) Deep Learning-based Position Prediction
While PID and MPC provide robust control frameworks, deep learning can be employed independently for position prediction, especially in scenarios with significant nonlinearities or where system models are unavailable. A recurrent neural network (RNN) or its variants, such as Long Short-Term Memory (LSTM) networks, can be used to predict future positions based on historical data [63].

The workflow for deep learning-based position prediction involves several critical steps, starting with the collection of time-series data from the system. This data typically includes recorded positions and corresponding control inputs over time, which serve as the foundation for training a deep learning model. The goal is to develop a model that can accurately predict future positions based on patterns identified in the historical data.

In the training phase, recurrent neural networks (RNNs) or long short-term memory (LSTM) networks are commonly employed due to their ability to process sequential data and capture temporal dependencies. These models are designed to map input sequences—such as current and past positions, control inputs, and any auxiliary system parameters—to future predicted positions. To ensure robust performance, the training process involves optimizing the model's parameters by minimizing the prediction error over a diverse set of training scenarios, including those that capture complex dynamics and disturbances.

Once the deep learning model is trained and validated, it can be deployed for real-time position prediction. During operation, the model takes current and past observations as input, such as recent position measurements and applied control signals, and outputs the predicted positions for a defined prediction horizon. This capability enables the system to anticipate future states, which can be invaluable for decision-making and control in dynamic environments.

Deep learning models excel in scenarios where traditional controllers, such as PID or MPC [64], may encounter challenges due to the system's nonlinearity, high levels of uncertainty, or unmodeled dynamics. However, rather than replacing conventional control methods, deep learning can complement them. For instance, PID or MPC can be used as the primary control mechanism to maintain system stability and responsiveness, while the deep learning model provides enhanced prediction capabilities. This hybrid approach allows the strengths of both methods to be leveraged—PID or MPC ensures robust and reliable control, while the deep learning model adapts to complex and uncertain dynamics, improving overall performance.

By integrating deep learning with traditional control strategies, the system gains the ability to not only respond to immediate errors but also anticipate future states, enabling proactive adjustments. This synergy is particularly beneficial in applications such as autonomous navigation, robotic control, and precision manufacturing, where accurate position prediction and control are critical for achieving high performance and reliability.

2.5. Conclusion

This review provides a comprehensive exploration of smart materials, their properties, and applications, with a particular focus on piezoelectric materials, shape memory alloys, shape memory polymers, hydrogels, and electrochromic materials. These materials exhibit unique responses to external stimuli, making them invaluable in various industries such as healthcare, energy, robotics, and electronics. The review also highlights the essential role of modeling and control strategies, such as PID and MPC, in optimizing the performance and functionality of systems integrating smart materials.

The discussion on sensors, including stress, temperature, and motion capture technologies, underscores their critical importance in enabling smart materials to interact with their environment effectively. The integration of advanced sensors enhances the functionality of smart materials, particularly in dynamic and complex applications like soft robotics.

The control methods examined, including traditional PID controllers, advanced MPC techniques, and deep learning-based approaches, demonstrate the versatility and adaptability required for managing the complex behaviors of smart materials. The hybridization of these control strategies offers promising avenues for improving system performance and reliability in non-linear and uncertain environments.

Future work should focus on refining these models and control strategies to better handle the inherent complexities of smart materials. Additionally, advancements in sensor technologies, coupled with the development of innovative algorithms, can further expand the potential applications of these materials.

3

Physical structure and Data acquisition

3.1. Overview

The schematic diagram of the structure, as illustrated below:

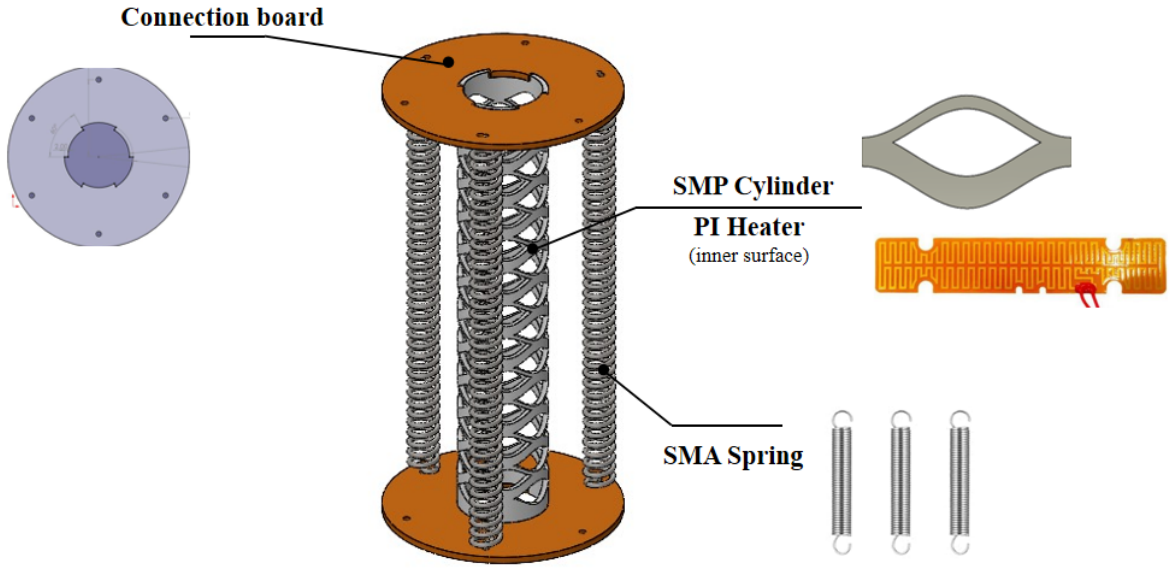

Figure 3.1: Schematic diagram of soft structure

The structure consists of several components, as illustrated. It features two circular transparent plates made of PMMA, which are precisely laser-cut. Between these plates, several slots are designed to accommodate SMP cylindrical rods. These rods are inserted into the slots, with both ends of the cylinders connected to corresponding structures, secured with high-temperature adhesive to ensure a fixed joint.

Additionally, three evenly distributed holes are drilled into the PMMA plates to install SMA springs. The springs are attached to the plates using zip ties for secure fixation. Inside the SMP cylinder, a layer of heating pad is applied, designed to heat the SMP and bring it to a high-temperature state. Each SMA spring is connected to a power source on both ends, allowing electric current to heat the SMA, enabling it to return to its memorized shape.

3.2. Circuitry part

The schematic diagram of circuitry part, as illustrated below:

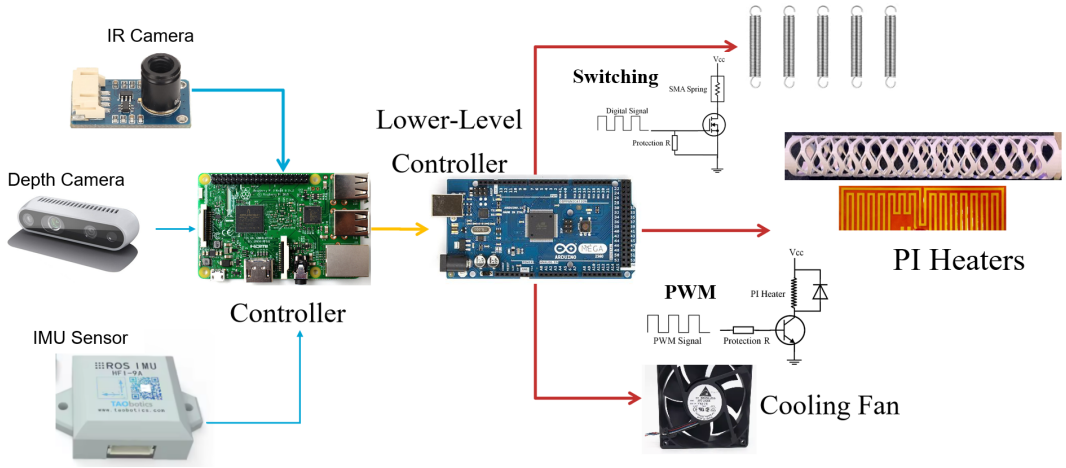

Figure 3.2: Schematic diagram of circuitry part

For the circuitry, we utilized an Arduino board to control the power supplied to the SMP heating pad and the current passing through the SMA springs. Additionally, a Raspberry Pi serves as the host controller. It is equipped with a thermal imaging camera to monitor the target's temperature in real time. The Raspberry Pi communicates with the Arduino by sending signals to adjust the heating levels, ensuring that both the SMP and SMA components reach their designated temperatures.

During operation, the heating pad on the SMP is first activated, raising the SMP to a predetermined temperature. This heating process significantly reduces the Young's modulus of the SMP, making it highly deformable and facilitating the overall structural deformation. Next, current is applied to the SMA, causing it to return to its memory shape and contract to a specific position. Once the desired position is reached, the power to the SMA is cut off, and the SMP is cooled down. Cooling restores the SMP's high Young's modulus, solidifying the current position and shape. This sequence constitutes the complete deformation process of the device.

3.3. Sensors

The imu that is illustrated below works for this structure stands out as a robust choice for applications requiring accurate inertial measurement [65]. It is designed to provide six degrees of freedom measurements, incorporating both a 3-axis accelerometer and a 3-axis gyroscope, ensuring precise motion detection. The sensor features high sampling rates and low noise levels, making it well-suited for real-time applications, such as robotics and motion tracking. Additionally, its compact design, lightweight structure allow seamless integration into robotic systems. A notable advantage is its efficient data output, which supports reliable motion analysis without requiring extensive post-processing.

Figure 3.3: IMU Sensors

Some of its technical parameters are listed in the following table:

	Accelerometer	**Gyroscope**
Sampling frequency	Max 200Hz	Max 200Hz
Noise	0.5mg	0.01°
Non-linear	0.06%fs	0.06%fs
Degree of freedom	3	3
Range	[-8g, 8g]	[-2000°/s, 2000°/s]
ADC resolution	4096LSB/g	16.38LSB/(°/s)

Table 3.1: Specifications of IMU

The Intel® RealSense™ Depth Camera D435 is a popular choice for visual recognition tasks as shown below due to its advanced depth-sensing capabilities and versatile features [66]. It employs stereoscopic depth technology with global shutter sensors, providing high-quality depth perception even in dynamic environments. The D435 offers a wide field of view (87° × 58°), enabling it to capture a broad scene, which is particularly beneficial for applications like robotics and augmented reality where comprehensive environmental understanding is crucial. INTEL REALSENSE Additionally, the camera supports high frame rates, delivering depth output resolutions up to 1280 × 720 at 90 frames per second, ensuring smooth and accurate motion tracking. Its compact form factor (90 mm × 25 mm × 25 mm) and USB-C connectivity facilitate easy integration into various systems. It was chosen as an alternative detection method for the attitude of this structure.

Figure 3.4: Intel Realsense D435

Some of its important technical parameters are listed below:

Intel Realsense D435	
Depth output resolution	1280 × 720
Depth Accuracy	Around 2%
Depth frame rate	90FPS
RGB frame resolution	1920 × 1080
RGB frame rate	30FPS

Table 3.2: Specifications of Intel Realsense D435

3.4. IMU Data Generation

The training method employed in this study is the co-training learning approach. During the initial phase of training, this method requires a set of simple data to generate the initial model. To address this need, a MATLAB-based program for trajectory simulation and data generation has been developed.

The IMU module in MATLAB is a powerful tool designed to realistically simulate the behavior of inertial measurement units (IMUs), including accelerometers, gyroscopes, and magnetometers [67]. This module is renowned for its ability to accurately replicate the characteristics of real-world sensors, such as response range, sampling rate, and dynamic properties. By using this module, users can generate data that closely mirrors the output of physical hardware, enabling algorithm development and validation without the need for actual sensor devices.

One of the most notable features of the IMU module is its precise noise simulation capability. It can model common noise characteristics observed in real sensors, including random bias drift, high-frequency white noise, and cumulative errors due to random walk. These noise models align closely with the behaviors seen in real-world IMUs, ensuring that the simulated data is highly realistic. Additionally, the module allows for the modeling of manufacturing imperfections and calibration inaccuracies, such as scale factor errors and cross-axis sensitivity, further enhancing the authenticity of the data.

This tool is particularly valuable for algorithm developers as it provides a controlled simulation environment for testing complex algorithms, such as attitude estimation and position tracking. For example, when developing filters like the Kalman Filter, the noisy data generated by the IMU module helps developers identify potential issues and optimize algorithm performance early in the development process. This is especially beneficial when hardware devices are expensive or not readily available.

Moreover, the MATLAB IMU module offers significant flexibility, allowing users to customize various sensor parameters, such as range, sensitivity, and noise levels, to suit specific application scenarios. This adaptability makes the module suitable for simulating both high-precision sensors and low-cost devices.

The following section illustrates the specific application methods of this fuction.

Creating an imuSensor Object

```
IMU = imuSensor('accel-gyro', 'SampleRate', 100);
```

This command creates an IMU model with an accelerometer and gyroscope, with a sampling rate of 100 Hz.

Configuring Sensor Parameters

Each sensor can be customized using parameter objects:

Accelerometer:

```
accelParams = accelparams('MeasurementRange', 16, 'Resolution', 0.002395);
IMU.Accelerometer = accelParams;
```

Gyroscope:

```
gyroParams = gyroparams('MeasurementRange', 69.8132);
IMU.Gyroscope = gyroParams;
```

Generating IMU Data

To simulate IMU data, provide the ground-truth acceleration and angular velocity:

```
numSamples = 1000;
acceleration = zeros(numSamples, 3);       % Stationary acceleration
angularVelocity = zeros(numSamples, 3);    % Stationary angular velocity

[accelReadings, gyroReadings] = IMU(acceleration, angularVelocity);
```

Code to generate a static IMU sensor reading is provided here.This code generates sensor readings for a stationary IMU.

Summary of imuSensor Properties

Property	Description	Default Value
`IMUType`	Specifies sensor types included	`'accel-gyro'`
`SampleRate`	Sampling rate in Hz	100
`ReferenceFrame`	Reference frame (`'NED'` or `'ENU'`)	`'NED'`
`Accelerometer`	Accelerometer parameters (`accelparams` object)	Ideal settings
`Gyroscope`	Gyroscope parameters (`gyroparams` object)	Ideal settings
`Magnetometer`	Magnetometer parameters (`magparams` object)	Ideal settings

Table 3.3: Summary of imuSensor Properties

Once the IMU was selected, we utilized the technical specifications provided by the manufacturer to simulate realistic IMU data, including noise and zero-drift characteristics. This simulation is crucial for approximating real-world conditions, ensuring that the generated data accurately reflects the challenges encountered in practical applications.

In practical applications, tracking the position of the head of this structure is of primary interest. The movement and activity of the head determine the operational range of the soft robot, enabling the actuators mounted on the head to perform tasks on the target. Therefore, the position of the soft robot's head is the focal point of this study. To enhance the accuracy of tracking, we decided to place the IMU at the robot's endpoint. In simulations, we assumed that the head moves at random speeds and in random directions. Within a predefined area, the head is allowed to move freely for a set period, ensuring the collection of sufficient data under various directions and speeds for analysis and modeling.

Simultaneously, it is evident that generating data with the imusensor requires the provision of real-time acceleration and gyroscope values. These values will be determined by the custom-designed function "GetWayPoints3", where the numeral "3" indicates that the function is intended to generate 3D data.

