UNIVERSITÄT
BAYREUTH

Master Thesis

Nimisha Vernekar

July 2024
Version: Final

UNIVERSITÄT BAYREUTH

Universität Bayreuth

Fakultät Mathematik, Physik, Informatik

Institut für Informatik

Lehrstuhl für Angewandte Informatik VIII

Optimizing Acoustophoretic Volumetric Displays for Accurate Rendering of Levitated Particles: A Data-Driven Approach

Master Thesis

Nimisha Vernekar

1. Reviewer Prof. Dr. Jörg Müller
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

2. Reviewer Dr. Viktorija Paneva
Reasearch Institute CODE
Universität der Bundeswehr München

Supervisors Viktorija Paneva and Arthur Flieg

July 2024

Nimisha Vernekar

Master Thesis

Optimizing Acoustophoretic Volumetric Displays for Accurate Rendering of Levitated Particles: A Data-Driven Approach, July 2024

Reviewers: Prof. Dr. Jörg Müller and Dr. Viktorija Paneva

Supervisors: Viktorija Paneva and Arthur Flieg

Universität Bayreuth

Lehrstuhl für Angewandte Informatik VIII

Institut für Informatik

Fakultät Mathematik, Physik, Informatik

Universitätsstrasse 30

95447 Bayreuth

Germany

Abstract

Acoustophoretic volumetric displays offer a promising platform for rendering levitated particles in three-dimensional space, holding immense potential for applications in fields such as virtual reality, medical imaging, and scientific visualization. Despite this potential, achieving accurate image rendering is challenging due to the complex interplay of acoustic forces, particle dynamics, and system parameters. This thesis introduces a data-driven approach to optimize volumetric displays, enhancing the accuracy of rendered particle positions for path following.

Essential methods employed include the development of advanced neural network models, specifically Long Short-Term Memory (LSTM) networks and Feedforward Neural Networks (FNN), trained on particle trajectory data. The LSTM model, a testament to the effectiveness of our approach, demonstrated a significant improvement in accuracy. It achieved a Mean Squared Error (MSE) of 4.305e-05 and a Mean Absolute Error (MAE) of 0.0044 in the scaled version, compared to an MSE of 0.02878 and an MAE of 0.1095 in the unscaled version. This represents a reduction in prediction errors of over 99.8% and 96% respectively.

The approach involves developing advanced neural network models with a focus on achieving high prediction accuracy. The "Baseline Scaled" model demonstrated exceptional performance, achieving a Mean Squared Error (MSE) of 6.5018e-05 and a Mean Absolute Error (MAE) of 0.00515, with an R-squared value of 0.9999, signifying near-perfect predictions.

Experimental results demonstrate that the proposed data-driven optimization approach significantly enhances the rendering accuracy of acoustophoretic volumetric displays. These findings advance state-of-the-art volumetric display technologies, paving the way for their broader application in various scientific and technological fields.

Acknowledgement

First, I would like to thank my supervisors, Dr. Viktorija Paneva and Dr. Arthur Flieg, for their invaluable advice, support and encouragement throughout the whole research process. Their expertise, insightful feedback, and unshakeable patience have been instrumental in shaping this work and pushing me to strive for excellence.
I would like to thank my family for their unconditional love, encouragement and understanding during this journey. I am deeply thankful to my husband, Samarth, for his unwavering love, encouragement, and patience. His steadfast support has been my anchor during the ups and downs of this academic endeavor. I extend my heartfelt gratitude to my friends, Abhishek, Anuj, Vedant, Gokul, and Naveen, for their unwavering support and countless coffee breaks that provided much-needed respite and motivation.
I also want to thank the numerous other individuals whose support and contributions, have been instrumental in the completion of this thesis. Their assistance, whether direct or indirect, has not gone unnoticed and is deeply appreciated.

Contents

1 Introduction

1.1 Background and Motivation

Acoustic levitation is an innovative technology that enables the creation of volumetric displays by manipulating particles in the air using sound waves. This technique leverages the principle of acoustic radiation forces to trap and move particles, presenting a novel approach to rendering three-dimensional (3D) content visible to the naked eye without the need for traditional screens or headsets. The concept of acoustophoretic volumetric displays, which create dynamic visual content by rapidly moving a single levitated particle, has opened new avenues for interactive and immersive display technology.

Recent advancements in this field have demonstrated the feasibility of generating complex volumetric shapes by optimizing the trajectories of the levitated particles. [Fus+19] introduced a system capable of achieving high-speed particle movement to render volumetric displays, highlighting the potential for creating detailed and intricate visual content through the precise control of particle trajectories . Their work laid the foundation for subsequent research aimed at enhancing the accuracy and capability of acoustophoretic displays.

[Hir+19] made additional advancements by introducing the Multimodal Acoustic Trapping Display (MATD), which utilizes phased arrays of ultrasound transducers to provide a combination of visual, tactile, and audible input. This system not only levitates and moves particles but also generates tactile sensations and audible sounds, thus providing a multimodal interactive experience. MATD showcases the versatility of acoustic levitation in delivering comprehensive 3D content without the constraints of traditional display systems.

Furthermore, the OptiTrap system, which forms the backbone of this thesis, by [Pan+22], represents a substantial advancement in optimizing trap trajectories for acoustic levitation displays. This system employs a structured numerical approach to compute physically feasible and nearly time-optimal trajectories, enabling the creation of larger and more complex shapes than those previously demonstrated. OptiTrap automates the process of defining the levitated persistence of vision (PoV)

content. This ensures that the particle's motion remains within the limits of the device while maintaining high accuracy and efficiency.

Despite these advancements, several challenges remain in optimizing acoustophoretic volumetric displays for accurate rendering. These challenges include the need for precise control over particle dynamics and the development of algorithms capable of generating optimal trajectories in real time. This thesis aims to create a data-driven approach for optimizing acoustophoretic volumetric displays to improve the accuracy of levitated particle rendering. By leveraging advanced computational models and optimization techniques, this study seeks to improve the performance and reliability of these displays using a data-driven approach, paving the way for their widespread adoption and utilization in diverse applications.

1.2 Problem Statement

Despite significant advancements in the field of acoustophoretic volumetric displays, the challenges we face are complex and persistent. Achieving precise control over the particle dynamics and generating optimal trajectories in real-time is no easy task. The current state-of-the-art systems, such as OptiTrap, have shown promise, but they still struggle to accurately render particle positions. The inherent complexities of the acoustic forces, particle dynamics, and system parameters contribute to this issue. Therefore, there is a pressing need for a robust, data-driven approach to optimize the trap positions and improve the fidelity of the rendered particle paths.

1.3 Research Objectives

1. **To collect and pre-process high-quality datasets:**

    - Gather detailed trap positions, theta (position parameter), simulation data, and measured data from the OptiTrap setup of the acoustophoretic volumetric display.

    - Clean and merge the datasets to ensure that they accurately represent the operating conditions of the system.

2. **To conduct comprehensive data analysis and validation:**

- Fourier analysis was performed on both the sampled measured and simulated data to identify the significant frequency components and discrepancies.

3. **To develop a two-stage neural network model:**

   - **Stage 1:** Create a neural network that maps trap positions and theta to the measured positions and theta, learning the relationship between the trap configurations and the actual positions of the beads.

   - **Stage 2:** Design a neural network that modifies the trap positions to minimize the deviation between the measured points (from Stage 1) and the simulated points, using a cost function to adjust the trap positions accordingly.

4. **To implement an iterative optimization process:**

   - Develop an iterative process in which the modified trap positions from the second neural network are fed back into the first neural network and repeat the process until the deviation between the measured and simulated positions is minimized.

5. **To evaluate the performance of the optimized system:**

   - Quantitatively assess the accuracy of the produced particle positions under various operating conditions.

   - The performance of the optimized particle path configurations was compared with that of the baseline setup to demonstrate the improvements achieved through the data-driven optimization approach.

1.4 Thesis Structure

Chapter 2

This section provides a comprehensive overview of the development and current state of volumetric displays, as well as their principles and applications. In addition, the key concepts and definitions relevant to the study of acoustophoretic displays and volumetric visualization are introduced. Previous research and advancements in the field were reviewed, highlighting the key findings and methodologies used in the current study.

Chapter 3

This chapter delves into the fundamental principles of particle levitation using acoustic forces, explaining the mechanisms that enable the stable levitation and manipulation of particles. It discusses previous applications of data-driven methods for optimizing volumetric displays and sets the stage for the methodologies employed in this thesis.

Chapter 4

This section describes the methods used to gather and prepare data for analysis, including techniques for ensuring data quality and relevance during data collection and pre-processing.

Chapter 5

This chapter details the physical components used in the experimental setup, including ultrasonic transducers and other essential hardware, and describes the software tools and frameworks employed to control the hardware and process data. The procedures used to calibrate the system to ensure accurate particle levitation and movement were also discussed.

Chapter 6

This section outlines the process of feature extraction from the collected data to inform the model development. This section also discusses the criteria for selecting appropriate models and the process of training these models to optimize the volumetric display. The metrics used to evaluate the performance of the models and the overall system are also explained.

Chapter 7

This chapter provides a detailed analysis of the various developments of Model 2 designed to improve the accuracy of predicting particle positions in acoustophoretic volumetric displays. This section also presents the quantitative results of the experiments, including performance metrics that assess the system's accuracy and efficiency. It also provides visual representations of the levitated particles, demonstrating the system's capabilities and performance.

Chapter 8

This chapter interprets the experimental findings and discusses their implications and relevance in the field of acoustophoretic displays. It highlights the insights gained from applying data-driven techniques to optimize volumetric displays and acknowledges this study's limitations and the challenges encountered during the research.

Chapter 9

This chapter summarises the key findings of this thesis and emphasises the contributions made to the field of acoustophoretic displays. This study outlines the novel contributions and advancements of this research. This section provides suggestions for future research to further enhance the capabilities and applications of acoustophoretic volumetric displays.

2 Literature Review

2.1 Overview of Acoustophoretic Displays

An acoustophoretic volumetric display is a revolutionary form of 3D display technology that uses acoustic waves to manipulate and levitate particles inside a specific area, creating a visual representation that can be viewed from any angle. Unlike traditional 2D screens or holographic displays that manipulate light to create images, acoustophoretic displays use sound waves to generate three-dimensional representations. This technology leverages the principles of acoustic levitation, wherein ultrasonic sound waves create stationary wave patterns, leading to regions of low pressure that can trap and control tiny particles such as Styrofoam beads, water droplets, and minuscule LEDs.

Acoustophoretic displays, by manipulating particles within a precisely defined 3D setting, can create shapes, images, or animations that appear as floating, tangible entities. Their versatility is evident in the absence of physical support or containers, allowing them to create intricate and dynamic visuals.

The core mechanism of acoustophoretic displays involves the use of ultrasonic transducers to generate standing waves. These waves produce nodes (regions of minimal acoustic pressure) and antinodes (regions of maximal acoustic pressure), where the particles can be trapped and manipulated. The typical frequencies used in these setups range from 20 kHz to several hundred kHz, depending on the size and type of particles being levitated. The transducers can be arranged in various configurations to create complex acoustic fields, allowing for precise control over the position and movement of particles.

Moreover, the non-contact nature of acoustic levitation makes it ideal for applications where contamination-free environments are crucial. This includes biomedical applications, such as handling biological samples or manipulating delicate substances without physical contact. Additionally, this technology can be employed in material science to study the properties of materials without container-induced contamination.

Overall, acoustophoretic displays represent a significant advancement in display technology, offering new ways to visualize and interact with digital content. The combination of acoustic levitation principles and advanced control techniques enables the creation of highly detailed and dynamic 3D visuals that can be experienced in a fully immersive manner.

2.2 Challenges in Acoustophoretic Point-of-View Displays

While the potential applications of acoustophoretic displays are vast, significant challenges remain, particularly in achieving high-quality images. One major challenge is the need for particles to move at extremely high velocities to create images, resulting in trap positions that no longer correspond to the particle positions. This discrepancy impacts the accuracy of the display. Additionally, achieving and maintaining precise control over particle motion at these velocities is complex, and any deviation can lead to image distortion.

2.3 Previous Research and Related Work

An innovative work in this field is the investigation carried out by [Mar+15]. This study demonstrated the use of phased arrays of ultrasonic transducers to create dynamic acoustic holograms. These holograms allow precise control over the position and movement of suspended particles. [Mar+15] utilized frequencies of approximately 40 kHz and achieved the precise manipulation of polystyrene beads with diameters ranging from 1 mm to 3 mm.

[Hir+19] made significant contributions to their study. This study introduced a system in which multiple transducers operate at 40 kHz levitated particles, creating interactive mid-air displays. The study highlighted the potential of combining visual displays with tactile feedback using midair haptics, enabling users to interact with the levitated particles through touch-less gestures, thus enhancing the feeling of engagement.

[For+13] explored the use of acoustophoretic methods for contactless transport and the handling of matter. Their study, published in PNAS, demonstrated the manipulation of particles of up to a few millimetres in size using ultrasonic transducers operating at 40 kHz. This study provides valuable insights into the potential

applications of acoustic levitation in fields requiring precise, contamination-free manipulation of particles.

[Ino+19] advanced this field by developing acoustic boundary holograms for macroscopic rigid-body levitation. Their work demonstrated the ability to levitate and manipulate larger objects, such as small tools or biological samples, using standing wave patterns created by transducers operating at 40 kHz. This study highlighted the scalability of acoustic levitation techniques.

[WHA18] integrated acoustic levitation into robotic systems, facilitating the non-contact manipulation of delicate objects. Their experiments with fluid droplets manipulated by ultrasonic transducers at 40 kHz, demonstrated potential applications in biomedical and chemical processing where contamination-free handling is crucial.

[RSK22] investigated the kilohertz-frequency rotation of acoustically levitated particles. Using frequencies ranging from 20 kHz to 100 kHz, they achieved precise control over particle orientation, which is essential for applications in additive manufacturing and 3D printing. This research suggests that acoustic levitation can enhance manufacturing accuracy and efficiency.

[AMA20] reviewed recent advances and challenges in acoustic levitation, discussing the effectiveness of standing wave levitation for small particles and droplets, and the flexibility offered by travelling wave levitation for larger objects. Their comprehensive analysis provided a foundation for future innovations in this technology.

2.4 Summary

A diverse array of applications can be achieved by integrating acoustophoretic process into volumetric displays, including medical imaging, scientific visualization, interactive entertainment, and art installations. These displays offer a novel approach to visualizing complex data and interacting with digital content in a way that feels physically solid. The studies reviewed in this chapter illustrate the significant progress made in understanding and utilizing acoustic levitation, thereby providing a solid foundation for future innovations and applications in this exciting field.

3 Theoretical Foundations

3.1 Particle Levitation Principles and Mechanisms

Particle levitation through acoustic waves is a sophisticated technology that involves suspending particles in mid-air by using sound waves. This process is based on several core principles and mechanisms, including acoustic radiation pressure, standing waves, travelling waves, and the use of coordinated sound emitters called phased arrays.

Acoustic Radiation Pressure: Acoustic radiation pressure is the force exerted by sound waves on particles within a sound field. This force results from the transfer of energy from the sound waves to particles, creating a steady push that can lift particles against gravity [AMA20]. The strength of this push depends on factors such as the frequency of sound waves (how high or low the pitch is), their loudness (amplitude), and the properties of the particles, such as their size, shape, and density.