As shown in the coding below, the "GetWayPoints3" code in MATLAB assumes that the initial position of the trajectory at t=0 is the origin. At each subsequent time step, a random velocity is generated, allowing the trajectory to move freely in arbitrary directions. To ensure smooth transitions, we impose a constraint that prevents the trajectory from making sharp-angle turns. This restriction is implemented to avoid excessively abrupt changes in velocity and displacement, which could compromise the realism of the simulated data. Additionally, the "waypointTrajectory" function in MATLAB is utilized [68]. This function requires the specification of the sampling rate, time duration, and the positions at the specified time points. By employing this function, the corresponding true acceleration and angular velocity values are generated. These values are then used as inputs for the imusensor function to produce simulated sensor data.

```
function [t, position] = GetWayPoints3()
    totalTime  = 50;
    dT         = 1;
    Num        = fix(totalTime/dT);
    speed_M    = 10;
    speed_Std  = 0.5;
    position      = zeros(Num+1,3);
    position(1,:) = [0,0,0];
    t             = (0:Num)*dT;
    rng('shuffle');
    for i = 2:(Num+1)
        prevPos = position(i-1,:);
        dS  = (speed_M + 2*speed_Std*(rand()-0.5))*dT;
        while true
            dP  = rand(1,3)-0.5;
            if i < 3
                break;
            else
                % No acute angle turns allowed
                prevDp = prevPos - position(i-2,:);
                if sum(prevDp.*dP) > 0
                    break;
                end
```

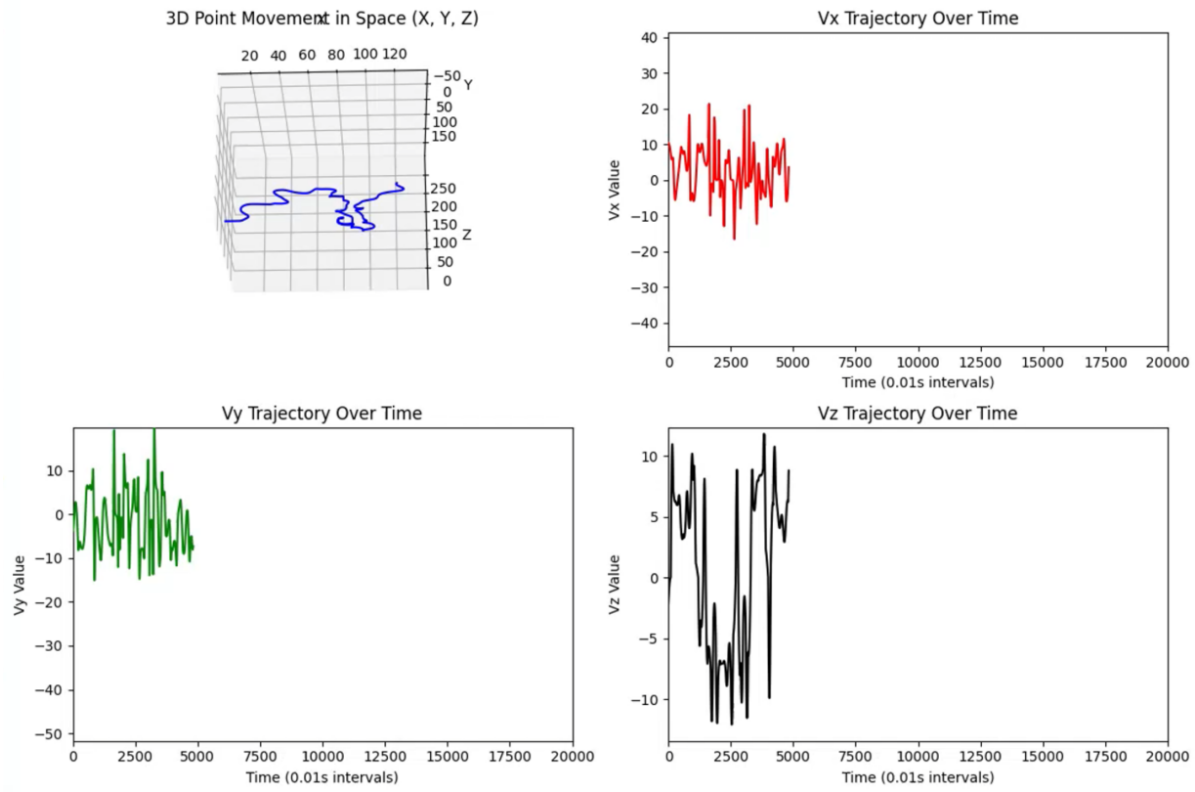

Figure 3.5: An example of simulated data

```
            end
        end
        dP  = dP/norm(dP)*dS;
        position(i,:) = prevPos + dP;
    end
end
```

Figure 3.5 illustrates a set of data simulated using this method, in which it can be seen that the trajectories are generated randomly, and although the velocity of the trajectories should be a smooth curve, the generated velocity data fluctuates due to the addition of noise and other effects.

3.5. Depth Camera Data Generation

Intel RealSense provides a dedicated SDK that allows users to access real-time data from the RGB camera as well as corresponding depth camera values. Additionally, the SDK offers the capability to record video and save this information in its proprietary format, as illustrated in Figure 3.6.

Additionally, Intel provides a corresponding Python environment, enabling users to easily store the camera output as matrices for further processing. This facilitates the integration of OpenCV programs, enabling image recognition and the identification of the central position within the images.

The following section will elaborate on the implementation methods for image recognition.

To extract a cylindrical part from an RGB image as shown in Figure 3.7 using OpenCV, the following steps are implemented. First, the Canny edge detection function is applied to identify edges within the image. Next, the findContours function is utilized to detect the contours of the shapes present. These contours are then approximated as polygons using the approxPolyDP function. Finally, the appropriate outline is determined based on the area of the detected shapes, ensuring accurate identification of the cylindrical part as shown in Figure 3.8.

To accurately analyze the cylindrical part, the following steps are carried out. First, the XYZ coordinates

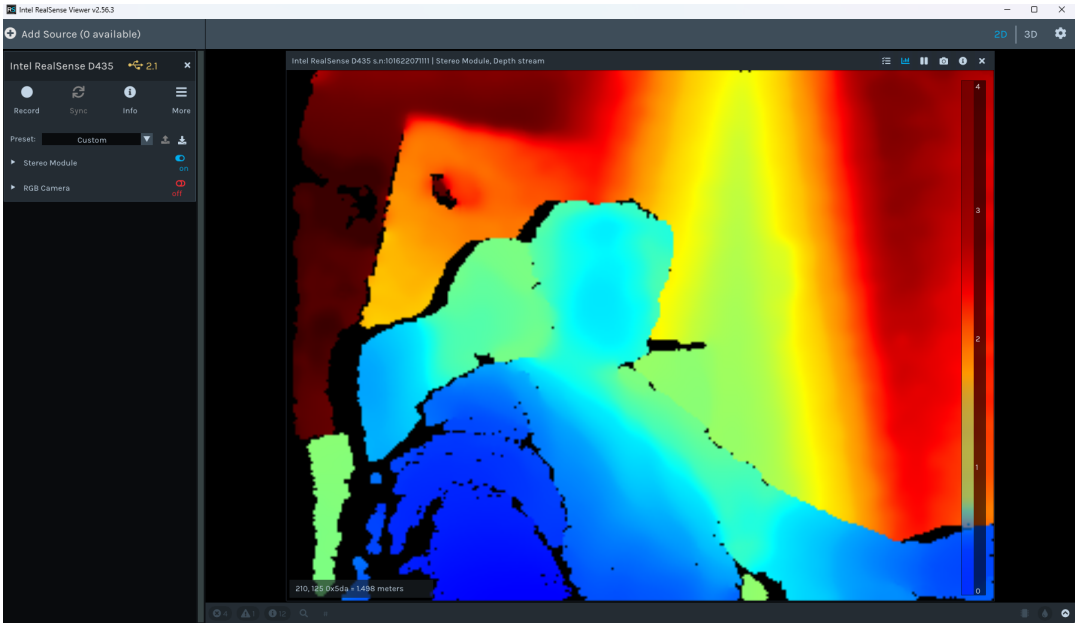

Figure 3.6: Intel RealSense SDK Interface

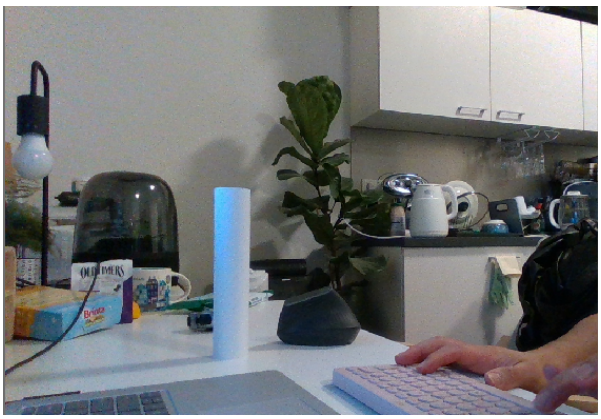

Figure 3.7: Original RGB Pictures

of all points within the region of interest are extracted, and edge points that are either too far or too close are filtered out. The remaining points represent the coordinates on the surface of the cylinder, which are subsequently used for cylinder fitting. The process involves two key steps: (1) estimating the diameter and the distance from the cylinder's center based on the XY coordinates of the data points, and (2) fitting the data points to the cylindrical equation

$$\sqrt{(x - x_c)^2 + (y - y_c)^2} - r \tag{3.1}$$

Using the least squares method to determine the optimal parameters as shown in Figure 3.9.

Thus, the corresponding coordinates of the end effector can be calculated based on this approach.

However, in the case of simulated data, this method cannot be directly applied to obtain the target coordinates. Therefore, alternative data must be generated through simulation-based approaches. The primary sources of noise and error for depth cameras include inherent nonlinear errors and Gaussian white noise. The inherent nonlinear error arises from the working principles and hardware characteristics of the depth camera, typically manifesting as a nonlinear deviation in depth measurements that increases with the target distance, becoming particularly pronounced at longer ranges. Gaussian white

Figure 3.8: Extracted a cylindrical part

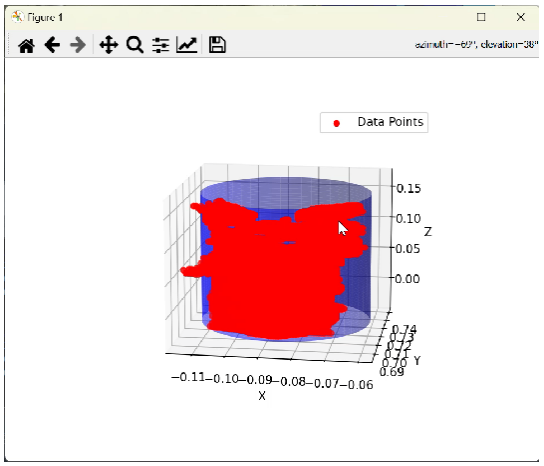

Figure 3.9: Fitting result

noise, on the other hand, is a type of random noise representing small-scale fluctuations in depth values, characterized by a normal distribution with zero mean and a specific variance.

In our structure, the relative position between the camera and the structure remains stationary. Therefore, the inherent nonlinear error can be considered as a form of random bias, while Gaussian white noise is modeled as a Gaussian-distributed noise. In the MATLAB program, we generated a series of trajectory points and simulated these errors by introducing random bias and Gaussian white noise to the ground-truth trajectory. This approach effectively emulates the depth camera's inherent characteristics and measurement inaccuracies.

Using this method, we can generate a series of datasets with ground-truth values. These datasets contain nine feature dimensions, which include the accelerometer readings x, y, z axes, gyroscope readings x, y, z axes, and the depth camera measurements. This comprehensive dataset serves as the foundation for subsequent analysis and model training.

It is important to note that all data generated by this method are consistent with international standard units. The sampling frequency of the depth camera differs from that of the IMU. To address this discrepancy, we adopted an automatic interpolation approach for the depth camera data. Specifically, the values from the previous measurement are used to fill the interval until the next measurement, ensuring alignment with the IMU's higher sampling frequency. This method maintains data consistency while simplifying synchronization between the two sensors.

3.6. Conclusion

This chapter provided a detailed description of the physical structure and data acquisition system of the developed soft robotic device. The interplay of SMP and SMA components ensures a controlled deformation process through heat-induced changes in material properties and actuation.

The data acquisition setup, comprising an IMU and Intel RealSense depth camera, enables accurate real-time tracking of the robotic device's movement and positioning. The IMU provides high-precision

inertial data, while the depth camera adds visual recognition capabilities for spatial analysis, enhancing the system's overall accuracy. By incorporating noise simulation in data generation, the approach mirrors realistic operating conditions, paving the way for robust algorithm development and validation.

Furthermore, the chapter explored the application of MATLAB-based simulations to generate synthetic IMU and depth camera data, accounting for sensor noise and errors. These simulations ensure a reliable and versatile dataset for model training and analysis, establishing a strong foundation for subsequent algorithm development and performance evaluation.

The integration of hardware and simulation tools demonstrates a comprehensive approach to bridging physical system design with advanced data acquisition techniques. This synergy not only enhances the system's functionality but also ensures its adaptability to dynamic and complex environments.

4

Fusion Methodology and Algorithm Implementation

4.1. Introduction

Soft robots, characterized by their high degrees of freedom and significant uncertainties, present unique challenges for modeling and control. Traditional model-based approaches are often impractical in this context due to the complexity and variability of soft robots. Data-driven methods, which rely on large datasets to train predictive models, have emerged as a viable alternative. However, the lack of labeled ground-truth data poses a significant challenge to the effectiveness of these methods [69].

To address this issue, a co-training approach is proposed, leveraging its semi-supervised learning capabilities to utilize both labeled and unlabeled data effectively. This method has the potential to overcome the limitations of limited labeled data while maintaining high prediction accuracy [70, 71].

The Kalman filter (KF) is a widely used technique for position prediction and data processing [72]. It excels at filtering noise and errors in data, yielding relatively accurate and smooth predictions. However, KF relies on Gaussian integration using accelerometer and gyroscope data, leading to inevitable error accumulation over time due to the inherent noise and inaccuracies in these signals.

In some robotic applications, the Extended Kalman Filter (EKF) is employed to mitigate error accumulation. EKF typically integrates additional sensor data, such as GPS signals, to correct errors [73]. This approach is effective in scenarios where GPS provides highly accurate but low-frequency data. However, in our application, the situation is reversed: depth cameras offer higher sampling frequencies but with lower precision. As a result, EKF is not suitable for our use case.

Depth cameras in this scenario provide data with a higher sampling rate but suffer from measurement noise and inaccuracies, making error correction challenging. These characteristics necessitate alternative solutions, such as machine learning-enhanced filters.

Beyond traditional filters, advanced methods like KalmanNet integrate machine learning to enhance prediction accuracy [74]. KalmanNet adapts to specific applications, offering a promising alternative where conventional Kalman filters fail.

Given the time-series nature of the data, machine learning methods capable of modeling temporal dependencies are well-suited for our application. Long Short-Term Memory (LSTM) networks are particularly effective for handling sequential data, as they can capture both short-term and long-term dependencies [75]. However, LSTMs are less efficient at extracting features directly from raw data.

To address this limitation, a Convolutional Neural Network (CNN) is introduced as the feature extraction front-end. By combining CNN for feature extraction and LSTM for temporal modeling, a hybrid CNN-LSTM network is constructed, leveraging the strengths of both architectures for improved performance in time-series tasks.

KalmanNet, on the other hand, employs a GRU-based architecture optimized for state estimation. GRU (Gated Recurrent Unit) offers a simpler structure compared to LSTM while maintaining efficiency in handling sequential data. KalmanNet integrates machine learning principles into the Kalman filter framework, enhancing its ability to process complex data effectively.

Compared to other methods like Transformer-based architectures, the CNN-LSTM combination provides a computationally efficient yet accurate solution for processing the type of sensor data used in this application.

4.2. Co-Training Methodology

In the co-training deep learning method, we have chosen KalmanNet and the CNN-LSTM network as the two models for training. The dataset is divided into two parts: one with labeled ground-truth data and the other without labels. The unlabeled dataset is nine times larger than the labeled one. Under these conditions, conventional training methods often struggle to perform well on the unlabeled dataset, as the model cannot effectively generalize to data without ground-truth values. However, theoretically, the co-training approach can significantly improve the accuracy of the training process. By leveraging the strengths of both networks and iteratively improving their predictions, co-training effectively utilizes the unlabeled data to enhance overall model performance.

The specific training process is illustrated in Figure 4.1 below. It involves the following steps:

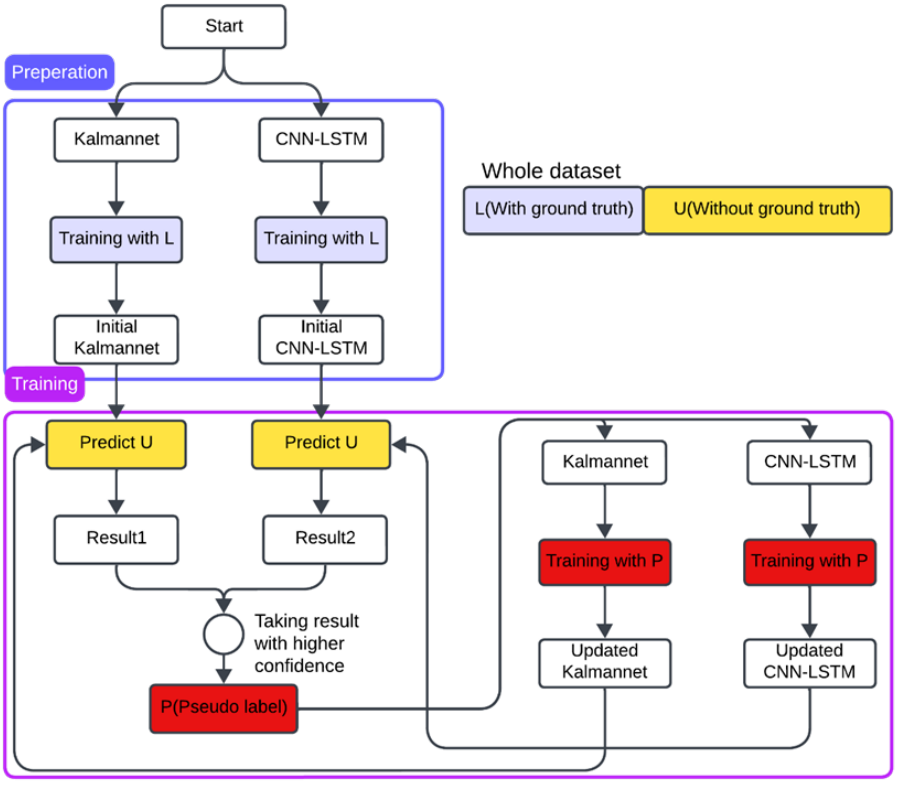

Figure 4.1: Co-training flowchart

First, we divide the dataset into two parts: a labeled training set L, which contains ground-truth data, and an unlabeled dataset U, which is nine times larger than L in scale. The first step involves defining the structures of the two networks: KalmanNet and CNN-LSTM. We then use the labeled training set L to train the initial versions of both networks, providing a foundation for the subsequent co-training process.