Standing Waves: In a standing wave setup, sound waves are bounced back and forth between two surfaces, creating a pattern of fixed points where the sound pressure is minimal (nodes) and points where the pressure is maximal (antinodes). Particles tend to become trapped at these nodes because a stable pressure environment allows precise control. For example, [Hir+19] used sound waves with a frequency of approximately 40 kHz (which is beyond the range of human hearing) to levitate small particles, such as Styrofoam beads and water droplets. This demonstrates that standing waves can create stable conditions for levitating particles ranging from micrometres to a few millimetres in size. Their paper includes the image of the experimental setup and the levitated particles, which can be highly illustrative, refer Fig 3.1

Traveling waves: Traveling wave levitation uses moving sound waves to create a continuously shifting pressure field. Unlike standing waves, travelling waves do not have fixed nodes or antinodes. Instead, they generate a moving push that transports particles over larger distances. This method is useful for applications requiring dynamic movement of particles, such as creating complex animations or interactive displays. [WHA18] used travelling waves to manipulate fluid droplets,

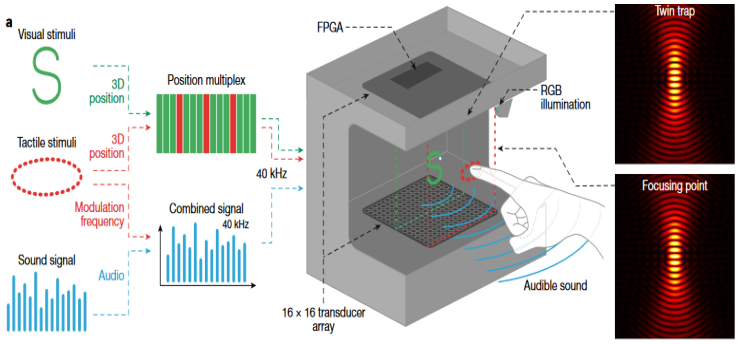


Fig. 3.1: Example Experimental Setup from [Hir+19]

allowing them to merge or mix without directly touching them, which helps avoid contamination.

Coordinated Sound Emitters (Phased Arrays): A phased array is a setup in which multiple small sound emitters (transducers) are arranged in a specific pattern and work together. By precisely controlling the timing (phase) and loudness (amplitude) of the sound waves emitted by each transducer, complex sound fields that can trap and move particles in three dimensions can be created. For instance, [Mar+15] used phased arrays to create detailed sound fields that acted like invisible hands, allowing them to control the position and movement of suspended particles with high precision. This approach enables the creation of highly detailed and interactive volumetric displays that enhance both visual and tactile experience.

Levitation of Various Particle Types: Acoustic levitation is versatile and can handle various types of particles, including solid beads, liquid droplets, and small electronic components, such as LEDs. The size of the particles that can be levitated depends on the frequency and amplitude of the sound waves, as well as the density and acoustic properties of the particles. [For+13] successfully manipulated particles up to a few millimetres in size using 40 kHz ultrasonic waves. Similarly, [RSK22] explored the rotation of acoustically levitated particles at kHz frequencies, highlighting the versatility of acoustic levitation in handling various particle types and sizes.

Detailed Mechanisms:

Node and Antinode Formation: When sound waves bounce back and forth, they create patterns of high and low pressure. Nodes are low-pressure points at which particles can be trapped, whereas antinodes are high-pressure points. By adjusting the frequency and amplitude of sound waves, researchers can control where these nodes and antinodes form, creating stable points for levitation.

Dynamic Adjustment and Stability: The stability of the levitated particles is maintained through the continuous adjustment of the sound wave parameters. Real-time feedback mechanisms and sensor data can be used to dynamically alter the phase and amplitude of the emitted sound waves, ensuring consistent levitation and precise positioning, even under varying environmental conditions.

Interaction with Particle Properties: The effectiveness of acoustic levitation depends on the intrinsic properties of the particles. Factors such as the size, shape, density, and acoustic impedance play significant roles in determining the response of particles to the acoustic field. By tailoring the sound wave characteristics to match the specific properties of the particles, researchers can achieve the optimal levitation conditions.

3.2 Data-Driven Approaches in Volumetric Display Optimization

The optimization of volumetric displays, particularly those that utilize acoustic levitation, can be significantly enhanced using data-driven approaches. These approaches leverage computational models, machine learning algorithms, and optimization techniques to improve the performance and capabilities of volumetric displays.

Computational Models and Simulations: Computational models are detailed computer programs that simulate the behaviour of particles in an acoustic field. These models can help researchers predict how particles move and interact when exposed to sound waves. [Ino+19] used computational models to design acoustical boundary holograms to levitate larger objects. Their models simulated the interaction between sound waves and objects of various sizes and shapes, allowing them to optimize the arrangement of sound emitters (ultrasonic transducers) for stable levitation. One major takeaway from their work was the ability to create stable levitation conditions for larger and more complex objects by fine-tuning acoustic field parameters.

Machine Learning Algorithms: Machine learning is a type of artificial intelligence in which computers learn from data to make predictions or decisions. In the context of volumetric displays, machine learning algorithms can analyze large amounts of data to identify patterns and optimize the control settings for sound waves. For example, these algorithms can determine the best frequency and loudness settings for ultrasonic transducers to achieve stable levitation. By continuously learning from the data, these systems can make real-time adjustments to keep the particles levitated and precisely positioned.

Optimization Techniques: Optimization techniques are mathematical methods used to find the best solution to a problem. In volumetric displays, these techniques can fine-tune various parameters, such as the arrangement of sound emitters, timing (phase), and loudness (amplitude) of the sound waves.

- **Genetic Algorithms**: Genetic algorithms mimic the process of natural selection, where the best solutions are selected and combined to produce new, improved solutions. This iterative process continues until an optimal solution was obtained. This method is beneficial for determining the optimal configurations for complex systems with many variables.

- **Particle Swarm**: Particle swarm optimization is inspired by the social behaviour of animals, such as birds flocking or fish schooling. In this method, a group of potential solutions, called particles, moves through the solution space. Each particle adjusts its position based on its own experience and that of neighbouring particles. Over time, the swarm converges to an optimal solution. This technique is effective for exploring a wide range of possible configurations and determining the most effective configuration.

[Hir+19]applied these optimization techniques to enhance volumetric display systems. Using genetic algorithms, they were able to identify and select the best configurations for their system from a large set of potential solutions. This process involves iteratively combining the best settings to achieve optimal control over sound emitters. In addition, they utilized particle swarm optimization to explore various configurations of sound emitters. This technique allows them to dynamically adjust the settings of the ultrasonic transducers in real-time, ensuring that the particles remain stable and precisely positioned. This resulted in a significant improvement in the visual quality of the display, with particles forming clear and stable images in mid-air. These enhancements have enabled their volumetric display system to provide a more immersive and engaging user experience.

Feedback Mechanisms: Feedback mechanisms are systems that monitor the performance of a volumetric display in real time and make necessary adjustments. Sensors track the position and stability of levitated particles and provide data to the control system. These data are then used to dynamically alter the sound wave settings, ensuring that the particles remain in their intended positions even if external conditions change. These mechanisms are crucial for interactive applications, in which users might interact with levitated particles, requiring continuous adjustments to maintain stability.

Applications and Future Directions: Data-driven approaches are revolutionizing the capabilities of volumetric displays, enabling new applications in fields such

as medical imaging, scientific visualization, and interactive entertainment. For instance, in medical imaging, these displays can create detailed 3D images of internal body structures, helping doctors diagnose and treat conditions more effectively. In scientific visualization, they can be used to model complex phenomena such as fluid dynamics or molecular structures, providing researchers with deeper insights. Interactive entertainment can benefit from more immersive and engaging experiences, where users can interact with 3D holograms in real time. Future research should focus on integrating artificial intelligence and machine learning more deeply into the control systems of these displays, further enhancing their adaptability and performance.