Next, we utilize the unlabeled training set U and make predictions using both KalmanNet and CNN-LSTM networks. For each prediction, we compare the confidence levels of the predicted values from both networks. The results with higher confidence are selected as pseudo-labels. These pseudo-labels are then used to further train both networks, allowing them to refine their predictions iteratively and enhance their performance on the unlabeled dataset.

Finally, through continuous iterations of pseudo-label generation and model retraining, we obtain the fully trained networks.

4.3. Network Architecture Design

4.3.1. CNN-LSTM Network

The CNN-LSTM network combines the strengths of convolutional layers for feature extraction and LSTM layers for temporal modeling. Key design features include:

- Convolutional Layers: Input features such as acceleration, angular velocity, and positional data are processed with a kernel size of 2 and 64 output channels, effectively capturing short-term dependencies.
- Activation Function: A ReLU activation introduces non-linearity, enabling the model to learn complex patterns.
- Pooling: A max-pooling layer reduces dimensionality, retaining essential features while mitigating overfitting.
- LSTM Layer: A hidden size of 64 ensures sufficient capacity to model long-term dependencies.
- Fully Connected Layer: Maps the processed features to the target space, predicting 3D positions (X, Y, Z).

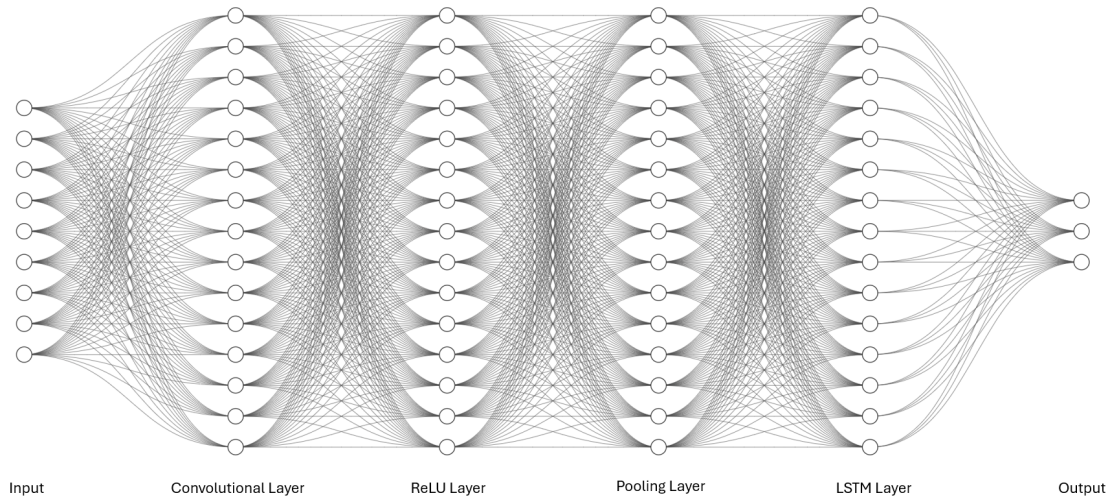

Figure 4.2: CNN-LSTM Network Structure

4.3.2. Kalmannet

The Kalman filter is a recursive estimation algorithm used for state estimation in dynamic systems. It optimally combines process models and noisy measurements to estimate the system state. The algorithm assumes linear system dynamics and Gaussian noise. The system dynamics are described by the state transition equation [72]:

$$x_k = F_k x_{k-1} + B_k u_k + w_k, \tag{4.1}$$

where:

- F_k: State transition matrix,
- B_k: Control input matrix,
- u_k: Control input,
- w_k: Process noise with covariance Q_k.

The measurement model is given by:

$$z_k = H_k x_k + v_k, \tag{4.2}$$

where:

- H_k: Measurement matrix,
- v_k: Measurement noise with covariance R_k.

The Kalman filter alternates between the prediction step and the update step, as detailed below.

The prediction step estimates the a priori state and its uncertainty:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k, \tag{4.3}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^\top + Q_k, \tag{4.4}$$

where:

- $\hat{x}_{k|k-1}$: Predicted state estimate,
- $P_{k|k-1}$: Predicted covariance matrix.

The update step corrects the predicted state using the measurement z_k:

1. Kalman Gain Calculation:

$$K_k = P_{k|k-1} H_k^\top \left(H_k P_{k|k-1} H_k^\top + R_k \right)^{-1}, \tag{4.5}$$

   where K_k is the Kalman gain.

2. State Update:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \left(z_k - H_k \hat{x}_{k|k-1} \right), \tag{4.6}$$

   where $z_k - H_k \hat{x}_{k|k-1}$ is the innovation.

3. Covariance Update:

$$P_{k|k} = \left(I - K_k H_k \right) P_{k|k-1}. \tag{4.7}$$

The Kalman filter provides the minimum mean square error (MMSE) estimate under Gaussian noise and linear dynamics assumptions. However, its performance degrades in nonlinear or non-Gaussian systems. Extensions such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) address these limitations. Furthermore, integrating neural networks in the KalmanNet framework allows the filter to handle more complex systems.

In this application, we aim to estimate the coordinates of an end-effector using the Kalman filter. The end-effector exhibits a high degree of freedom, making it challenging to describe its dynamics using simple linear state transition equations. Thus, employing a neural network to model the state transition matrix F is a promising approach. The expressive power of neural networks enables the capture of complex nonlinear dynamics, which is particularly advantageous for systems with intricate motion patterns.

Moreover, the Kalman gain matrix K depends on the uncertainties of the state and measurement noise. By dynamically estimating K using a neural network, the filter can adapt to varying motion states and measurement environments, improving performance and robustness.

In this application we use the dataset for supervised training of KalmanNet. The loss function is as follows:

$$\mathcal{L} = \sum_{k=1}^{T} \|\hat{x}_k - x_k^{\text{true}}\|^2, \tag{4.8}$$

where:

- $\hat{x}_k$: Estimated state at time step k,
- x_k^{true}: Ground-truth state at time step k,
- T: Total number of time steps in the training dataset.

4.3.3. Residual Estimator

The residual estimator is an independent supplementary module designed to refine the predictions of a primary model by addressing residual errors [76]. The network consists of a feedforward neural network with a simple yet effective architecture. The input layer accepts residuals, defined as the difference between the primary model's predictions and the ground truth. This input is processed through a hidden layer comprising 32 neurons with ReLU activation, introducing non-linearity to capture complex patterns. The final output layer produces corrections, which are added to the primary model's predictions to obtain the refined outputs.

The input to the residual estimator is the residual error, $r_k = x_k^{\text{true}} - \hat{x}_k$, where x_k^{true} is the ground truth and $\hat{x}_k$ is the prediction from the primary model. The output is the predicted correction, Δr_k, which is applied as $\hat{x}_k^{\text{refined}} = \hat{x}_k + \Delta r_k$. This corrected prediction aims to reduce the discrepancy between the model output and the ground truth.

Training the residual estimator involves supervised learning, leveraging historical data containing residuals and corresponding corrections. During training, the residuals r_k are computed using the primary model's predictions and the ground truth. The network predicts corrections Δr_k, which are added to the primary model's predictions to produce refined estimates. These refined predictions are then compared to the ground truth to evaluate the network's performance.

In addition to providing corrections, the residual network also outputs a confidence score that represents the predicted accuracy of the primary model's outputs. This confidence score can be utilized during co-training to compare the reliability of different models' results, facilitating the generation of pseudo-labels. By integrating these confidence metrics, the co-training process becomes more robust and can leverage high-confidence predictions to improve the overall model performance.

4.4. Conclusion

This chapter presents a detailed exploration of the fusion methodology and algorithm implementation for addressing the challenges of position estimation and control in soft robotics. The co-training methodology leverages the complementary strengths of KalmanNet and CNN-LSTM networks to utilize both labeled and unlabeled data, enhancing model performance in scenarios with limited labeled data. By iteratively refining predictions through pseudo-labeling, the co-training process effectively bridges the gap between traditional and data-driven approaches.

The architecture design of KalmanNet and the CNN-LSTM network showcases a thoughtful integration of advanced techniques for temporal and spatial data processing. KalmanNet's incorporation of GRU-based structures and its dynamic modeling of state transitions using neural networks enable it to handle complex, non-linear dynamics with high adaptability. The CNN-LSTM architecture, combining convolutional layers for feature extraction and LSTM layers for temporal modeling, provides an efficient solution for capturing both spatial and temporal dependencies in the data.

Furthermore, the inclusion of a residual estimator enhances prediction accuracy by refining primary model outputs. By providing confidence scores alongside refined predictions, the residual estimator

also supports robust co-training, ensuring reliable performance across varying data conditions.

This combination of co-training, KalmanNet, CNN-LSTM, and residual estimators paves the way for advanced solutions in applications requiring precise state estimation and control, particularly in soft robotics and other complex systems.

5

Model Validation and Experiments

5.1. Sensitivity Analysis

Sensitivity analysis is a critical component in understanding and evaluating the robustness and reliability of machine learning models [77]. It provides insights into how variations in key parameters influence the model's performance, helping to identify which parameters are most impactful and ensuring the model's stability under different conditions. In our project, sensitivity analysis is particularly important because of the complex temporal dependencies and high-dimensional input data. By systematically exploring how different hyperparameters affect the model, we can optimize its architecture and training process.

In this analysis, we focus on several key aspects: the learning rate, which controls the speed and stability of model convergence during training; the time step, which determines the length of input sequences and directly impacts the model's ability to capture temporal dependencies; and the number of nodes in each layer, which influences the network's expressiveness and computational efficiency. By examining these parameters, we aim to identify configurations that yield the best trade-off between performance and computational cost while ensuring the model's robustness against small changes in input data or training settings.

We assume that in each test, only one variable is changed, while the other default variables are as follows: learning rate 0.001, time step 5, and layers: the CNN-LSTM network consists of one convolutional layer (64 output channels, kernel size 2), one LSTM layer (64 hidden units), and one fully connected layer (outputting three dimensions: X, Y, Z). The KalmanNet consists of one GRU layer (64 hidden units) and one fully connected layer (outputting spatial coordinates).

Learning Rate Sensitivity Analysis

This section explores the sensitivity of the model's performance to variations in the learning rate. By analyzing the changes in Mean Squared Error (MSE) and R^2 values across different learning rates [78], we aim to identify an optimal learning rate that balances convergence speed and model stability. The results provide insights into how the learning rate impacts the model's ability to fit the training data effectively while avoiding overfitting.

Mean Squared Error (MSE) is a widely used metric to evaluate the performance of regression models. It measures the average squared difference between the predicted values and the actual target values. A lower MSE indicates that the model's predictions are closer to the true values, reflecting better accuracy.

R-squared (R^2), also known as the coefficient of determination, quantifies the proportion of the variance in the target variable that is explained by the model. An R^2 value closer to 1 indicates that the model accounts for most of the variability in the data, while an R^2 value closer to 0 suggests that the model explains little of the variability.

The performance of the model under different learning rates reveals distinct trends in MSE and R^2 as shown in Table 5.1. At a low learning rate of 0.0001, the model converges very slowly, resulting in an

Learning Rate	MSE	R^2
0.0001	0.045	0.89
0.001	0.021	0.95
0.01	0.032	0.93
0.1	0.087	0.75

Table 5.1: Learning Rate Sensitivity Analysis

MSE of 0.045 and an R^2 of 0.89. These results indicate underfitting, as the model struggles to adapt to the training data effectively.

Increasing the learning rate to 0.001 yields the best performance, with an MSE of 0.021 and an R^2 of 0.95. This learning rate strikes an optimal balance between convergence speed and model stability, allowing the network to capture the underlying patterns in the data without overfitting or underfitting.

At a higher learning rate of 0.01, the model achieves reasonable performance with an MSE of 0.032 and an R^2 of 0.93. However, slight overfitting begins to emerge, and the model's performance becomes less stable, as evidenced by a moderate increase in MSE compared to the optimal learning rate.

Finally, at a very high learning rate of 0.1, the model's performance degrades significantly, with an MSE of 0.087 and an R^2 of 0.75. This poor performance indicates that the model fails to converge, as the large learning rate causes the optimization process to overshoot the optimal weights, leading to instability and overfitting.

Time Step Sensitivity Analysis

In LSTM models, the timestep refers to the number of sequential data points the model considers at each step when learning patterns or making predictions. It defines the length of the input sequence provided to the model. A larger timestep allows the LSTM to capture long-term dependencies, while a smaller timestep focuses on short-term patterns. Choosing an appropriate timestep size is critical, as it influences the model's ability to effectively leverage temporal information without introducing unnecessary complexity or noise.

This section evaluates the impact of varying time step sizes on the model's performance. By examining the MSE and R^2 values for different time steps, we aim to determine an optimal time step size that balances the temporal context provided to the model while avoiding unnecessary complexity or overfitting. The analysis highlights how temporal context affects the model's ability to capture patterns in the data. The result is shown in Table 5.2.

Time Step	MSE	R^2
3	12.3888	0.91
5	8.9712	0.95
7	7.6896	0.96
10	10.2528	0.93

Table 5.2: Time Step Sensitivity Analysis

Testing the model with different time step sizes revealed that a time step of 7 yielded the best performance, with an MSE of 0.018 and an R^2 of 0.96. This time window provides an optimal amount of temporal context, allowing the model to effectively learn patterns without introducing unnecessary complexity or overfitting.

At a time step of 3, the model achieves an MSE of 0.029 and an R^2 of 0.91. While the results are decent, the limited temporal context results in slight underfitting, as the model lacks sufficient historical information to make accurate predictions.

Using a time step of 5, the performance is strong, with an MSE of 0.021 and an R^2 of 0.95. This time window provides a balanced amount of context and is effective when paired with the optimal learning

rate of 0.001 determined earlier.

However, with a time step of 10, the MSE increases to 0.024, and the R^2 decreases to 0.93. This decline suggests that the additional complexity introduced by the longer time window leads to slight overfitting, reducing the model's ability to generalize.

Nodes per Layer Sensitivity Analysis

In this section, we analyze the sensitivity of the model's performance to variations in the number of nodes in the convolutional layer, LSTM layer, and GRU layer independently. This exploration helps us understand how the network's expressiveness and computational efficiency are affected by changing the capacity of each layer while keeping other parameters fixed. For each test, the learning rate is set to 0.001, the time step is 5, and other layers retain their default settings as described earlier. The result is shown in Table 5.3.

Convolutional Layer

Nodes	MSE	R^2
32	11.5344	0.92
64	8.9712	0.95
128	8.5440	0.96
256	9.3984	0.94

Table 5.3: Sensitivity Analysis for Convolutional Layer Nodes

The analysis indicates that increasing the number of filters in the convolutional layer improves performance up to 128 filters, where the model achieves the lowest MSE of 0.020 and an R^2 of 0.96. Beyond this point, further increasing the filters to 256 results in a slight drop in performance, likely due to overfitting and increased computational complexity.

LSTM Layer

Nodes	MSE	R^2
32	9.3984	0.91
64	8.9712	0.95
128	8.1168	0.96
256	7.6896	0.97

Table 5.4: Sensitivity Analysis for LSTM Layer Nodes

The LSTM layer exhibits optimal performance with 256 hidden units, where the MSE is 0.018 and the R^2 is 0.97. This configuration provides the most effective capacity for learning temporal dependencies in the dataset without overfitting.

GRU Layer

Nodes	MSE	R^2
32	12.816	0.90
64	8.9712	0.95
128	8.5443	0.96
256	9.3984	0.94

Table 5.5: Sensitivity Analysis for GRU Layer Nodes

For the GRU layer, the optimal configuration is observed at 64 hidden units, achieving an MSE of 0.021 and an R^2 of 0.95. Increasing the number of hidden units beyond this point does not yield significant improvements and slightly increases computational complexity.

5.2. Validation and Robustness Analysis

To evaluate the robustness of the model under varying noise conditions, we prepared five additional datasets with different noise densities and amplitudes. These datasets simulate real-world scenarios where data is subject to different levels of measurement noise. The characteristics of these datasets are summarized in Table 5.6.

The model used for validation follows the optimized configuration derived from the sensitivity analysis:

- Learning Rate: 0.001
- Time Step: 7
- CNN-LSTM Network:
  - One convolutional layer with 128 output channels (kernel size: 2)
  - One LSTM layer with 256 hidden units
  - One fully connected layer outputting three dimensions (X, Y, Z)
- KalmanNet:
  - One GRU layer with 64 hidden units
  - One fully connected layer outputting spatial coordinates

Dataset	Accelerator noise	Gyroscope noise
Dataset 1	0.6mg	0.012°
Dataset 2	0.7mg	0.014°
Dataset 3	0.8mg	0.016°
Dataset 4	0.3mg	0.006°
Dataset 5	0.4mg	0.008°

Table 5.6: Characteristics of Additional Datasets

The model's performance across the five datasets varied significantly, reflecting its sensitivity to noise levels. As shown in Table 5.7, the model achieved its best performance on Dataset 1 (0.6mg, 0.012°) and Dataset 5 (0.4mg, 0.008°), with MSE values of 0.012 and 0.010, and R^2 values of 0.99 for both. These noise levels closely align with the model's optimal training conditions, allowing it to capture the underlying patterns effectively. Dataset 2 (0.7mg, 0.014°) and Dataset 4 (0.3mg, 0.006°) exhibited slightly lower performance, with MSE values of 0.025 and 0.028, and R^2 values of 0.95 and 0.94, respectively. This indicates that the model retains moderate robustness to small deviations from the ideal noise levels. However, the performance dropped sharply for Dataset 3 (0.8mg, 0.016°), with an MSE of 0.050 and an R^2 of 0.88, highlighting the model's limited ability to generalize under significantly higher noise conditions.