4 Dataset Analysis and Preparation

4.1 OptiTrap Setup and Shape Selection

The experiments were conducted using the OptiTrap system, which utilizes acoustic levitation to precisely control and manipulate particles in mid-air. For this study, we selected the shape of a fish as a proof-of-concept from the already available shapes from the paper[Pan+22]. The fish shape was chosen due to its complex geometry, which provides a rigorous test for the accuracy and capabilities of the volumetric display system. This selection allows for a comprehensive evaluation of the system's performance in rendering detailed and intricate curves, making it an ideal candidate for optimizing particle path following.

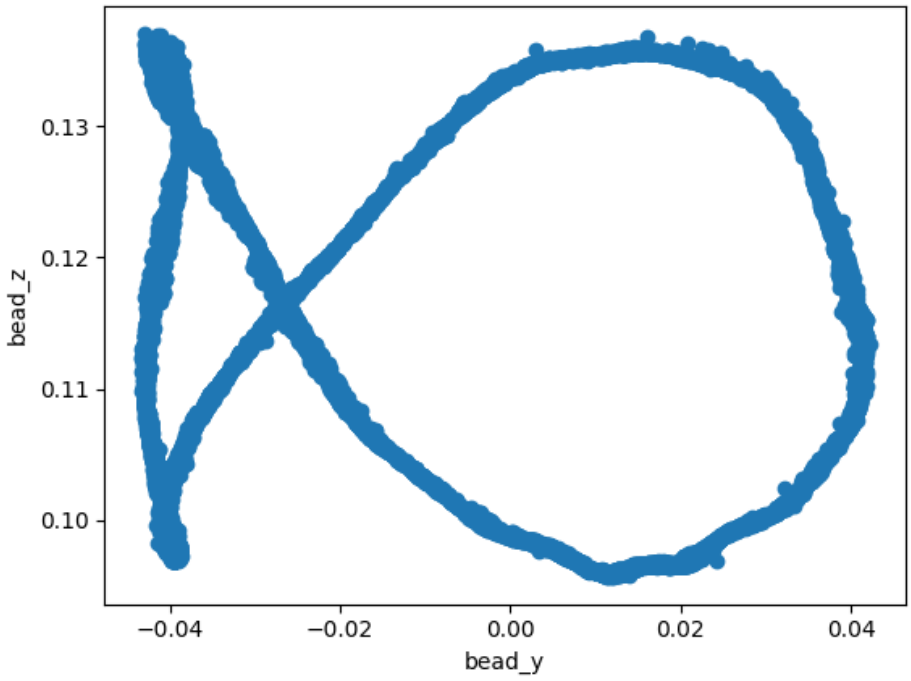


Fig. 4.1: Plot of fish shape using Measured Positions from the OptiTrap(in m)

The fish shape used in the experiments is defined by a set of spatial coordinates (x, y, and z) that represent the positions of the acoustic traps required to create the desired shape, in this case, the fish. The shape depicted in Fig. 4.1 represents a trajectory of a particle or object, visualized in a 2D projection(Bead X is always 0)

with Bead Y on the x-axis and Bead Z on the y-axis. This plot is created with the measured data obtained from the levitator which rendered the corresponding traps for the shape of the fish in mid-air. The unit here is in Meters(m).
The coordinates of the shape can be explained as follows:
The plot starts at a position near the coordinate (0.04, 0.119) and returns to the same point, completing a loop. The path begins at this position, moves upward, towards a peak of 0.145m on the Y-axis (bead_z), then down towards the left, forms a tight loop and continues to go upwards, forms another tight loop and then descends and turns upwards to the starting position. For simplicity, in the rest of the study, we have used the measurements in terms of Millimetre(mm) for easier calculations.

The Bead Y coordinates vary between approximately -40mm and 40mm, showing a wide lateral movement.
The Bead Z coordinates range from around 95mm to 145mm, indicating vertical movement with varying heights.
In the plot 4.1, we can notice the imperfect loop and deflecting path of the particle as measured by the OptiTrack cameras, refer (5) and our objective is to optimize the path followed by the particle such that it is as close to the original traps and corresponding simulation path as possible, refer Fig. 4.2

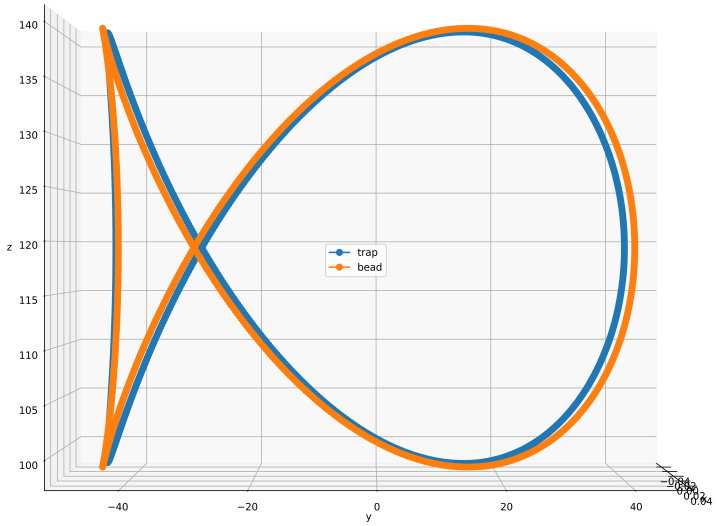


Fig. 4.2: Trap Positions and Simulation for Fish shape in mm

Traditionally, the OptiTrap system employs 1000 trap positions, corresponding to 1000 simulation points for the periodic motion. The system also includes 3000 points for the ramp-up and ramp-down phases, with each phase contributing to 1500 points respectively.

The Ramp-up and Ramp-down Phases are essential to the particle levitation as the bead needs to gain appropriate momentum and velocity before it is run on the actual trap positions and the same logic for when the system is stopped or completed with the rendering of the desired trap positions, there has to be a smooth transition between the two modes so that the bead doesn't fly off from the area under acoustic pressure.

However, the measured data obtained from the OptiTrack cameras and the Motive application shows a different distribution of points:

- Ramp-up and Ramp-down Phases:
  Each phase consists of 120 points, forming 2 complete cycles of the fish shape.

- Periodic Motion:
  Each cycle of the fish shape is recorded with 24 points, repeated 60 times for a complete dataset.

This results in a detailed dataset that reflects the actual behaviour of the particles within the trap positions, which is essential for validating the simulation models.

Data acquisition and pre-processing, which centres on the OptiTrap system, played a pivotal role in this study. This system utilizes acoustic levitation for precise control and manipulation of particles in mid-air. The collected datasets consisted of trap positions, simulation data, and measured data, each serving a specific function for evaluating and enhancing the performance of the system, refer the Fig. 4.3 for understanding the flow of data throughout the process.

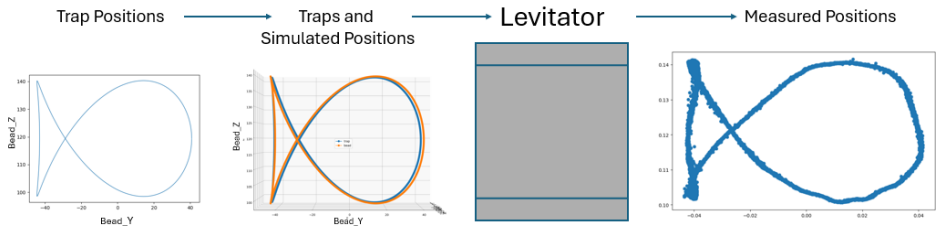


Fig. 4.3: Flow of Data

Trap positions refer to the spatial coordinates (x, y, and z) where acoustic traps are strategically placed to influence particle movement. These positions are meticulously determined based on the particle's initial location, expected shape, mode and calculated deltas, allowing fine-tuning to achieve optimal levitation and control. Ac-

curate trap positioning is essential because it provides a framework for manipulating particles within the acoustic field.