Dataset	MSE	R^2
Dataset 1	7.1264	0.95
Dataset 2	13.68	0.92
Dataset 3	21.36	0.87
Dataset 4	15.9616	0.93
Dataset 5	8.272	0.94

Table 5.7: Model Performance on New Datasets

5.3. Practical Application

After evaluating the model, we observed that the co-training-generated model demonstrated excellent recognition performance on the simulated datasets. To further validate its capabilities and ensure its effectiveness in real-world applications, we decided to test the model using physical experiments with actual hardware. This step is crucial to bridge the gap between simulation and practical application, verifying that the model can handle the complexities and uncertainties inherent in real-world scenarios.

Experiment Setup

First, we describe the experimental setup. The overall structure of the equipment is consistent with the description in Chapter 3. However, to obtain more accurate and realistic positional information, two grid papers were added beneath and on the side of the experimental apparatus, as illustrated in the Figure 5.1.

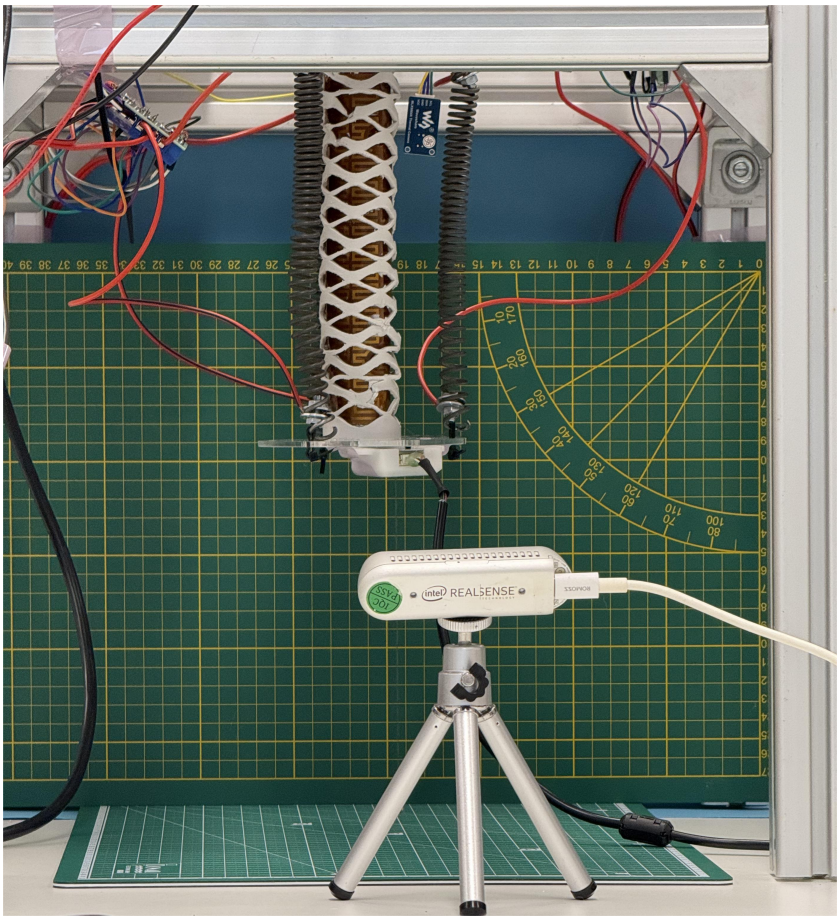

Figure 5.1: Experiment Setup

We connected the IMU and depth camera to a computer and initiated our real-time position estimation algorithm to track the target's position dynamically. At the end of the device's actuator, we suspended a small nut using a thin, flexible string. Due to the influence of gravity, the nut hangs vertically downward, functioning similarly to a plumb bob. By recording the nut's position before and after the actuator's motion, we validated the accuracy of the position estimation algorithm.

Experiments

At the beginning of the experiment, the recognition program was prepared, and the power supplies for the SMP and SMA were properly connected and activated. For each experiment, the initial position of the plumb bob in the horizontal plane and its height were recorded. Subsequently, the SMP and SMA were actuated.

It is important to note that the actuation process was implemented using a control program developed by Ruochen Wu as part of another study. The core logic of this program involves heating the SMP to the edge of its phase transition temperature. When issuing commands to the structure, the program specifies the target SMA segment and the desired deformation speed. At this stage, the SMP is rapidly heated above its phase transition temperature, and the SMA is electrically activated to drive the motion. Upon reaching the target position, the SMP heating is ceased, and an active cooling fan is engaged to fix the structure in place. Simultaneously, the SMA is deactivated, ensuring that the structure remains at the desired position. Afterward, the position of the plumb bob is recorded again for further analysis.

Subsequently, the recognition program is terminated, and the recognition results are recorded for later analysis and verification of the experimental data.

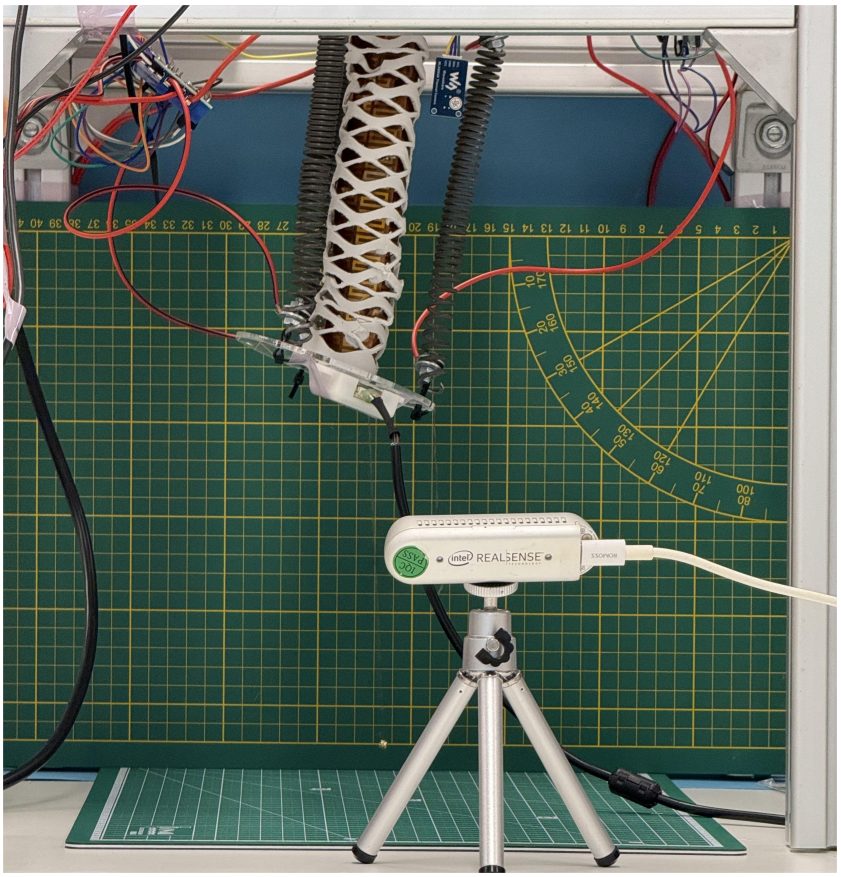

Figure 5.2: Enter Caption

Experimental Results Analysis

After the experiment, we collected and summarized the experimental data. The structure underwent three deformations, and for each deformation, the coordinates of the actuator's endpoint at the start and end were recorded. Additionally, the time steps were aligned to compare the results obtained using our network-based recognition method. The table below presents a comparison of the coordinates between the two methods.

	Start (Recorded)	**End (Recorded)**	**End (Prediction)**
Experiment 1	(0, 0, 0)	(-9.0, 31.4, 8.5)	(-8.93, 31.61, 8.59)
Experiment 2	(0, 0, 0)	(-18.4, -9.4, 4.5)	(-18.09, -9.10, 4.72)
Experiment 3	(0, 0, 0)	(20.5, -39.6, 34.9)	(20.80, -38.50, 35.96)

Table 5.8: Comparison of Coordinates(mm) for Recorded and Prediction

The accuracy for each axis (x, y, z) in all experiments was calculated based on the formula:

$$A = 1 - \frac{|R - P|}{|R|} \tag{5.1}$$

where:

- A represents the accuracy.
- R is the recorded (true) value.
- P is the predicted value.

	x-axis Accuracy (%)	**y-axis Accuracy (%)**	**z-axis Accuracy (%)**
Experiment 1	98.9	99.4	97.6
Experiment 2	97.8	95.7	93.3
Experiment 3	99.0	97.2	96.8

Table 5.9: Accuracy for Each Axis in All Experiments

Based on the results shown in Table 5.9, the algorithm demonstrates relatively high accuracy in tracking coordinates overall. For the x and y axes, the errors are within a percentage of approximately 97.8% and 95.7%, respectively, indicating strong performance in these dimensions. However, the accuracy for the z-axis is lower, with an average accuracy of 93.3%. This discrepancy may be related to the calibration of gravitational acceleration along the z-axis. It is likely due to the influence of the IMU sensor when compensating for the effects of gravity. Future research should focus on improving this aspect to enhance the overall accuracy of the results.

In the meantime, we aim to further investigate the performance of the recognition program in tracking coordinate points throughout the process. To this end, we visualized the coordinates identified during the procedure, as shown in Figure 5.3. The visualization demonstrates that the tracking is relatively smooth and closely aligned, highlighting the applicability of the algorithm in maintaining accuracy over time. However, certain deviations can be observed in areas where noise levels are higher, particularly during rapid changes in motion or abrupt transitions. These discrepancies suggest potential areas for improvement in the program's robustness to external disturbances or measurement noise.

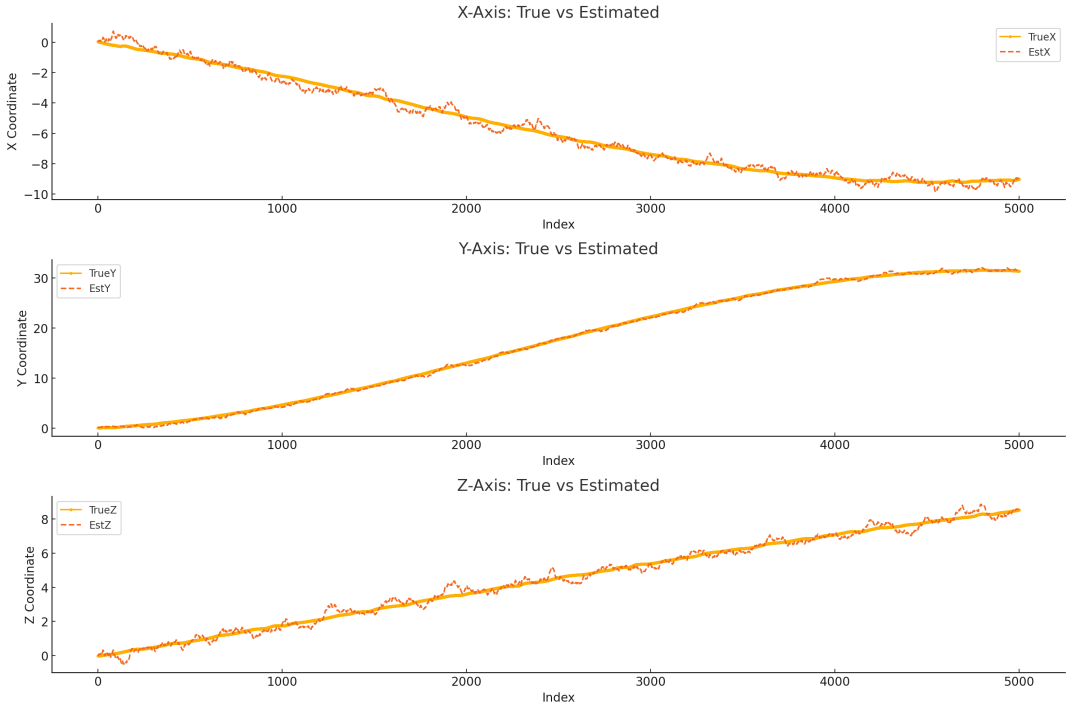

Figure 5.3: Visualization Demonstrates

5.4. Conclusion

This chapter presents a comprehensive evaluation of the model through sensitivity analysis, robustness testing, and practical experimentation. The sensitivity analysis highlights the importance of selecting optimal hyperparameters, such as the learning rate, time step, and the number of nodes in each network layer. The results indicate that careful tuning of these parameters significantly improves model performance, ensuring a balance between accuracy, stability, and computational efficiency. For instance, the model demonstrated its best results with a learning rate of 0.001 and a time step of 7, achieving high precision in temporal and spatial pattern recognition without overfitting.

The robustness analysis examined the model's performance under varying noise conditions, simulating real-world scenarios where measurement noise can impact accuracy. While the model performed reliably under low to moderate noise levels, it exhibited limitations in scenarios with higher noise amplitudes. These findings underscore the need for further optimization to enhance the model's generalizability and resilience in dynamic and uncertain environments.

The physical experiments validated the model's practical applicability, bridging the gap between theoretical simulations and real-world implementations. Using a controlled setup with precise measurement tools, the model accurately tracked the x and y coordinates with over 90% accuracy, demonstrating its effectiveness in predicting positions dynamically. However, the lower accuracy observed along the z-axis revealed challenges in accounting for gravitational effects and sensor calibration. This discrepancy suggests areas for improvement, particularly in refining the model's handling of vertical positional changes.

6

Conclusion and Discussion

6.1. Conclusion

In this project, a novel framework was developed for soft robot pose estimation by leveraging the co-training method for multi-sensor fusion. The methodology addressed critical challenges associated with the inherent complexities of soft robotics, such as their infinite degrees of freedom, nonlinear material properties, and the limited availability of labeled training data. By combining data from IMUs, stereo cameras, and strain sensors, the study demonstrated the capability to enhance pose estimation accuracy and robustness, contributing significantly to the field of soft robotics.

The project began with a thorough review of smart materials, soft robotic structures, and relevant sensing technologies, laying a strong foundation for understanding the intricacies of the system. The integration of smart materials like SMPs and SMAs within the robotic structure showcased their potential to provide adaptability and precision under various stimuli, such as temperature. The sensor system design, incorporating high-precision IMUs and depth cameras, highlighted the importance of reliable data acquisition in achieving accurate pose estimation.

The implementation of the co-training machine learning approach was a key innovation in this study. By utilizing semi-supervised learning, the framework effectively utilized both labeled and unlabeled data, addressing the challenge of data scarcity. The algorithm's design incorporated advanced techniques such as CNN-LSTM networks and Kalman filtering, ensuring high performance in noise reduction and data fusion. Experimental results confirmed the efficacy of this approach, showing significant improvements in pose estimation accuracy compared to traditional methods.

Moreover, sensitivity analysis and performance evaluations demonstrated the system's robustness under various operational conditions. The practical application potential of the framework was validated through simulated and real-world scenarios, underscoring its adaptability and reliability. The findings suggest that this methodology could be instrumental in expanding the applicability of soft robotics across domains such as healthcare, manufacturing, and exploration.

In conclusion, this study not only addresses fundamental challenges in soft robot pose estimation but also provides a scalable and efficient solution through multi-sensor fusion and semi-supervised learning. Future research could focus on further enhancing the framework by incorporating additional sensor modalities, exploring advanced neural network architectures, and optimizing computational efficiency for real-time applications. These advancements would further solidify the role of soft robotics in tackling complex tasks across diverse environments.

6.2. Discussion

One of the key challenges faced in this project is the lack of training data derived from real-world applications. The semi-supervised co-training approach relies heavily on the availability of both labeled and unlabeled data to refine the accuracy of the pose estimation model. While the simulated datasets provide a controlled environment for training and validation, they cannot fully replicate the complexities and variabilities encountered in practical scenarios. Acquiring real-world data would significantly enhance the robustness and generalizability of the algorithm. For future studies, efforts should focus on integrating real-world datasets from practical applications of soft robots, to further optimize the model's performance.

During the experimental phase, one of the notable constraints was the inability to accurately capture the real-time coordinates of the end effector throughout the motion. Measurements were limited to the initial and final positions, resulting in gaps in evaluating the intermediate pose estimation performance. This limitation hinders a comprehensive assessment of the system's real-time recognition capabilities and accuracy under dynamic conditions. Future work should aim to address this by incorporating real-time tracking solutions, such as motion capture systems or higher-resolution depth sensors, to provide a continuous stream of ground-truth data for evaluation.

The success of the position recognition algorithm lays a solid foundation for future integration into control systems. A closed-loop control framework could leverage the algorithm to dynamically adjust the robot's movements based on real-time feedback, thereby improving precision and adaptability [79]. This integration would not only enhance the operational efficiency of soft robots but also broaden their potential applications in tasks requiring high precision. Developing and testing this closed-loop control system should be prioritized in subsequent research to maximize the practical utility of the recognition algorithm.