Simulation data were generated to predict the behaviour of particles under the influence of predefined trap positions. These data are fundamental for understanding the system performance under controlled conditions and validating the theoretical models used to design the trap configurations.

The measured data were obtained during the actual experiments, in which the OptiTrack camera system captured the real-time motion of the particles. These data reflect the actual behavior of the particles within the trap positions and are essential for validating the simulation models. The measured data provide a realistic account of the system's functionality, highlighting any discrepancies between the simulated and actual outcomes and guiding further refinements in the system design and operation.

The primary objective of this study was to optimize the path followed by the levitated particles. To accomplish this, the goal was to reduce the deviation between the simulation points and the measured positions. It is imperative to reduce this deviation in order to enhance the reliability and accuracy of the OptiTrap system, as a lower deviation suggests that the system's real-world performance closely aligns with its theoretical expectations.

In the following sections, we will go through how the dataset was setup and what methods were used to work towards the objective of this thesis.

4.2 Data Acquisition and Pre-processing

The dataset used in this study included several key features crucial for understanding the behaviour of the OptiTrap system and improving its performance. These key features from the dataset are the trap positions, simulated positions, measured positions, and theta values. Understanding the importance of these features is vital to effectively train and optimize neural network models based on their functionality.

- **Measured Points (measured_x, measured_y, measured_z):**
  The measured positions were the actual coordinates recorded by the device for the same trap configuration. These positions reflect the real-world performance of the system and capture any deviations or discrepancies from the simulated positions. These deviations can arise from various factors, such as environmental disturbances, humidity, light, inaccuracies in the trap positions, or limitations in the control algorithms.

- **Trap Positions (traps_x, traps_y, traps_z):**
  The trap positions are the coordinates in the xyz-plane where the acoustic traps are created. These traps utilize acoustic levitation to precisely control and manipulate Styrofoam beads. By carefully positioning these traps, a complex shape, such as that of a fish, can be formed. The stability and precision of these traps are fundamental to the system's performance.

- **Reference Points (reference_x, reference_y, reference_z):**
  The simulated positions were the expected coordinates of the beads when the system operated with the corresponding trap configurations. These positions were generated through simulations applying NLP (Non-Linear Programming) algorithms of the acoustic field and particle interactions[Pan+22]. These represent the ideal behaviour of a system under perfect conditions.

- **Theta (θ):**
  Theta θ values represent the particle parameter, which indicates how far along the particle path is the position of the bead; this corresponds to each set of traps and simulated and measured positions. In this context, theta provides a temporal dimension to the dataset, allowing for analysis of the system's behaviour over time. Theta ranged from 0 to 2π. Notably, the same theta value can appear multiple times with different measured values, indicating measurements from multiple cycles for the same trap and simulation configuration.

The trap positions are fed into the levitator as a CSV file that contains only the x,y, and z positions of the bead to form one complete rotation of the desired shape in space. This was used as the periodic data, which needs to run as a cycle of 60 on the levitator for the human eye to be able to see the rendered shape in mid-air due to the phenomeno of PoV(Persistence of Vision). When running, the system always performs Ramp-Up motion, leading to periodic motion for 60 cycles and then Ramp-Down, and this continues until the system is stopped. When we desire to run the trap positions on the levitator, the Ramp-Up is first run for 120 points, then the actual trap positions periodically for 60 cycles, each cycle with at least 1000 points, which in total makeup the desired shape, and then Ramp-Down again for 120 points. The measurements were taken as datasets in the CSV files, which are detailed logs of the particle motion over time. The features in the measured data include timestamps and bead x-, y-, and z-positions in metres and as mentioned earlier, we use the unit of mm for simplicity.

4.3 Fourier Analysis and Data Cleaning

The primary work that was done was to analyze and optimize the periodic behaviour of particles under acoustic levitation. In addition, the transient behaviour of the ramp-up and ramp-down phases is not representative of the steady state and will find no meaningful analysis. By removing these sections, we ensured that the dataset represented appropriate operating conditions for system stability. It is crucial to ensure that only periodically relevant data are analyzed. For instance, in this scenario, an amplitude time plot for only one 3-minute file was analyzed. The plot consisted of 27 complete periodic cycles.

To remove irrelevant sections and improve the data quality, it is essential to perform a cleaning process on the data. The amplitude-time plot indicates the number of cycles and modes within the data. Signal behaviour was visualized, patterns were removed, and the effects of both ramp-up and ramp-down in these data were determined. A plot of the amplitude versus time before and after cleaning is presented below in Fig 4.4 and 4.5

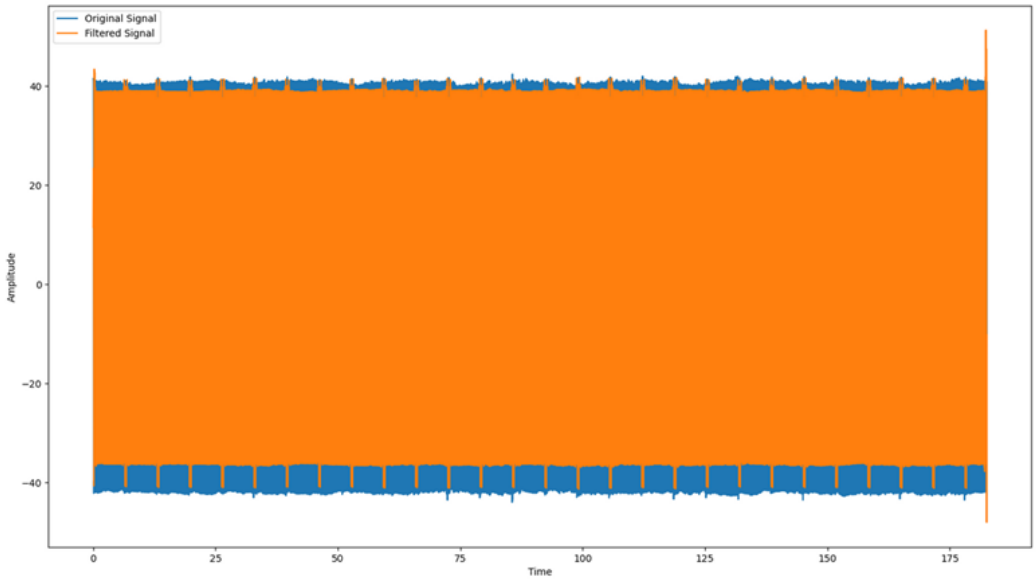


Fig. 4.4: Frequency plot of a single 3-minute raw file, showing spikes at ramp-up and ramp-down points indicating 27 cycles.