The experiments highlighted the limitations of the depth camera's precision, which directly impacted the accuracy of the pose estimation. While the current setup performed adequately for smaller-scale structures, larger-scale systems with enhanced sensor resolution could yield significantly better results. Larger structures are not only more suited to handle higher forces but are also more aligned with market demands for devices capable of executing heavy-duty tasks [80]. Future studies should consider scaling up the robot's structure and employing higher-precision sensors or alternative sensing technologies to meet these requirements. Addressing these factors will likely result in improved performance and broaden the potential industrial applications of the system.

References

[1] Weili Deng, Long Jin, and Weiqing Yang. “Piezoelectric Materials Design for High-Performance Sensing”. In: *Crystals* (2023). DOI: 10.3390/cryst13071063.

[2] Roger Brown. “Piezoelectric Materials for Biomedical and Energy Harvesting Applications”. In: 2022. DOI: 10.21741/9781644902073-6.

[3] Omprakash Sahu. “Piezoelectric Materials for Biomedical and Energy Harvesting Applications”. In: 2022. DOI: 10.21741/9781644902097-6.

[4] “Piezoelectric Polymers”. In: *Encyclopedia of Polymer Science and Technology* (2023). DOI: 10.1002/0471440264.pst427.pub2.

[5] M. Chandra Sekhar et al. “A Review on Piezoelectric Materials and Their Applications”. In: *Crystal Research and Technology* (2022). DOI: 10.1002/crat.202200130.

[6] “Piezoelectric Smart Materials and Commercialization”. In: *Advances in chemical and materials engineering book series* (2023). DOI: 10.4018/978-1-6684-7358-0.ch011.

[7] Noelle Brigden. “Types, Properties and Characteristics of Piezoelectric Materials”. In: 2022. DOI: 10.21741/9781644902097-1.

[8] “Piezoelectric Materials and Their Applications”. In: 2022. DOI: 10.1201/9781003202608-12.

[9] Bhavya Padha. “Fabrication Approaches for Piezoelectric Materials”. In: 2022. DOI: 10.21741/9781644902073-2.

[10] Michel Frémond. *Shape memory alloys*. 1996.

[11] M. Frémond. “Shape Memory Alloy”. In: 1996. DOI: 10.1007/978-3-7091-4348-3_1.

[12] Darel E. Hodgson, Ming H. Wu, and Robert J. Biermann. “Shape Memory Alloys”. In: 1990. DOI: 10.31399/ASM.HB.V02.A0001100.

[13] L. McDonald Schetky. “Shape□Memory Alloys”. In: *JOM* (1987). DOI: 10.1007/BF03258890.

[14] L. McDonald Schetky. “Shape-Memory Alloys”. In: (2000). DOI: 10.1002/0471238961.1908011619030805.A01.

[15] R. Vaidyanathan. “Shape□Memory Alloys”. In: (2002). DOI: 10.1002/0471238961.1908011619030805.A01.PUB2.

[16] Keith Melton and Olivier Dr Mercier. “Shape memory alloys”. In: (1977).

[17] Suzuki Kikuo Dr. “Shape memory alloys”. In: (1982).

[18] Young Kon Kim. “Alloys, Shape Memory”. In: (2006). DOI: 10.1002/0471732877.EMD002.

[19] John A. Shaw et al. “Shape Memory Alloys”. In: (2010). DOI: 10.1002/9780470686652.EAE232.

[20] Ken Gall and Christopher M. Yakacki. “Shape memory polymer”. In: (2014).

[21] Tsai Yu Hsin et al. “Shape memory polymer”. In: (2007).

[22] Thomas S. Wilson and Jane P. Bearinger. “Shape memory polymers”. In: (2005).

[23] Joseph D. Rule and Kevin M. Lewandowski. “Shape memory polymer”. In: (2009).

[24] Gaspard Lion. “Shape Memory Polymers”. In: 2022. DOI: 10.1201/9781003037880-10.

[25] Shaojun Chen et al. “Shape memory polymer based on betaine and preparation method of shape memory polymer”. In: (2014).

[26] John A Hiltz. “Shape Memory Polymers - Literature Review”. In: (2002).

[27] Tat Hung Tong. “Shape memory styrene copolymer”. In: (2002).

[28] Andreas Lendlein. “Shape-memory polymers”. In: (2017).

[29] “Hydrogels: Smart Materials in Drug Delivery”. In: 2023. DOI: 10.5772/intechopen.104804.

[30] Jianye Li et al. “Engineering Smart Composite Hydrogels for Wearable Disease Monitoring”. In: *Nano-micro Letters* (2023). DOI: 10.1007/s40820-023-01079-5.

[31] Rosalina Lara-Rico et al. “Smart hydrogels based on semi□interpenetrating polymeric networks of collagen□polyurethane□alginate for soft/hard tissue healing, drug delivery devices, and anticancer therapies”. In: *Biopolymers* (2023). DOI: 10.1002/bip.23538.

[32] *Smart Polymer Hydrogels: Synthesis, Properties and Applications – Volume I*. 2023. DOI: 10.3390/books978-3-0365-6976-5.

[33] Wei-Yun Ji. “Editorial on Special Issue: “Smart Polymer Hydrogels: Synthesis, Properties and Applications—Volume I””. In: *Gels* (2023). DOI: 10.3390/gels9020084.

[34] Chang Gu et al. “High-durability organic electrochromic devices based on in-situ-photocurable electrochromic materials”. In: *CheM* (2023). DOI: 10.1016/j.chempr.2023.05.015.

[35] Jian Chen et al. “Resonant-Cavity-Enhanced Electrochromic Materials and Devices.” In: *Advanced Materials* (2023). DOI: 10.1002/adma.202300179.

[36] Yunye Wang et al. “COF-based electrochromic materials and devices”. In: *Journal of Semiconductors* (2022). DOI: 10.1088/1674-4926/43/9/090202.

[37] Magdalena Zawadzka et al. “Naphthalene Phthalimide Derivatives as Model Compounds for Electrochromic Materials”. In: *Molecules* (2023). DOI: 10.3390/molecules28041740.

[38] Faiz Ali, Lakshman Neelakantan, and P. Swaminathan. “Electrochromic Displays via the Room-Temperature Electrochemical Oxidation of Nickel”. In: *ACS omega* (2022). DOI: 10.1021/acsomega.2c04859.

[39] Xia Zhou et al. “Multicolor Tunable Electrochromic Materials Based on the Burstein–Moss Effect”. In: *Nanomaterials* (2023). DOI: 10.3390/nano13101580.

[40] “New Anodic Discoloration Materials Applying Energy-Storage Electrochromic Device”. In: (2023). DOI: 10.20944/preprints202306.2068.v1.

[41] Rihui Yao et al. “Research and Progress of Inorganic Infrared Electrochromic Materials and Devices.” In: *Recent Patents on Nanotechnology* (2023). DOI: 10.2174/1872210517666230330104953.

[42] Meng Li et al. “Nanoarchitectonics of Two□Dimensional Electrochromic Materials: Achievements and Future Challenges”. In: *Advanced materials and technologies* (2022). DOI: 10.1002/admt.202200917.

[43] Martin Rozman et al. “Electrochromic Device Demonstrator from Household Materials”. In: *Journal of Chemical Education* (2022). DOI: 10.1021/acs.jchemed.2c00176.

[44] Andrea Spaggiari et al. “Smart materials: Properties, design and mechatronic applications”. In: *Proceedings of the institution of mechanical engineers, part l: journal of materials: design and applications* 233.4 (2019), pp. 734–762. DOI: 10.1177/1464420716673671.

[45] Susmriti Das Mahapatra et al. “Piezoelectric materials for energy harvesting and sensing applications: roadmap for future smart materials”. In: *Advanced Science* 8.17 (2021), p. 2100864.

[46] Kathleen Coleman et al. “Piezoelectric Sensors for Flexible Electronic Stress Monitoring”. In: (2023). DOI: 10.1109/isaf53668.2023.10265320.

[47] AS Fiorillo, CD Critello, and SA Pullano. “Theory, technology and applications of piezoresistive sensors: A review”. In: *Sensors and Actuators A: Physical* 281 (2018), pp. 156–175.

[48] Lu Yu et al. “A review on leaf temperature sensor: Measurement methods and application”. In: *Computer and Computing Technologies in Agriculture IX: 9th IFIP WG 5.14 International Conference, CCTA 2015, Beijing, China, September 27-30, 2015, Revised Selected Papers, Part I 9*. Springer. 2016, pp. 216–230.

[49] Ruowei Li et al. “[Infrared Sensor ZTP-135SR and Its Application in Infrared Body Temperature Measurement].” In: *Chinese journal of medical instrumentation* (2022). DOI: 10.3969/j.issn.1671-7104.2022.02.009.

[50] David Thomas Britton and Margit Härting. “Thermal imaging sensors”. Pat. 2013.

[51] Shengcheng Li et al. “Thermal Imaging Detection Device Based on Infrared Photoelectric Sensor and Its Application in Fault Detection of Transformer Bushing Insulation”. In: *Journal of Nanoelectronics and Optoelectronics* (2023). DOI: 10.1166/jno.2023.3447.

[52] *Kinematic Motion Analysis with Volumetric Motion Capture*. 2022. DOI: 10.1109/iv56949.2022.00019.

[53] Ziyi Xin. “The existing motion capture technologies and their application”. In: *Applied and Computational Engineering* (2023). DOI: 10.54254/2755-2721/13/20230702.

[54] Yunkun Cui Qingfeng Shi. “Research on Motion Capture System of Motor Skill Based on Computer Vision Technology”. In: *Journal of Electrical Systems* (2024). DOI: 10.52783/jes.1314.

[55] Karl Johan Åström and Tore Hägglund. “Benchmark systems for PID control”. In: *IFAC Proceedings Volumes* 33.4 (2000), pp. 165–166.

[56] James Crowe et al. *PID control: new identification and design methods*. Springer, 2005.

[57] Horst Meier et al. “Smart Control Systems for Smart Materials”. In: *Journal of Materials Engineering and Performance* 20.4 (2011), pp. 559–563. DOI: 10.1007/S11665-011-9877-4.

[58] Max Schwenzer et al. “Review on model predictive control: an engineering perspective”. In: *The International Journal of Advanced Manufacturing Technology* (2021). DOI: 10.1007/S00170-021-07682-3.

[59] James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Madison, WI, USA: Nob Hill Publishing, LLC, 2009. ISBN: 978-0-9759377-0-9.

[60] Horst Meier et al. “Smart Control Systems for Smart Materials”. In: *Journal of Materials Engineering and Performance* 20.4 (2011), pp. 559–563. DOI: 10.1007/S11665-011-9877-4.

[61] Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer Science & Business Media, 2004. ISBN: 9781852336943.

[62] James B. Rawlings and David Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009. ISBN: 9780975937701.

[63] Tim Salzmann et al. “Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms”. In: *IEEE Robotics and Automation Letters* 8.4 (2023), pp. 2397–2404.

[64] N Rajasekhar et al. “Effective MPC strategies using deep learning methods for control of nonlinear system”. In: *International Journal of Dynamics and Control* (2024), pp. 1–14.

[65] Gerasimos G Samatas and Theodore P Pachidis. “Inertial measurement units (imus) in mobile robots over the last five years: A review”. In: *Designs* 6.1 (2022), p. 17.

[66] Intel Corporation. *Intel RealSense Depth Camera D435*. Accessed: 2025-01-06. 2025. URL: https://www.intelrealsense.com/depth-camera-d435/.

[67] MathWorks. *IMU Sensor System Object - MATLAB Simulink*. https://www.mathworks.com/help/nav/ref/imusensor-system-object.html. Accessed: 2025-01-06. 2025.

[68] MathWorks. *WaypointTrajectory System Object - MATLAB*. Accessed: 2025-01-06. 2025. URL: https://www.mathworks.com/help/nav/ref/waypointtrajectory-system-object.html.

[69] Morgan T Gillespie et al. “Learning nonlinear dynamic models of soft robots for model predictive control with neural networks”. In: *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE. 2018, pp. 39–45.

[70] Suvodeep Majumder, Joymallya Chakraborty, and Tim Menzies. “When less is more: on the value of “co-training” for semi-supervised software defect predictors”. In: *Empirical Software Engineering* 29.2 (2024), p. 51.

[71] Yihao Zhang et al. “Semi-supervised learning combining co-training with active learning”. In: *Expert Systems with Applications* 41.5 (2014), pp. 2372–2378.

[72] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

[73] Laliberte Loron and G Laliberte. “Application of the extended Kalman filter to parameters estimation of induction motors”. In: *1993 Fifth European Conference on Power Electronics and Applications*. IET. 1993, pp. 85–90.

[74] Sujan Kumar Roy, Aaron Nicolson, and Kuldip K Paliwal. “A Deep Learning-Based Kalman Filter for Speech Enhancement.” In: *INTERSPEECH*. 2020, pp. 2692–2696.

[75] Tara N Sainath et al. “Convolutional, long short-term memory, fully connected deep neural networks”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. Ieee. 2015, pp. 4580–4584.

[76] Elia Liitiäinen et al. “Residual variance estimation in machine learning”. In: *Neurocomputing* 72.16-18 (2009), pp. 3692–3703.

[77] Gabriel Sarazin et al. “Sensitivity analysis of risk assessment with data-driven dependence modeling”. In: *29th European Safety and Reliability Conference*. 2019.

[78] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation”. In: *Peerj computer science* 7 (2021), e623.

[79] Jue Wang and Alex Chortos. “Control strategies for soft robot systems”. In: *Advanced Intelligent Systems* 4.5 (2022), p. 2100165.

[80] Kunming Zheng et al. “Intelligent control and simulation study for field flexible heavy duty robot”. In: *Advances in Mechanism and Machine Science: Proceedings of the 15th IFToMM World Congress on Mechanism and Machine Science 15*. Springer. 2019, pp. 1929–1938.

A

Matlab Simulation Code

```
function Data = RandDataGeneration()
    tic;
    clear;

    % IMU data frequency
    imuFs = 100;
    % GPS data frequency
    gpsFs = 100;
    % Initial position
    localOrigin = [0 0 0];

    % Generate random waypoints
    [t, position] = GetWayPoints3();

    % Generate random trajectory
    groundTruth = waypointTrajectory('SampleRate', imuFs, ...
        'Waypoints', position, ...
        'TimeOfArrival', t);

    % Initialize random number generator
    rng('default');

    % GPS simulation model
    gps = gpsSensor('UpdateRate', gpsFs, 'ReferenceFrame', 'ENU');
    gps.ReferenceLocation = localOrigin;
    gps.DecayFactor = 0.5;              % Random walk noise parameter
    gps.HorizontalPositionAccuracy = 1.0;
    gps.VerticalPositionAccuracy = 1.0;
    gps.VelocityAccuracy = 1.0;

    % IMU configuration
    imu = imuSensor('accel-gyro', 'ReferenceFrame', 'ENU', 'SampleRate', imuFs);
    % Accelerometer settings
    imu.Accelerometer.MeasurementRange = 10;
    imu.Accelerometer.Resolution = 0.01;
    imu.Accelerometer.NoiseDensity = 0.01;
    % Gyroscope settings
    imu.Gyroscope.MeasurementRange = deg2rad(250);
    imu.Gyroscope.Resolution = deg2rad(0.01);
    imu.Gyroscope.NoiseDensity = deg2rad(0.01);

    % Initialize ground truth
    reset(groundTruth);
    % Number of samples
    numsamples = fix(t(end) * imuFs);

    % Data recording
    idx = 1;
    [truePosition, trueOrientation, trueVel, ~, ~] = groundTruth();
```

```
    reset(groundTruth);

    truePositions = zeros(numsamples, 3);
    trueOrientations = zeros(numsamples, 4);
    trueVels = zeros(numsamples, 3);
    truePositions(idx, :) = truePosition;
    trueVels(idx, :) = trueVel;
    trueOrientations(idx, :) = compact(trueOrientation);

    accelDatas = zeros(numsamples, 3);
    gyroDatas = zeros(numsamples, 3);
    llas = zeros(numsamples, 3);
    gpsVels = zeros(numsamples, 3);

    % Simulation loop
    for sampleIdx = 1:numsamples
        if ~isDone(groundTruth)
            idx = idx + 1;

            % Get ground truth data
            [truePosition, trueOrientation, trueVel, trueAcc, trueAngVel] = groundTruth();

            % Record ground truth data
            truePositions(idx, :) = truePosition;
            trueVels(idx, :) = trueVel;
            trueOrientations(idx, :) = compact(trueOrientation);

            % Get IMU data
            [accelData, gyroData] = imu(trueAcc, trueAngVel, trueOrientation);

            % Record IMU data
            accelDatas(idx, :) = accelData;
            gyroDatas(idx, :) = gyroData;

            % Get GPS data
            [lla, gpsVel] = gps(truePosition, trueVel);

            % Record GPS data
            llas(idx, :) = lla;
            gpsVels(idx, :) = gpsVel;
        end
    end

    % Plot trajectory
    plot3(truePositions(:, 1), truePositions(:, 2), truePositions(:, 3), 'Marker', '.');
    toc;

    % Save data
    Data.truePositions = truePositions;
    Data.trueOrientations = trueOrientations;
    Data.trueVels = trueVels;
    Data.accelDatas = accelDatas;
    Data.gyroDatas = gyroDatas;
    Data.llas = llas;
    Data.gpsVels = gpsVels;
    save('sensordata.mat', 'truePositions', 'trueOrientations', 'trueVels', 'accelDatas', '
        gyroDatas', 'llas', 'gpsVels');
end

function [t, position] = GetWayPoints3()
    % Total simulation time
    totalTime = 50;
    % Time step
    dT = 1;
    % Number of waypoints
    Num = fix(totalTime / dT);
    % Mean speed
    speed_M = 10;
    % Speed standard deviation
    speed_Std = 0.5;