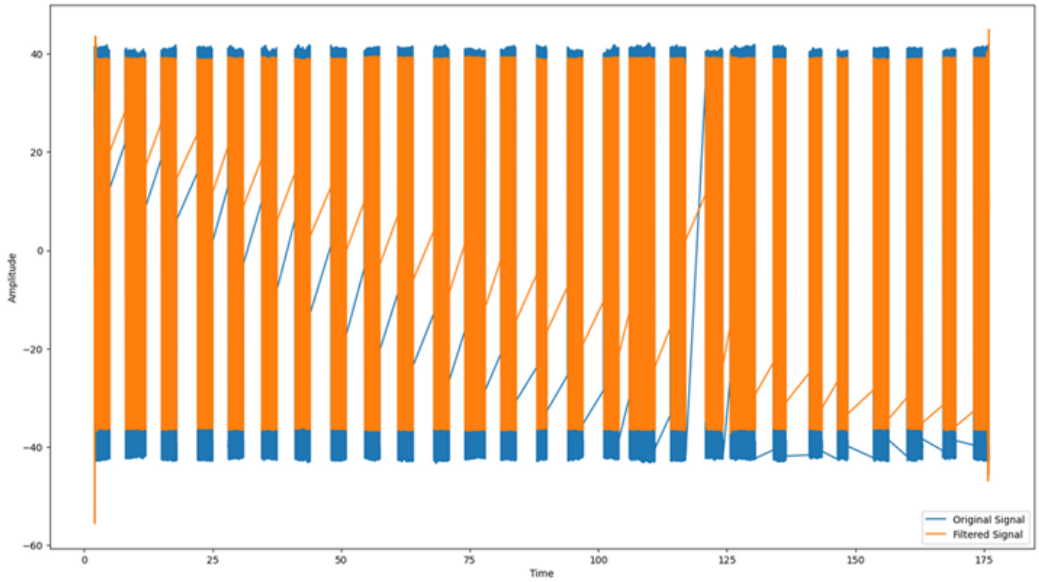


Fig. 4.5: Frequency plot after cleaning up the data in one file, with gaps due to the removal of ramp-up and ramp-down sections.

In the fig 4.4, the spikes of orange that keeps occuring periodically over the blue region implies the end and beginning of each cycle, these spikes are the Ramp-Up and Ramp-Down sections of the measured data. Since these Ramp-Up and Ramp-Down data are not beneficial to understanding the underlying patterns of the path following problems and its optimization, we need to clean the data so that we are left with only the periodic measured data, refer Fig. 4.5.

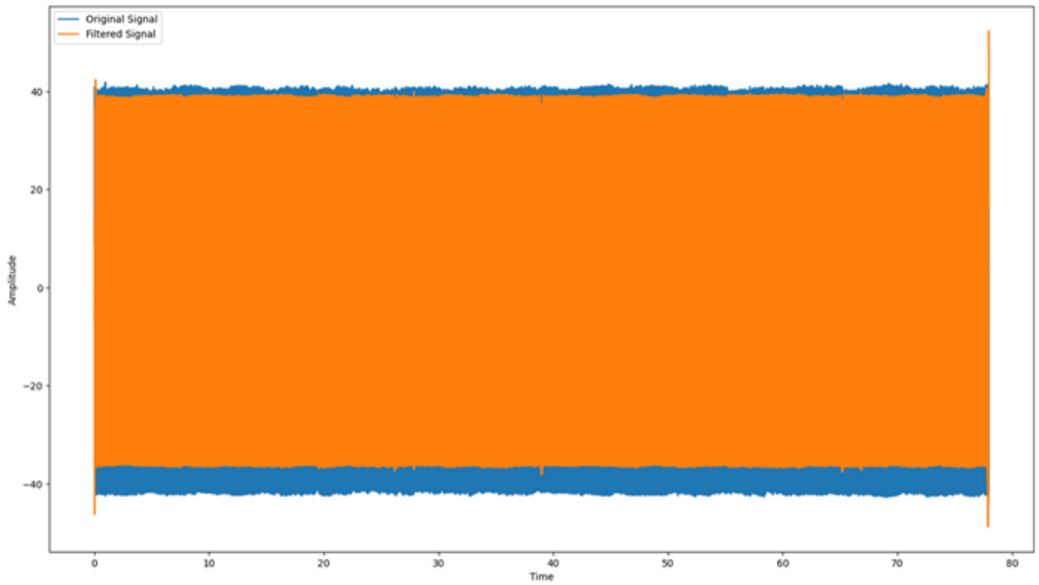


Fig. 4.6: Frequency plot after merging two cleaned files and sampling them.

The cleanup process involved identifying and removing the ramp-up and ramp-down sections. This allowed a clearer visualization of the signal behaviour and pattern identification. After cleaning, two such (cleaned) files were merged as shown in Fig 4.6, this merged data is a more stable dataset with which we can carry out our further analysis of the bead positions and understand why there is a discrepancy between the simulated reference positions and measured positions.

4.4 Dataset Creation

After cleaning the two measured data files and deleting the Ramp-Up and Ramp-Down sections, they were combined. This dataset served as the basis for all developments in this thesis. As previously mentioned, the measured data contained multiple cycles of fish shape, and each cycle contained 24 points. This is a periodic behaviour of the measured data. The presence of timestamps in the measured data caused a time discrepancy due to gaps caused by the removal of Ramp-Up and Ramp-Down data during cleaning. This must be rectified. The simplest method to achieve this is to manually update the timestamp column in the CSV file. However, this was a tedious task and required handling using Python code. The idea was to create a new column called an offset, which contains adjusted timestamps that reflect a continuous sequence with minimal discrepancies.

We aim to create a new column, time_w_offset, which adjusts the timestamps in a specific pattern. For the first 24 points, the timestamps remain unchanged. For the subsequent 24 points, 0.10 is subtracted from their timestamps and stored in time_w_offset. For the next set of 24 points, 0.20 (2 x 0.10) is subtracted from their timestamps, and this adjusted value is stored in time_w_offset. This pattern continues, with each successive group of 24 points having an additional 0.10 subtracted from their timestamps. Since we have 24 measured points in each periodic cycle, the interval size is set to 24, which specifies the number of rows over which the offset is applied. The calculated offset was subtracted from the original offset values, effectively adjusting the timestamps in a staggered manner.

However, plotting the adjusted data revealed that the beads moved backwards frequently. Attempts to fix this included:

- The deletion of problematic points was initially considered in this study. However, this approach proved impractical because of the sheer number of points exhibiting a backward movement. Manual identification and removal of each problematic point would be time-consuming and labour-intensive and could potentially lead to the loss of valuable data points that contribute to the overall pattern.

- Replacing problematic points with non-problematic ones is another strategy that has been attempted. The idea was to identify the points causing backward movement and substitute them with points that follow the expected forward trajectory. Despite these efforts, the approach failed to produce a smooth and continuous fish diagram. The replacement points were not seamlessly

integrated with the surrounding data, leading to disruptions in the overall pattern and failure to capture the fluid motion of the fish shape, refer Fig.4.7

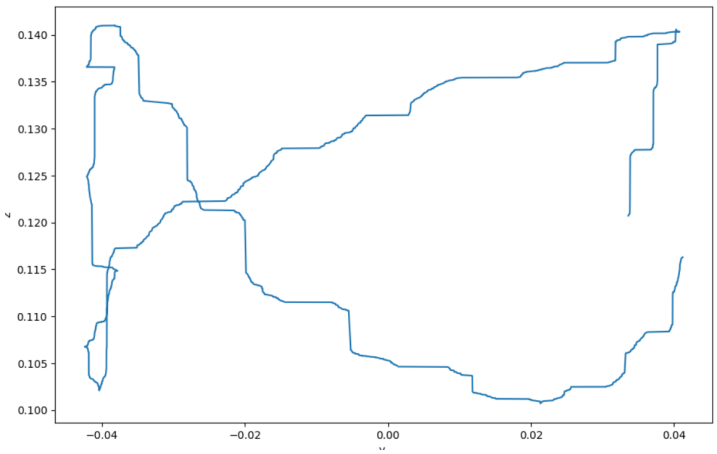


Fig. 4.7: Adjusted time offset Plot of Bead_Y vs Bead_Z

- Sorting the data by time and adjusting the bead_y and bead_z columns to follow the pattern observed in the original cycle was also explored. This method involved reordering the data to ensure a logical progression in time and then aligning the y and z coordinates to mimic the expected trajectory of the fish. However, this approach does not achieve the desired outcome, refer Fig. 4.8. Although sorting improved the temporal sequence, it did not resolve the underlying issue of backward movement. Adjustments to bead_y and bead_z were insufficient to create a coherent and smooth fish shape, indicating that the problem was more complex than simple reordering and alignment could be fixed.