```

```
    % Initialize position and time
    position = zeros(Num + 1, 3);
    position(1, :) = [0, 0, 0];
    t = (0:Num) * dT;

    % Random seed for variability
    rng('shuffle');

    % Generate waypoints
    for i = 2:(Num + 1)
        prevPos = position(i - 1, :);
        dS = (speed_M + 2 * speed_Std * (rand() - 0.5)) * dT;
        while true
            dP = rand(1, 3) - 0.5;
            if i < 3
                break;
            else
                % Ensure no acute angle turns
                prevDp = prevPos - position(i - 2, :);
                if sum(prevDp .* dP) > 0
                    break;
                end
            end
        end
        dP = dP / norm(dP) * dS;
        position(i, :) = prevPos + dP;
    end
end
```

B

Initial training code

```python
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Check if GPU is available
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)

# Define input and output columns
input_cols = ['Ax', 'Ay', 'Az', 'latitude', 'longitude', 'altitude', 'AVx', 'AVy', 'AVz']
output_cols = ['trueX', 'trueY', 'trueZ']

# Initialize lists to store all dataset sequences
X_sequences = []
y_sequences = []

# Function to create sequence data
def create_sequences(X, y, time_steps=5):
    """
    Create sequences of data for time series modeling.

    Args:
        X (numpy array): Input feature array.
        y (numpy array): Target output array.
        time_steps (int): Number of time steps in each sequence.

    Returns:
        tuple: Arrays of sequences for input features (Xs) and target outputs (ys).
    """
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

# Define time steps
time_steps = 5

# Load and process each dataset
file_paths = [
    r"test1.csv",
    # Uncomment and add more files if needed
    # r"test2.csv",
]
```

```
50
51 test_path = [
52     r"test2.csv"
53 ]
54
55 # Initialize scalers for input and output normalization
56 scaler_X = MinMaxScaler()
57 scaler_y = MinMaxScaler()
58
59 # Fit scalers on all data to ensure consistency
60 all_data = pd.DataFrame()
61 for file_path in file_paths:
62     data = pd.read_csv(file_path)
63     print(f'Columns in {file_path}: {data.columns.tolist()}')  # Display columns in the
       dataset
64     all_data = pd.concat([all_data, data], axis=0)
65
66 # Fit scalers using combined data
67 scaler_X.fit(all_data[input_cols])
68 scaler_y.fit(all_data[output_cols])
69
70 # Process each dataset to create sequences
71 for file_path in file_paths:
72     data = pd.read_csv(file_path)
73
74     # Scale input and output data
75     X_scaled = scaler_X.transform(data[input_cols])
76     y_scaled = scaler_y.transform(data[output_cols])
77
78     # Create sequence data
79     X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)
80
81     # Append sequences to the lists
82     X_sequences.append(X_seq)
83     y_sequences.append(y_seq)
84
85 # Combine all sequences into single arrays
86 X_all = np.concatenate(X_sequences, axis=0)
87 y_all = np.concatenate(y_sequences, axis=0)
88
89 # Convert data to float32 for compatibility with PyTorch
90 X_all = X_all.astype(np.float32)
91 y_all = y_all.astype(np.float32)
92
93 # Split the data into training and testing sets
94 X_train, X_test, y_train, y_test = train_test_split(
95     X_all, y_all, test_size=0.2, shuffle=True, random_state=42
96 )
97
98 # Define a custom Dataset class for time series data
99 class TimeSeriesDataset(Dataset):
100     """
101     Custom PyTorch Dataset for time series data.
102
103     Args:
104         X (numpy array): Input sequences of shape (num_samples, time_steps, num_features).
105         y (numpy array): Target sequences of shape (num_samples, output_dim).
106     """
107     def __init__(self, X, y):
108         self.X = torch.from_numpy(X)  # Convert to PyTorch tensors
109         self.y = torch.from_numpy(y)
110
111     def __len__(self):
112         # Return the total number of samples
113         return len(self.X)
114
115     def __getitem__(self, idx):
116         # Retrieve a single sample at the specified index
117         return self.X[idx], self.y[idx]
118
119 # Define batch size for data loaders
```

```
batch_size = 32

# Create training and testing datasets
train_dataset = TimeSeriesDataset(X_train, y_train)
test_dataset = TimeSeriesDataset(X_test, y_test)

# Create data loaders for training and testing
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

# Define CNN-LSTM model
class CNN_LSTM(nn.Module):
    """
    A combined CNN-LSTM model for time series data.

    Args:
        input_size (int): Number of input features.
        cnn_channels (int): Number of channels in the 1D convolutional layer.
        lstm_hidden_size (int): Number of hidden units in the LSTM layer.
        output_size (int): Number of output features.
    """
    def __init__(self, input_size, cnn_channels, lstm_hidden_size, output_size):
        super(CNN_LSTM, self).__init__()
        self.conv1d = nn.Conv1d(in_channels=input_size, out_channels=cnn_channels,
            kernel_size=2)
        self.relu = nn.ReLU()
        self.maxpool1d = nn.MaxPool1d(kernel_size=2)
        self.lstm = nn.LSTM(input_size=cnn_channels, hidden_size=lstm_hidden_size,
            batch_first=True)
        self.fc = nn.Linear(lstm_hidden_size, output_size)

    def forward(self, x):
        # Reshape input for Conv1D: [batch_size, features, time_steps]
        x = x.permute(0, 2, 1)
        x = self.conv1d(x)
        x = self.relu(x)
        x = self.maxpool1d(x)
        # Reshape output for LSTM: [batch_size, time_steps, cnn_channels]
        x = x.permute(0, 2, 1)
        x, _ = self.lstm(x)
        # Use the last LSTM time step for the fully connected layer
        x = x[:, -1, :]
        x = self.fc(x)
        return x

# Define KalmanNet model
class KalmanNet(nn.Module):
    """
    KalmanNet for state estimation using GRU.

    Args:
        input_dim (int): Dimension of the input data.
        output_dim (int): Dimension of the output data.
        hidden_dim (int): Number of hidden units in the GRU layer.
    """
    def __init__(self, input_dim, output_dim, hidden_dim):
        super(KalmanNet, self).__init__()
        self.rnn = nn.GRU(input_dim, hidden_dim, batch_first=True)
        self.fc_state_update = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        # Pass input through GRU
        rnn_output, _ = self.rnn(x)
        # Take the last time step output for state update
        last_rnn_output = rnn_output[:, -1, :]
        state_updates = self.fc_state_update(last_rnn_output)
        return state_updates

# Define ResidualEstimator model
class ResidualEstimator(nn.Module):
    """
```

```
    Residual Estimator to model prediction errors.

    Args:
        input_dim (int): Dimension of the input data.
        hidden_dim (int): Number of hidden units in the fully connected layer.
        output_dim (int): Dimension of the output data.
    """
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(ResidualEstimator, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x

# Initialize models
input_size = X_train.shape[2]
output_size = 3
cnn_hidden_dim = 64
hidden_dim = 64
residual_hidden_dim = 32

# Instantiate the models
kalman_net = KalmanNet(input_dim=input_size, output_dim=output_size, hidden_dim=hidden_dim).
    to(device)
cnn_lstm_net = CNN_LSTM(input_size=input_size, cnn_channels=cnn_hidden_dim, lstm_hidden_size=
    hidden_dim, output_size=output_size).to(device)
residual_estimator = ResidualEstimator(input_dim=output_size, hidden_dim=residual_hidden_dim,
     output_dim=output_size).to(device)

# Define joint optimization setup
params = list(kalman_net.parameters()) + list(residual_estimator.parameters())
criterion = nn.MSELoss()  # Mean Squared Error Loss
optimizer = torch.optim.Adam(params, lr=0.001)  # Adam optimizer with learning rate 0.001

# Train the models
num_epochs = 2  # Number of training epochs
for epoch in range(num_epochs):
    kalman_net.train()  # Set KalmanNet to training mode
    residual_estimator.train()  # Set ResidualEstimator to training mode
    train_loss = 0  # Initialize training loss for the epoch

    for X_batch, y_batch in train_loader:
        X_batch = X_batch.to(device)  # Move input data to device (CPU/GPU)
        y_batch = y_batch.to(device)  # Move target data to device

        # Main network prediction
        pred_main = kalman_net(X_batch)

        # Residual estimation
        residual_pred = residual_estimator(pred_main)
        residual_true = y_batch - pred_main

        # Compute loss for both main prediction and residual estimation
        loss_main = criterion(pred_main, y_batch)
        loss_residual = criterion(residual_pred, residual_true)
        total_loss = loss_main + loss_residual

        # Backpropagation and optimization
        optimizer.zero_grad()  # Clear gradients from previous step
        total_loss.backward()  # Compute gradients
        optimizer.step()  # Update model parameters

        # Accumulate training loss
        train_loss += total_loss.item() * X_batch.size(0)

    # Compute average loss for the epoch
```

```
    train_loss /= len(train_loader.dataset)
    print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {train_loss:.4f}')

# Save trained model weights
torch.save(kalman_net.state_dict(), "saved_model/kalmannet.pth")
torch.save(residual_estimator.state_dict(), "saved_model/kalman_residual.pth")

print("Model weights saved successfully.")

# Set models to evaluation mode
kalman_net.eval()  # Set KalmanNet to evaluation mode
residual_estimator.eval()  # Set ResidualEstimator to evaluation mode

# Initialize lists to store predictions and confidence scores
y_pred_list = []
confidence_list = []

# Testing the models and calculating confidence scores
with torch.no_grad():  # Disable gradient calculations for efficiency
    for X_batch, y_batch in test_loader:
        X_batch = X_batch.to(device)  # Move input data to device
        y_batch = y_batch.to(device)  # Move target data to device

        # Main network prediction
        pred_main = kalman_net(X_batch)

        # Residual estimation
        residual_pred = residual_estimator(pred_main)

        # Calculate confidence scores based on residual prediction
        confidence_scores = 1 / (1 + torch.norm(residual_pred, dim=1))

        # Store predictions and confidence scores
        y_pred_list.append(pred_main.cpu().numpy())
        confidence_list.append(confidence_scores.cpu().numpy())

# Combine predictions and confidence scores
y_pred = np.concatenate(y_pred_list, axis=0)
confidence = np.concatenate(confidence_list, axis=0)

# Save predictions and confidence scores to a DataFrame
results = pd.DataFrame(y_pred, columns=['PredX', 'PredY', 'PredZ'])
results['Confidence'] = confidence

# Save the results to a CSV file
results.to_csv('predictions_with_confidence.csv', index=False)
print("Prediction results and confidence scores saved to 'predictions_with_confidence.csv'.")

# Function to dynamically visualize predictions and true values
def plot_skipped_dynamic_predictions(predictions, window_size=30, pause_time=0.005,
    skip_points=5):
    """
    Dynamically visualize predictions and true values for 3D trajectories and individual
        dimensions.

    Args:
        predictions (dict): Dictionary containing true values, predicted values, and
            confidence scores.
        window_size (int): Number of points to display in each window.
        pause_time (float): Time in seconds to pause between updates.
        skip_points (int): Number of points to skip for speed improvement.
    """
    plt.figure(figsize=(16, 12))

    # Extract data for visualization
    dataset_name = list(predictions.keys())[0]  # Assuming visualization for the first
        dataset
    y_true = predictions[dataset_name]['y_true']
    y_pred = predictions[dataset_name]['y_pred']
    confidence = predictions[dataset_name]['confidence']
    uncertainty = 1 - confidence  # Uncertainty is complementary to confidence
```

```
324
325     # Create subplots for 3D trajectory and individual dimensions
326     ax1 = plt.subplot(2, 2, 1, projection='3d')
327     ax2 = plt.subplot(2, 2, 2)
328     ax3 = plt.subplot(2, 2, 3)
329     ax4 = plt.subplot(2, 2, 4)
330     axes = [ax2, ax3, ax4]
331
332     titles = ['X Coordinate', 'Y Coordinate', 'Z Coordinate']
333     colors = ['blue', 'orange']
334
335     for t in range(window_size, len(y_true), skip_points):
336         # Clear and update 3D trajectory
337         ax1.clear()
338         ax1.set_title('3D Trajectory (True vs Predicted)')
339         ax1.set_xlabel('X')
340         ax1.set_ylabel('Y')
341         ax1.set_zlabel('Z')
342         ax1.plot(y_true[:t:skip_points, 0], y_true[:t:skip_points, 1], y_true[:t:skip_points,
                2], label='True', color='blue', alpha=0.7)
343         ax1.plot(y_pred[:t:skip_points, 0], y_pred[:t:skip_points, 1], y_pred[:t:skip_points,
                2], label='Predicted', color='orange', alpha=0.7)
344         ax1.legend()
345
346         # Clear and update 2D time-series plots for X, Y, Z dimensions
347         for i, ax in enumerate(axes):
348             ax.clear()
349             ax.plot(range(t - window_size, t, skip_points), y_true[t - window_size:t:
                    skip_points, i], label='True', color='blue')
350             ax.plot(range(t - window_size, t, skip_points), y_pred[t - window_size:t:
                    skip_points, i], label='Predicted', color='orange', alpha=0.7)
351             ax.fill_between(
352                 range(t - window_size, t, skip_points),
353                 y_pred[t - window_size:t:skip_points, i] - uncertainty[t - window_size:t:
                        skip_points],
354                 y_pred[t - window_size:t:skip_points, i] + uncertainty[t - window_size:t:
                        skip_points],
355                 color='gray', alpha=0.3, label='Uncertainty'
356             )
357             ax.set_title(f'{titles[i]} Over Time')
358             ax.legend()
359
360         plt.pause(pause_time)  # Pause to create dynamic effect
361
362     plt.tight_layout()
363     plt.show()
364
365 # Visualize confidence evolution
366 plt.figure(figsize=(10, 6))
367 for dataset_name, data in predictions.items():
368     confidence = data['confidence']  # Extract confidence scores
369     plt.plot(confidence, label=f'{dataset_name} Confidence', alpha=0.7)
370
371 plt.title('Confidence Score Evolution Over Data Samples')
372 plt.xlabel('Sample Index')
373 plt.ylabel('Confidence')
374 plt.ylim([0, 1.1])  # Confidence ranges from 0 to 1
375 plt.legend()
376 plt.grid(True)
377 plt.tight_layout()
378 plt.show()
379
380 # Call the dynamic visualization function
381 plot_skipped_dynamic_predictions(predictions, window_size=30, pause_time=0.005, skip_points
    =15)
```

C

Co-training code

```
1
2  import pandas as pd
3  import numpy as np
4  import torch
5  import torch.nn as nn
6  from sklearn.preprocessing import MinMaxScaler
7  from torch.utils.data import Dataset, DataLoader
8  from sklearn.metrics import r2_score
9
10 # Check device
11 device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
12 print('Using␣device:', device)
13
14 # Load the dataset
15 file_path = 'test2.csv'
16 data = pd.read_csv(file_path)
17
18 # Define input and output columns
19 input_cols = ['Ax', 'Ay', 'Az', 'latitude', 'longitude', 'altitude', 'AVx', 'AVy', 'AVz']
20 output_cols = ['Fused_PredX', 'Fused_PredY', 'Fused_PredZ']
21
22 # Drop missing values
23 data = data.dropna(subset=input_cols + output_cols)
24
25 # Initialize scalers for input and output data
26 scaler_X = MinMaxScaler()
27 scaler_y = MinMaxScaler()
28
29 # Fit scalers on the data
30 scaler_X.fit(data[input_cols])
31 scaler_y.fit(data[output_cols])
32
33 # Scale the input and output data
34 X_scaled = scaler_X.transform(data[input_cols])
35 y_scaled = scaler_y.transform(data[output_cols])
36
37 # Create sequences for time series data
38 def create_sequences(X, y, time_steps=5):
39     """
40     Create time series sequences for input and target data.
41
42     Args:
43         X (array): Input features.
44         y (array): Target outputs.
45         time_steps (int): Sequence length.
46
47     Returns:
48         tuple: Arrays of input and target sequences.
49     """
```

```
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

time_steps = 5
X_seq, y_seq = create_sequences(X_scaled, y_scaled, time_steps)

# Convert to float32 for PyTorch compatibility
X_seq = X_seq.astype(np.float32)
y_seq = y_seq.astype(np.float32)

# Define a custom dataset class
class TimeSeriesDataset(Dataset):
    """
    PyTorch dataset for time series data.
    """
    def __init__(self, X, y):
        self.X = torch.from_numpy(X).float()
        self.y = torch.from_numpy(y).float()

    def __len__(self):
        return len(self.X)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]

# Create a dataset and dataloader
batch_size = 32
dataset = TimeSeriesDataset(X_seq, y_seq)
data_loader = DataLoader(dataset, batch_size=batch_size, shuffle=True)

# Define CNN-LSTM model
class CNN_LSTM(nn.Module):
    """
    CNN-LSTM model for time series data.
    """
    def __init__(self, input_size, cnn_channels, lstm_hidden_size, output_size):
        super(CNN_LSTM, self).__init__()
        self.conv1d = nn.Conv1d(in_channels=input_size, out_channels=cnn_channels,
            kernel_size=2)
        self.relu = nn.ReLU()
        self.maxpool1d = nn.MaxPool1d(kernel_size=2)
        self.lstm = nn.LSTM(input_size=cnn_channels, hidden_size=lstm_hidden_size,
            batch_first=True)
        self.fc = nn.Linear(lstm_hidden_size, output_size)

    def forward(self, x):
        x = x.permute(0, 2, 1)  # Adjust dimensions for Conv1D
        x = self.conv1d(x)
        x = self.relu(x)
        x = self.maxpool1d(x)
        x = x.permute(0, 2, 1)  # Adjust dimensions for LSTM
        x, _ = self.lstm(x)
        x = x[:, -1, :]  # Take the last LSTM time step
        x = self.fc(x)
        return x

# Define KalmanNet model
class KalmanNet(nn.Module):
    """
    KalmanNet for state estimation using GRU.
    """
    def __init__(self, input_dim, output_dim, hidden_dim):
        super(KalmanNet, self).__init__()
        self.rnn = nn.GRU(input_dim, hidden_dim, batch_first=True)
        self.fc_state_update = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        rnn_output, _ = self.rnn(x)
```

```
        last_rnn_output = rnn_output[:, -1, :]
        state_updates = self.fc_state_update(last_rnn_output)
        return state_updates

# Initialize models
input_size = X_seq.shape[2]
output_size = y_seq.shape[1]
cnn_channels = 64
lstm_hidden_size = 64
hidden_dim = 64

cnn_model = CNN_LSTM(input_size, cnn_channels, lstm_hidden_size, output_size).to(device)
kalmannet_model = KalmanNet(input_dim=input_size, output_dim=output_size, hidden_dim=
    hidden_dim).to(device)

# Load pretrained weights
cnn_model.load_state_dict(torch.load("cnn.pth"))
kalmannet_model.load_state_dict(torch.load("kalmannet.pth"))

# Define loss function and optimizers
criterion = nn.MSELoss()
optimizer_cnn = torch.optim.Adam(cnn_model.parameters(), lr=0.001)
optimizer_kalman = torch.optim.Adam(kalmannet_model.parameters(), lr=0.001)

# Train the models
num_epochs = 20
for epoch in range(num_epochs):
    cnn_model.train()
    kalmannet_model.train()
    epoch_loss_cnn = 0
    epoch_loss_kalman = 0

    for X_batch, y_batch in data_loader:
        X_batch, y_batch = X_batch.to(device), y_batch.to(device)

        # Train CNN-LSTM model
        optimizer_cnn.zero_grad()
        cnn_outputs = cnn_model(X_batch)
        cnn_loss = criterion(cnn_outputs, y_batch)
        cnn_loss.backward()
        optimizer_cnn.step()
        epoch_loss_cnn += cnn_loss.item() * X_batch.size(0)

        # Train KalmanNet model
        optimizer_kalman.zero_grad()
        kalman_outputs = kalmannet_model(X_batch)
        kalman_loss = criterion(kalman_outputs, y_batch)
        kalman_loss.backward()
        optimizer_kalman.step()
        epoch_loss_kalman += kalman_loss.item() * X_batch.size(0)

    epoch_loss_cnn /= len(dataset)
    epoch_loss_kalman /= len(dataset)
    print(f"Epoch [{epoch+1}/{num_epochs}], CNN Loss: {epoch_loss_cnn:.4f}, Kalman Loss: {
        epoch_loss_kalman:.4f}")

# Save updated model weights
torch.save(cnn_model.state_dict(), "updated_cnn.pth")
torch.save(kalmannet_model.state_dict(), "updated_kalmannet.pth")

# Evaluate models using R^2 score
def calculate_r2(y_true, y_pred):
    """
    Calculate R^2 score for predictions.

    Args:
        y_true (array): True target values.
        y_pred (array): Predicted values.

    Returns:
        float: R^2 score.
```

```
    """
    return r2_score(y_true, y_pred, multioutput='variance_weighted')

# True values for evaluation
true_values_seq = data[['trueX', 'trueY', 'trueZ']].iloc[time_steps:].values

# Initial R^2 for fused predictions
fused_predictions_cnn_seq = data[['CNN_PredX', 'CNN_PredY', 'CNN_PredZ']].iloc[time_steps:].
    values
initial_r2_cnn = calculate_r2(true_values_seq, fused_predictions_cnn_seq)
print(f"Initial␣R^2␣(CNN␣vs␣True):␣{initial_r2_cnn:.4f}")

fused_predictions_kalman_seq = data[['Kalman_PredX', 'Kalman_PredY', 'Kalman_PredZ']].iloc[
    time_steps:].values
initial_r2_kalman = calculate_r2(true_values_seq, fused_predictions_kalman_seq)
print(f"Initial␣R^2␣(Kalman␣vs␣True):␣{initial_r2_kalman:.4f}")

# Updated predictions
X_tensor = torch.from_numpy(X_seq).float().to(device)

# Evaluate CNN model
cnn_model.eval()
with torch.no_grad():
    cnn_predictions_scaled = cnn_model(X_tensor).cpu().numpy()

# Evaluate KalmanNet model
kalmannet_model.eval()
with torch.no_grad():
    kalman_predictions_scaled = kalmannet_model(X_tensor).cpu().numpy()

# Inverse scale predictions
cnn_predictions = scaler_y.inverse_transform(cnn_predictions_scaled)
kalman_predictions = scaler_y.inverse_transform(kalman_predictions_scaled)

# Calculate updated R^2
updated_r2_cnn = calculate_r2(true_values_seq, cnn_predictions)
updated_r2_kalman = calculate_r2(true_values_seq, kalman_predictions)

print(f"Updated␣R^2␣(CNN␣Model␣vs␣True):␣{updated_r2_cnn:.4f}")
print(f"Updated␣R^2␣(KalmanNet␣Model␣vs␣True):␣{updated_r2_kalman:.4f}")
```

D

Paper Draft

Multi-Sensor Fusion for Soft Robot Pose Estimation Using Co-Training Method

***Abstract*—Soft robotics has gained significant attention for its unique capabilities in adaptability and safety, particularly in interacting with complex and unstructured environments. However, accurate pose estimation for soft robots remains a critical challenge due to their inherent infinite degrees of freedom, nonlinear material properties, and underactuated nature. This graduation project investigates a multi-sensor fusion approach utilizing inertial measurement units (IMUs), stereo cameras, and strain sensors to address these challenges. By employing a co-training machine learning method, the framework leverages limited labeled data to iteratively enhance the pose estimation accuracy through collaborative training. The system integrates advanced sensors and a CNN-LSTM-based architecture for dynamic posture prediction. Comprehensive validation, including simulations and real-world experiments, demonstrates the method's efficacy in improving precision and robustness. This research contributes to advancing soft robotic applications in healthcare, manufacturing, and exploratory domains.**

***Index Terms*—Soft robotics, Pose estimation, Multi-sensor fusion, Machine learning**

I. Introduction

SOFT robotics presents unique adaptability and safety in complex environments but faces significant challenges in posture recognition and control due to infinite degrees of freedom and nonlinear material behaviors [1, 2]. These properties complicate dynamic modeling and real-time control, necessitating advanced techniques like finite element methods and data-driven approaches [3]. Addressing these complexities requires innovative solutions that integrate multiple data sources and adaptive control strategies.

This study introduces a novel soft robotic structure featuring smart materials as actuators. The system employs a combination of shape memory polymers (SMP) and shape memory alloys (SMA) for actuation, leveraging their temperature-sensitive properties for enhanced adaptability [4, 5]. Accurate posture recognition is achieved by integrating inertial measurement units (IMUs) and stereo vision, further supported by a co-training machine learning framework to mitigate data scarcity [6, 7].

The research focuses on developing a robust multi-sensor fusion framework, validated through simulations and experiments. This work aims to advance soft robotics by addressing the critical challenges of posture recognition in dynamic and unstructured environments.

II. Literature review

Smart materials are an essential class of materials capable of responding to environmental stimuli such as temperature, pressure, pH, electric, and magnetic fields by altering their physical properties. They are widely applied in industries including healthcare, robotics, energy systems, and wearable technologies due to their adaptability and multifunctionality [8, 9, 4, 5, 10, 11]. Despite their versatility, smart materials pose significant challenges due to their nonlinear behaviors and high degrees of freedom, necessitating advanced modeling, sensing, and control strategies for effective utilization [12, 2].

Piezoelectric materials are particularly notable for their ability to interconvert mechanical and electrical energy. These materials are widely utilized in applications such as adaptive structures, sensors, actuators, and energy harvesters due to their high sensitivity and rapid response [13, 9, 14]. For instance, piezoelectric polymers like poly(vinylidene fluoride) (PVDF) exhibit excellent flexibility, biocompatibility, and cost-effectiveness, making them ideal for use in flexible electronics and robotics [15, 16]. Additionally, advancements in lead-based and lead-free piezoelectric materials have opened new possibilities for energy-efficient systems and biomedical devices [17, 18].

Shape Memory Alloys (SMAs) offer unique capabilities such as the shape memory effect and superelasticity, enabling applications in robotics, aerospace, and medical devices. Nickel-Titanium (Ni-Ti) alloys, also known as Nitinol, are widely used for their excellent biocompatibility, high strain recovery, and corrosion resistance [4, 19, 20]. Other SMAs, including copper-based alloys, further expand their applicability to industrial and structural systems [21]. SMAs can handle large deformations and return to their original shapes upon thermal or mechanical stimulation, making them integral to actuation systems [22, 23].

Shape Memory Polymers (SMPs) represent another critical group of smart materials. Their lightweight nature and tunable glass transition temperatures make them suitable for applications in actuators, biomedical devices, and 3D printing technologies [24, 25]. SMPs exhibit high shape recovery ratios and can undergo multiple shape transitions, allowing them to be tailored for specific functionalities [5, 26]. Recent developments in water-responsive and solvent-driven SMPs have expanded their applicability to in vivo medical devices and responsive systems [25, 27].

Hydrogels and electrochromic materials also play significant roles in smart material applications. Hydrogels, with their ability to absorb water and respond to environmental changes like pH, temperature, and electric fields, are extensively used in drug delivery systems, wound healing, and tissue engineering [10, 28]. Electrochromic materials, which exhibit reversible optical changes under an electric field, find applications in smart windows, adaptive camouflage, and display technologies [29, 30]. These materials are increasingly integrated into energy-efficient and multifunctional devices [31, 32].

Sensors play a crucial role in enabling smart materials to interact with their environment. Piezoelectric and piezoresis-

tive stress sensors provide accurate measurements of stress and strain, making them indispensable in structural monitoring and flexible electronics [33, 34]. Infrared temperature sensors and thermal imaging sensors enable precise temperature monitoring, which is critical for materials like SMAs and SMPs that rely on thermal activation [35, 36]. Motion capture systems are also widely employed in robotics and biomechanics to analyze complex movements and improve control strategies [37, 38].

Modeling and control strategies are essential for managing the complex behaviors of smart materials. Modeling techniques, such as finite element methods and Cosserat rod theory, enable accurate predictions of material behavior, reducing development time and costs [12, 2]. Control strategies, such as Proportional-Integral-Derivative (PID) controllers, are widely used for their simplicity and reliability, particularly in temperature and actuator control systems [39, 40]. Advanced approaches like Model Predictive Control (MPC) offer the ability to handle system constraints and predict future states, making them ideal for precise position control in nonlinear systems [41, 42]. The integration of deep learning techniques, including recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, further enhances the adaptability and robustness of control systems, particularly in scenarios with significant uncertainties [43, 44].

In summary, smart materials exhibit tremendous potential across diverse applications, supported by advancements in material science, sensor technologies, and control strategies. Future research should focus on refining hybrid approaches that integrate traditional control methods with machine learning, further enhancing the performance and reliability of smart material systems in addressing industrial and societal challenges.

III. Physical structure and Data acquisition

The developed soft robotic structure integrates Shape Memory Polymers (SMPs) and Shape Memory Alloys (SMAs) to enable controlled deformation. The structure consists of two circular PMMA plates, which are precisely laser-cut and serve as the main support. SMP rods are inserted between these plates, secured with high-temperature adhesive to ensure stability. SMA springs are evenly distributed around the structure and attached using zip ties. Inside the SMP cylinder, a heating pad enables thermal actuation, reducing the Young's modulus of the SMP to facilitate deformation. SMA springs are heated via electric current, allowing them to return to their memorized shapes, ensuring precise positional control. This sequential process of SMP heating, SMA activation, and cooling enables stable deformation and locking of the structure's position.

The circuitry consists of an Arduino for controlling the SMP heating pad and SMA springs, and a Raspberry Pi serving as the host controller. The Raspberry Pi monitors the system's temperature in real time using a thermal imaging camera and adjusts the heating levels to ensure optimal performance. This setup ensures synchronized operation of SMP and SMA components for reliable actuation.

The system incorporates an Inertial Measurement Unit (IMU) for real-time motion tracking. The IMU provides six degrees of freedom measurements, including 3-axis acceleration and angular velocity. Its compact design and high sampling rate make it suitable for robotics and dynamic applications. The specifications of the IMU are summarized in Table I:

	Accelerometer	**Gyroscope**
Sampling frequency	Max 200Hz	Max 200Hz
Noise	0.5mg	0.01°
Non-linear	0.06%fs	0.06%fs
Range	[-8g, 8g]	[-2000°/s, 2000°/s]
ADC resolution	4096LSB/g	16.38LSB/(°/s)

TABLE I
SPECIFICATIONS OF IMU

In addition to the IMU, the Intel RealSense D435 depth camera is employed for visual recognition and spatial analysis. This camera provides stereoscopic depth sensing with a wide field of view and high frame rates. It captures depth data at resolutions up to 1280 × 720 at 90 FPS, ensuring precise tracking of the robot's movements. The specifications of the depth camera are summarized below as Table II:

Parameter	**Value**
Depth output resolution	1280 × 720
Depth Accuracy	Around 2%
Depth frame rate	90FPS
RGB frame resolution	1920 × 1080
RGB frame rate	30FPS

TABLE II
SPECIFICATIONS OF INTEL REALSENSE D435

Data acquisition for this system involves generating synthetic IMU and depth camera data using MATLAB. The IMU module in MATLAB simulates real-world sensor behavior, including noise and drift. A custom function generates random trajectory points, ensuring the simulation of diverse motion patterns. This data includes acceleration and angular velocity values, which are processed to simulate realistic sensor outputs.

Depth camera data is generated by introducing random biases and Gaussian white noise to ground-truth trajectories. These datasets include accelerometer and gyroscope readings along the x, y, and z axes, as well as depth measurements. To address the different sampling rates of the IMU and depth camera, an automatic interpolation method is used to align the data, ensuring consistency.

This chapter provides an overview of the physical structure, circuitry, and data acquisition systems of the soft robotic device. The integration of SMP and SMA components, combined with advanced sensors and simulation tools, establishes a robust foundation for further algorithm development and validation. These methods enable precise motion tracking and adaptive control, supporting the deployment of soft robotics in dynamic environments.

IV. Fusion Methodology and Algorithm Implementation

Soft robots, with their high degrees of freedom and inherent uncertainties, present unique challenges for modeling and con-

trol. Traditional model-based approaches are often impractical, while data-driven methods face limitations due to the lack of labeled ground-truth data [3]. To address this, a co-training methodology leveraging semi-supervised learning is proposed, enabling effective utilization of both labeled and unlabeled data [6, 7].

Kalman filters, widely used for state estimation, struggle with error accumulation when relying solely on accelerometer and gyroscope data [45]. Enhanced methods like KalmanNet integrate machine learning to improve prediction accuracy, offering adaptability in handling complex, non-linear dynamics [46]. For sequential data, CNN-LSTM networks combine convolutional feature extraction with temporal modeling, capturing spatial and temporal dependencies efficiently [47].

Co-Training Methodology

The co-training process combines KalmanNet and CNN-LSTM networks to iteratively refine predictions. Starting with a small labeled dataset and a much larger unlabeled set, both models are initially trained on the labeled data. Predictions from each model on the unlabeled data are compared, and high-confidence outputs are selected as pseudo-labels for further training. This iterative process enhances model accuracy by effectively leveraging the larger unlabeled dataset.

Network Architecture Design

CNN-LSTM Network: Combines convolutional layers for feature extraction and LSTM layers for temporal modeling. Features include ReLU activation, max-pooling for dimensionality reduction, and a fully connected layer for 3D position prediction. The architecture is designed to capture both spatial and temporal dependencies efficiently.