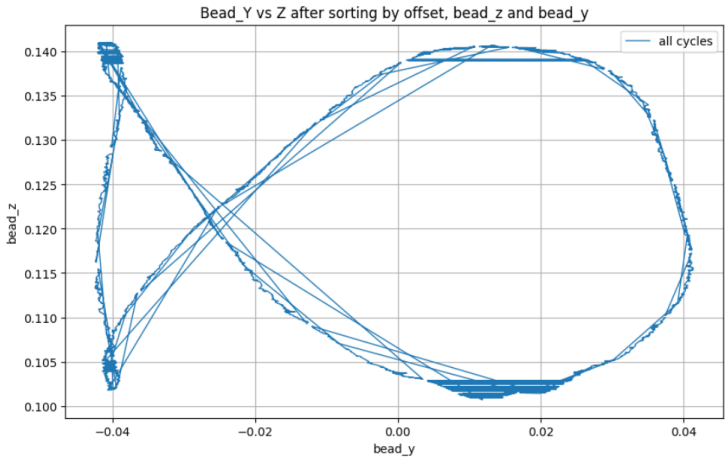
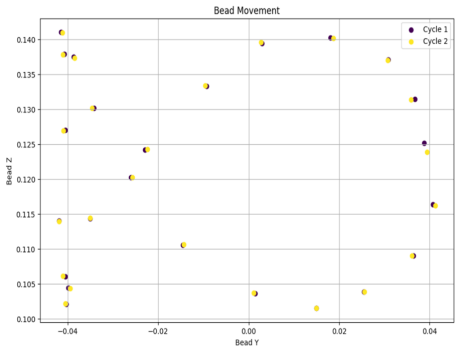
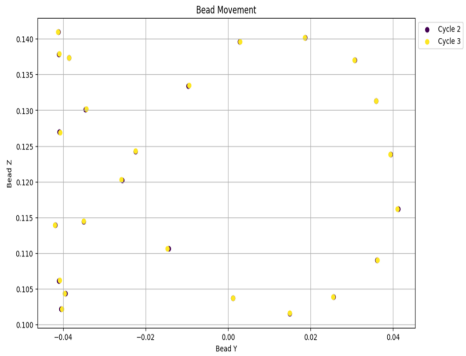


Fig. 4.8: Bead_Y vs Bead_Z after sorting by offset, Bead_Y and Bead_Z



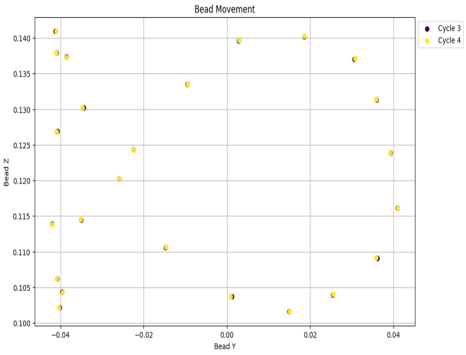
(a) Plot of points of cycle 1 and 2



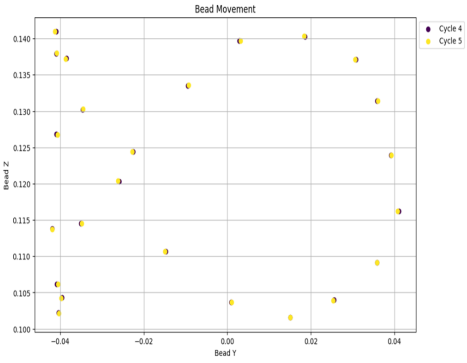
(b) Plot of points of cycle 2 and 3

Fig. 4.9: Plot of points of cycle 1, 2 and 3

Upon reviewing the plots, it was noticed that the jumps seemed to occur systematically and frequently, leading to suspicion that the cycle duration might not be exactly 100 ms as assumed. To investigate this, a comparison of all points of the two subsequent cycles was conducted individually. By plotting all the points of the initial cycle together with all the points from the second cycle in the yz-plane, we observed whether any, some, or all points of the second cycle lay behind the points of the first cycle along the trajectory of the fish, refer Fig. 4.9a.



(a) Plot of points of cycle 3 and 4

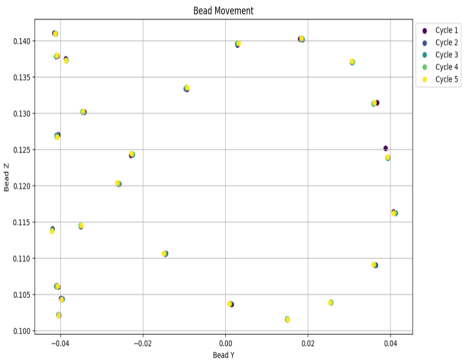


(b) Plot of points of cycle 4 and 5

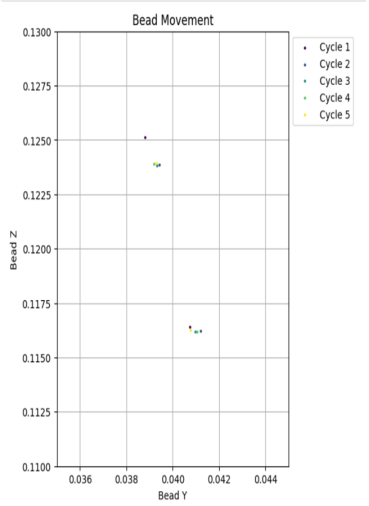
Fig. 4.10: Plot of points of cycle 3, 4 and 5

It is essential to repeat this comparison with several pairs of subsequent cycles to observe the patterns. This helped in deducing the effective cycle duration by looking at how much we need to adjust the timings of the individual cycles so that we obtain a forward motion along the shape trajectory.

Comparing the individual points of subsequent cycles revealed that the beads of the second cycle were not perfectly aligned behind the first cycle, especially at the nose of the fish. Points tended to shift slightly to either the left or right; refer to Fig. 4.9a. Subsequent comparisons between cycles 2 and 3 showed better alignment, refer Fig. 4.9b, and similar behaviours were observed in the comparisons between cycles 3, 4, and 5, refer Fig.4.10a and 4.10b.

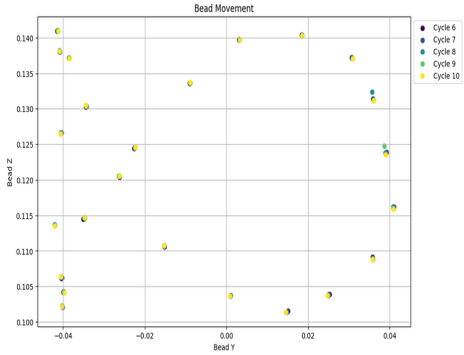


(a) Plot of points of cycle 1 to 5

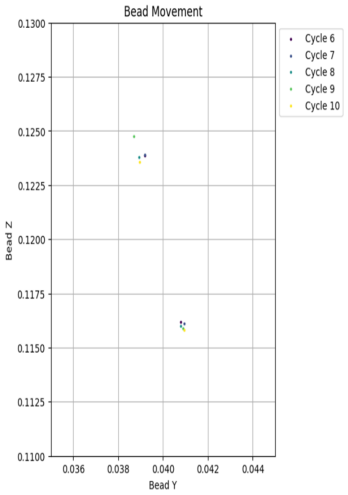


(b) Plot of points of cycle 1 to 5 zoomed in

Fig. 4.11: Plot of points of cycle 1 and 5



(a) Plot of points of cycle 6 to 10



(b) Plot of points of cycle 6 to 10 zoomed in

Fig. 4.12: Plot of points of cycle 6 and 10

When plotting the five cycles together, ref Fig. 4.11a, the point near the nose of the fish from cycle 1 appeared as an outlier compared to the rest of the cycles. Zooming in on the points at the nose with a reduced point size showed that while the points were slightly behind the previous ones, they also shifted on the x-axis, ref Fig.4.11b. Further plots of the next five cycles revealed some points that acted as outliers, particularly at the nose of the fish in cycles 8 and 9, ref Fig. 4.12a and Fig. 4.12b.