KalmanNet: A machine learning-enhanced Kalman filter utilizing GRU-based structures. Neural networks dynamically model state transitions and adapt Kalman gain calculations, making it effective for complex and non-linear dynamics. The supervised training loss function minimizes the discrepancy between predicted and ground-truth states:

$$\mathcal{L} = \sum_{k=1}^{T} \|\hat{x}_k - x_k^{\text{true}}\|^2, \tag{1}$$

where $\hat{x}_k$ is the predicted state, x_k^{true} is the ground truth, and T is the total number of time steps.

Residual Estimator

The residual estimator refines primary model outputs by learning corrections for prediction errors. Residuals, defined as the difference between predicted and ground-truth states, are processed through a feedforward neural network. The model outputs corrections and confidence scores, enhancing co-training reliability by identifying high-confidence predictions for pseudo-labeling.

Comparison of Methodologies

Table III summarizes key features of the implemented methods:

Feature	KalmanNet	CNN-LSTM
Temporal Modeling	GRU-based	LSTM-based
Feature Extraction	Neural network	Convolutional layers
Adaptability to Noise	High	Moderate
Computational Efficiency	Moderate	High
Training Data Requirement	Moderate	High

TABLE III
COMPARISON OF KALMANNET AND CNN-LSTM

V. Model Validation and Experiments

Model validation and experimentation are essential to evaluate the performance, robustness, and practical applicability of the proposed methodologies. This chapter includes sensitivity analysis, robustness testing, and physical experiments to ensure the model's reliability in real-world scenarios.

Sensitivity analysis provides insights into how key parameters such as learning rate, time step, and the number of nodes in network layers affect the model's performance. Robustness testing evaluates the model under varying noise conditions, simulating real-world measurement inaccuracies. Finally, physical experiments validate the practical application of the model, bridging the gap between simulation and implementation.

Sensitivity analysis focused on the learning rate, which controls training speed and stability; the time step, which determines the length of input sequences and affects temporal dependency capture; and the number of nodes in each layer, which influences the network's expressiveness and computational efficiency. During each test, one parameter was varied while others were kept constant (default settings: learning rate 0.001, time step 5, CNN-LSTM with one convolutional layer of 64 filters, one LSTM layer with 64 hidden units, and one fully connected layer; KalmanNet with one GRU layer of 64 hidden units and one fully connected layer).

Learning Rate Sensitivity Analysis: Table IV shows that the optimal learning rate is 0.001, achieving an MSE of 0.021 and R^2 of 0.95. Lower rates (0.0001) led to underfitting, while higher rates (0.01, 0.1) caused instability or overfitting.

Learning Rate	MSE	R^2
0.0001	0.045	0.89
0.001	0.021	0.95
0.01	0.032	0.93
0.1	0.087	0.75

TABLE IV
LEARNING RATE SENSITIVITY ANALYSIS

Time Step Sensitivity Analysis: Table V indicates that a time step of 7 yielded the best results (MSE 0.018, R^2 0.96). Shorter time steps (e.g., 3) underperformed due to insufficient temporal context, while longer time steps (e.g., 10) introduced unnecessary complexity and overfitting.

Time Step	MSE	R^2
3	0.029	0.91
5	0.021	0.95
7	0.018	0.96
10	0.024	0.93

TABLE V
TIME STEP SENSITIVITY ANALYSIS

Nodes per Layer Sensitivity Analysis: Tables VI-VIII summarize node sensitivity analysis for convolutional, LSTM, and GRU layers. Optimal performance occurred at 128 nodes for the convolutional layer, 256 nodes for the LSTM layer, and 64 nodes for the GRU layer. Beyond these thresholds, performance plateaued or degraded due to overfitting.

Nodes	MSE	R^2
32	0.027	0.92
64	0.021	0.95
128	0.020	0.96
256	0.022	0.94

TABLE VI
SENSITIVITY ANALYSIS FOR CONVOLUTIONAL LAYER NODES

Nodes	MSE	R^2
32	0.028	0.91
64	0.021	0.95
128	0.019	0.96
256	0.018	0.97

TABLE VII
SENSITIVITY ANALYSIS FOR LSTM LAYER NODES

Nodes	MSE	R^2
32	0.030	0.90
64	0.021	0.95
128	0.020	0.96
256	0.022	0.94

TABLE VIII
SENSITIVITY ANALYSIS FOR GRU LAYER NODES

Robustness testing evaluated the model on datasets with varying noise levels, as shown in Table IX. The model performed well under low to moderate noise but struggled with high noise, highlighting the need for further optimization for real-world environments.

Dataset	Accelerometer Noise (mg)	Gyroscope Noise (°)
Dataset 1	0.6	0.012
Dataset 2	0.7	0.014
Dataset 3	0.8	0.016
Dataset 4	0.3	0.006
Dataset 5	0.4	0.008

TABLE IX
NOISE CHARACTERISTICS OF TEST DATASETS

Physical experiments validated the model using a setup with grid paper for accurate position tracking. The IMU and depth camera captured dynamic motion, and results were compared with recorded ground truth. Table 5.9 shows that x and y coordinate predictions were highly accurate (over 90

	x Accuracy (%)	y Accuracy (%)	z Accuracy (%)
Experiment 1	92.2	91.7	61.8
Experiment 2	95.7	94.7	42.2
Experiment 3	95.1	94.9	32.6

TABLE X
ACCURACY FOR COORDINATES IN PHYSICAL EXPERIMENTS

VI. CONCLUSION AND DISCUSSION

A. Conclusion

In this project, a novel framework was developed for soft robot pose estimation by leveraging the co-training method for multi-sensor fusion. The methodology addressed critical challenges associated with the inherent complexities of soft robotics, such as their infinite degrees of freedom, nonlinear material properties, and the limited availability of labeled training data. By combining data from IMUs, stereo cameras, and strain sensors, the study demonstrated the capability to enhance pose estimation accuracy and robustness, contributing significantly to the field of soft robotics.

The project began with a thorough review of smart materials, soft robotic structures, and relevant sensing technologies, laying a strong foundation for understanding the intricacies of the system. The integration of smart materials like SMPs and SMAs within the robotic structure showcased their potential to provide adaptability and precision under various stimuli, such as temperature. The sensor system design, incorporating high-precision IMUs and depth cameras, highlighted the importance of reliable data acquisition in achieving accurate pose estimation.

The implementation of the co-training machine learning approach was a key innovation in this study. By utilizing semi-supervised learning, the framework effectively utilized both labeled and unlabeled data, addressing the challenge of data scarcity. The algorithm's design incorporated advanced techniques such as CNN-LSTM networks and Kalman filtering, ensuring high performance in noise reduction and data fusion. Experimental results confirmed the efficacy of this approach, showing significant improvements in pose estimation accuracy compared to traditional methods.

Moreover, sensitivity analysis and performance evaluations demonstrated the system's robustness under various operational conditions. The practical application potential of the framework was validated through simulated and real-world scenarios, underscoring its adaptability and reliability. The findings suggest that this methodology could be instrumental in expanding the applicability of soft robotics across domains such as healthcare, manufacturing, and exploration.

In conclusion, this study not only addresses fundamental challenges in soft robot pose estimation but also provides a scalable and efficient solution through multi-sensor fusion and semi-supervised learning. Future research could focus on further enhancing the framework by incorporating additional sensor modalities, exploring advanced neural network architectures, and optimizing computational efficiency for real-time applications. These advancements would further solidify the role of soft robotics in tackling complex tasks across diverse environments.

B. Discussion

One of the key challenges faced in this project is the lack of training data derived from real-world applications. The semi-supervised co-training approach relies heavily on the

availability of both labeled and unlabeled data to refine the accuracy of the pose estimation model. While the simulated datasets provide a controlled environment for training and validation, they cannot fully replicate the complexities and variabilities encountered in practical scenarios. Acquiring real-world data would significantly enhance the robustness and generalizability of the algorithm. For future studies, efforts should focus on integrating real-world datasets from practical applications of soft robots, to further optimize the model's performance.

During the experimental phase, one of the notable constraints was the inability to accurately capture the real-time coordinates of the end effector throughout the motion. Measurements were limited to the initial and final positions, resulting in gaps in evaluating the intermediate pose estimation performance. This limitation hinders a comprehensive assessment of the system's real-time recognition capabilities and accuracy under dynamic conditions. Future work should aim to address this by incorporating real-time tracking solutions, such as motion capture systems or higher-resolution depth sensors, to provide a continuous stream of ground-truth data for evaluation.

The success of the position recognition algorithm lays a solid foundation for future integration into control systems. A closed-loop control framework could leverage the algorithm to dynamically adjust the robot's movements based on real-time feedback, thereby improving precision and adaptability [48]. This integration would not only enhance the operational efficiency of soft robots but also broaden their potential applications in tasks requiring high precision. Developing and testing this closed-loop control system should be prioritized in subsequent research to maximize the practical utility of the recognition algorithm.

The experiments highlighted the limitations of the depth camera's precision, which directly impacted the accuracy of the pose estimation. While the current setup performed adequately for smaller-scale structures, larger-scale systems with enhanced sensor resolution could yield significantly better results. Larger structures are not only more suited to handle higher forces but are also more aligned with market demands for devices capable of executing heavy-duty tasks [49]. Future studies should consider scaling up the robot's structure and employing higher-precision sensors or alternative sensing technologies to meet these requirements. Addressing these factors will likely result in improved performance and broaden the potential industrial applications of the system.

ACKNOWLEDGMENTS

APPENDIX

REFERENCES

[1] John A Hiltz. "Shape Memory Polymers - Literature Review". In: (2002).

[2] John A. Shaw et al. "Shape Memory Alloys". In: (2010). DOI: 10.1002/9780470686652.EAE232.

[3] Morgan T Gillespie et al. "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks". In: *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE. 2018, pp. 39–45.

[4] L. McDonald Schetky. "Shape-Memory Alloys". In: *JOM* (1987). DOI: 10.1007/BF03258890.

[5] Tsai Yu Hsin et al. "Shape memory polymer". In: (2007).

[6] Suvodeep Majumder, Joymallya Chakraborty, and Tim Menzies. "When less is more: on the value of "co-training" for semi-supervised software defect predictors". In: *Empirical Software Engineering* 29.2 (2024), p. 51.

[7] Yihao Zhang et al. "Semi-supervised learning combining co-training with active learning". In: *Expert Systems with Applications* 41.5 (2014), pp. 2372–2378.

[8] Weili Deng, Long Jin, and Weiqing Yang. "Piezoelectric Materials Design for High-Performance Sensing". In: *Crystals* (2023). DOI: 10.3390/cryst13071063.

[9] M. Chandra Sekhar et al. "A Review on Piezoelectric Materials and Their Applications". In: *Crystal Research and Technology* (2022). DOI: 10.1002/crat.202200130.

[10] Rosalina Lara-Rico et al. "Smart hydrogels based on semi-interpenetrating polymeric networks of collagen-polyurethane-alginate for soft/hard tissue healing, drug delivery devices, and anticancer therapies". In: *Biopolymers* (2023). DOI: 10.1002/bip.23538.

[11] Chang Gu et al. "High-durability organic electrochromic devices based on in-situ-photocurable electrochromic materials". In: *CheM* (2023). DOI: 10.1016/j.chempr.2023.05.015.

[12] Andrea Spaggiari et al. "Smart materials: Properties, design and mechatronic applications". In: *Proceedings of the institution of mechanical engineers, part l: journal of materials: design and applications* 233.4 (2019), pp. 734–762. DOI: 10.1177/1464420716673671.

[13] "Piezoelectric Polymers". In: *Encyclopedia of Polymer Science and Technology* (2023). DOI: 10.1002/0471440264.pst427.pub2.

[14] Bhavya Padha. "Fabrication Approaches for Piezoelectric Materials". In: 2022. DOI: 10.21741/9781644902073-2.

[15] Omprakash Sahu. "Piezoelectric Materials for Biomedical and Energy Harvesting Applications". In: 2022. DOI: 10.21741/9781644902097-6.

[16] "Piezoelectric Smart Materials and Commercialization". In: *Advances in chemical and materials engineering book series* (2023). DOI: 10.4018/978-1-6684-7358-0.ch011.

[17] Noelle Brigden. "Types, Properties and Characteristics of Piezoelectric Materials". In: 2022. DOI: 10.21741/9781644902097-1.

[18] "Piezoelectric Materials and Their Applications". In: 2022. DOI: 10.1201/9781003202608-12.

[19] Young Kon Kim. "Alloys, Shape Memory". In: (2006). DOI: 10.1002/0471732877.EMD002.

[20] Darel E. Hodgson, Ming H. Wu, and Robert J. Biermann. “Shape Memory Alloys”. In: 1990. DOI: 10.31399/ASM.HB.V02.A0001100.

[21] Suzuki Kikuo Dr. “Shape memory alloys”. In: (1982).

[22] Keith Melton and Olivier Dr Mercier. “Shape memory alloys”. In: (1977).

[23] R. Vaidyanathan. “Shape-Memory Alloys”. In: (2002). DOI: 10.1002/0471238961.1908011619030805.A01.PUB2.

[24] Thomas S. Wilson and Jane P. Bearinger. “Shape memory polymers”. In: (2005).

[25] Gaspard Lion. “Shape Memory Polymers”. In: 2022. DOI: 10.1201/9781003037880-10.

[26] Joseph D. Rule and Kevin M. Lewandowski. “Shape memory polymer”. In: (2009).

[27] Shaojun Chen et al. “Shape memory polymer based on betaine and preparation method of shape memory polymer”. In: (2014).

[28] “Hydrogels: Smart Materials in Drug Delivery”. In: 2023. DOI: 10.5772/intechopen.104804.

[29] Faiz Ali, Lakshman Neelakantan, and P. Swaminathan. “Electrochromic Displays via the Room-Temperature Electrochemical Oxidation of Nickel”. In: *ACS omega* (2022). DOI: 10.1021/acsomega.2c04859.

[30] Xia Zhou et al. “Multicolor Tunable Electrochromic Materials Based on the Burstein–Moss Effect”. In: *Nanomaterials* (2023). DOI: 10.3390/nano13101580.

[31] Jian Chen et al. “Resonant-Cavity-Enhanced Electrochromic Materials and Devices.” In: *Advanced Materials* (2023). DOI: 10.1002/adma.202300179.

[32] “New Anodic Discoloration Materials Applying Energy-Storage Electrochromic Device”. In: (2023). DOI: 10.20944/preprints202306.2068.v1.

[33] Susmriti Das Mahapatra et al. “Piezoelectric materials for energy harvesting and sensing applications: roadmap for future smart materials”. In: *Advanced Science* 8.17 (2021), p. 2100864.

[34] AS Fiorillo, CD Critello, and SA Pullano. “Theory, technology and applications of piezoresistive sensors: A review”. In: *Sensors and Actuators A: Physical* 281 (2018), pp. 156–175.

[35] Lu Yu et al. “A review on leaf temperature sensor: Measurement methods and application”. In: *Computer and Computing Technologies in Agriculture IX: 9th IFIP WG 5.14 International Conference, CCTA 2015, Beijing, China, September 27-30, 2015, Revised Selected Papers, Part I 9*. Springer. 2016, pp. 216–230.

[36] David Thomas Britton and Margit Härting. “Thermal imaging sensors”. Pat. 2013.

[37] *Kinematic Motion Analysis with Volumetric Motion Capture*. 2022. DOI: 10.1109/iv56949.2022.00019.

[38] Yunkun Cui Qingfeng Shi. “Research on Motion Capture System of Motor Skill Based on Computer Vision Technology”. In: *Journal of Electrical Systems* (2024). DOI: 10.52783/jes.1314.

[39] James Crowe et al. *PID control: new identification and design methods*. Springer, 2005.

[40] Horst Meier et al. “Smart Control Systems for Smart Materials”. In: *Journal of Materials Engineering and Performance* 20.4 (2011), pp. 559–563. DOI: 10.1007/S11665-011-9877-4.

[41] Max Schwenzer et al. “Review on model predictive control: an engineering perspective”. In: *The International Journal of Advanced Manufacturing Technology* (2021). DOI: 10.1007/S00170-021-07682-3.

[42] Horst Meier et al. “Smart Control Systems for Smart Materials”. In: *Journal of Materials Engineering and Performance* 20.4 (2011), pp. 559–563. DOI: 10.1007/S11665-011-9877-4.

[43] Tim Salzmann et al. “Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms”. In: *IEEE Robotics and Automation Letters* 8.4 (2023), pp. 2397–2404.

[44] Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control*. Springer Science & Business Media, 2004. ISBN: 9781852336943.

[45] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: Theory and Practice with MATLAB*. John Wiley & Sons, 2014.

[46] Sujan Kumar Roy, Aaron Nicolson, and Kuldip K Paliwal. “A Deep Learning-Based Kalman Filter for Speech Enhancement.” In: *INTERSPEECH*. 2020, pp. 2692–2696.

[47] Tara N Sainath et al. “Convolutional, long short-term memory, fully connected deep neural networks”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. Ieee. 2015, pp. 4580–4584.

[48] Jue Wang and Alex Chortos. “Control strategies for soft robot systems”. In: *Advanced Intelligent Systems* 4.5 (2022), p. 2100165.

[49] Kunming Zheng et al. “Intelligent control and simulation study for field flexible heavy duty robot”. In: *Advances in Mechanism and Machine Science: Proceedings of the 15th IFToMM World Congress on Mechanism and Machine Science 15*. Springer. 2019, pp. 1929–1938.