By examining the timestamps of each set of points, we observed that the total timestamps for points 0 to 23, 24 to 47, and so on, were remarkably consistent, all around 0.095825 s. This consistency suggests that the cycle duration might require fine-tuning to achieve a smooth forward motion along the shape trajectory.

4.5 Introducing theta

To address these challenges and achieve more coherent motion, we explored the concept of theta θ as a fundamental parameter in the control and optimization process of the OptiTrap system. Theta represents the position of the particle along the trajectory path. Obtaining accurate theta values is crucial for defining the optimal trajectory of the levitated particle. To generate theta values, we initialized the environment, configured the essential libraries, and set up the system parameters. The desired trajectory shape was modelled, and decision variables, including theta, were defined within the non-linear programming (NLP) setup in [Pan+22]. Using the IPOPT solver provided by the CasADi library, we optimized the theta trajectory to minimize the particle travel time. This process allowed us to extract theta values, calculate particle positions, velocities, and accelerations, and perform numerical system inversion to determine the optimal trap positions.

In periodic modes, theta helps to analyze the periodic behaviour of the particle. By examining theta values over multiple cycles, patterns can be identified, and necessary adjustments can be made to improve the performance of the system.

In this study, we utilized two primary data frames: one containing measured data and the other containing trap positions and simulated data associated with the theta values. Theta values were generated corresponding to the count of measured data points, enabling a comprehensive comparison between the measured and simulated datasets.

4.5.1 Comparison and Selection Algorithm

After generating the theta values, the next step involved performing an algorithm to select, the closest point on the simulated data for each value in the measured data. This comparison and selection process is crucial for aligning the measured data with the simulated data and ensuring accurate analysis and optimization.

1. **Initialize Dataframes**
   We start by initializing the dataframes measured data and subset0_9 (a subset of measured data points)

2. **Define Chunk Size and Indices**
   The dataset was divided into chunks, each containing 24 data points representing one cycle of fish movement. The start and end indices were set based on the length of the dataset, and the total number of cycles was calculated.

```
chunk_size = 24
start_idx = 0
end_idx = len(selected_data3)
num_cycles = (end_idx - start_idx) // chunk_size
```

3. **Segmenting Data Points**
   The points within each cycle were categorized into five segments based on the provided batches: subset0_9, subset8_12, subset11_17, subset16_20, and subset19_24. These segments correspond to specific parts of the fish shape.

```
selected_points = {
    'subset0_9': [],
    'subset8_12': [],
    'subset11_17': [],
    'subset16_20': [],
    'subset19_24': []
}

cycle_count = 0
for i in range(num_cycles):
    cycle_start_idx = start_idx + i * chunk_size
    cycle_end_idx = min(start_idx + (i + 1) * chunk_size,
    end_idx)

    cycle_count += 1

    selected_points['subset0_9'].extend(range(cycle_start_idx +
    0, cycle_start_idx + 9))
    selected_points['subset8_12'].extend(range(cycle_start_idx +
     8, cycle_start_idx + 12))
    selected_points['subset11_17'].extend(range(cycle_start_idx
    + 11, cycle_start_idx + 17))
    selected_points['subset16_20'].extend(range(cycle_start_idx
    + 16, cycle_start_idx + 20))
    selected_points['subset19_24'].extend(range(cycle_start_idx
    + 19, cycle_start_idx + 24))
```

4. **Creating and Verifying DataFrames**
   Separate DataFrames are created for each segment. These points are plotted using distinct colors for visual differentiation, allowing for a clear examination of the bead movements.

```
selected_rows = {}
for key in selected_points:
    selected_rows[key] = measured_data.iloc[selected_points[key
    ]]

colors = {'subset0_9': 'blue', 'subset8_12': 'orange', '
    subset11_17': 'green', 'subset16_20': 'red', 'subset19_24': '
    purple'}
```

```
for key in selected_rows:
    plt.scatter(selected_rows[key]['measured_y'], selected_rows[key]['measured_z'], color=colors[key], s=1, label=key)
```

5. **Saving the Selected Points**
   The filtered points for each segment were saved in separate CSV files for further analysis and validation.

```
for key in selected_rows:
    selected_rows[key].to_csv(f'{key}_points.csv', index=True)
```

6. **Calculating Distances and Finding Nearest Points**
   For each segment, the Euclidean distance between each point in the measured data and all the points in the simulated data is calculated. This process involved concatenating the relevant ranges from the simulated data, computing the distances, and identifying the nearest simulated data points.

```
subset_df = pd.concat([simulated_df.iloc[:2730], simulated_df.iloc[9050:]])
distances = {}
nearest_indices = {}
nearest_rows = {}

for key in selected_rows:
    distances[key] = np.sqrt((selected_rows[key]['measured_y'].values[:, None] - subset_df['bead_y'].values) ** 2 +
                             (selected_rows[key]['measured_z'].values[:, None] - subset_df['bead_z'].values) ** 2)
    nearest_indices[key] = np.argmin(distances[key], axis=1)
    nearest_indices_adjusted[key] = np.where(nearest_indices[key] < 2730, nearest_indices[key], nearest_indices[key] + 9050)
    nearest_rows[key] = subset_df.iloc[nearest_indices_adjusted[key]]
```

   The nearest rows for each segment are printed for verification and further examination.

```
for key in nearest_rows:
    print(f"Nearest rows for {key}:")
    print(nearest_rows[key])
```

7. **Saving and Plotting Nearest Points**
   The nearest rows are saved in a CSV file. Subsequently, the selected points from the measured data and their corresponding nearest points from the simulated data were plotted for visual comparison. The plot in Fig. 4.13 helps verify the alignment and accuracy of the selected nearest points for subset19_24. This is

done for each of the subsets, and you can view the entire plot of fish in the Fig. 4.14

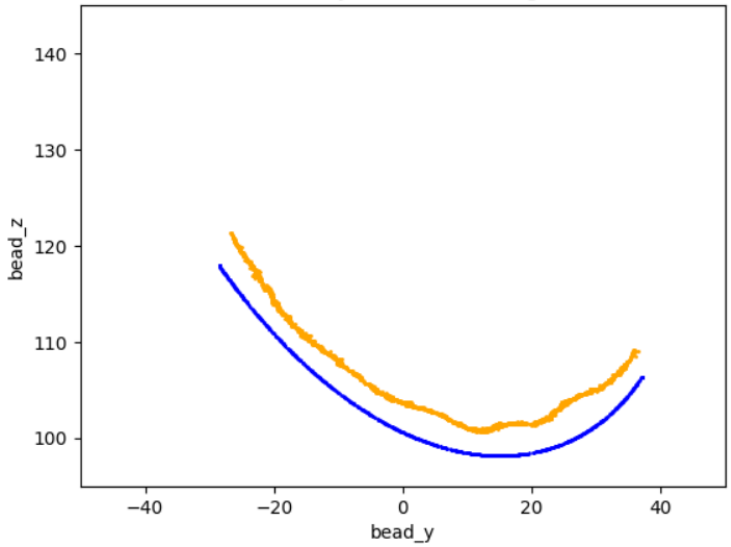


Fig. 4.13: Plot of measured data(orange) and nearest points from simulated data(blue) for the subset19_24

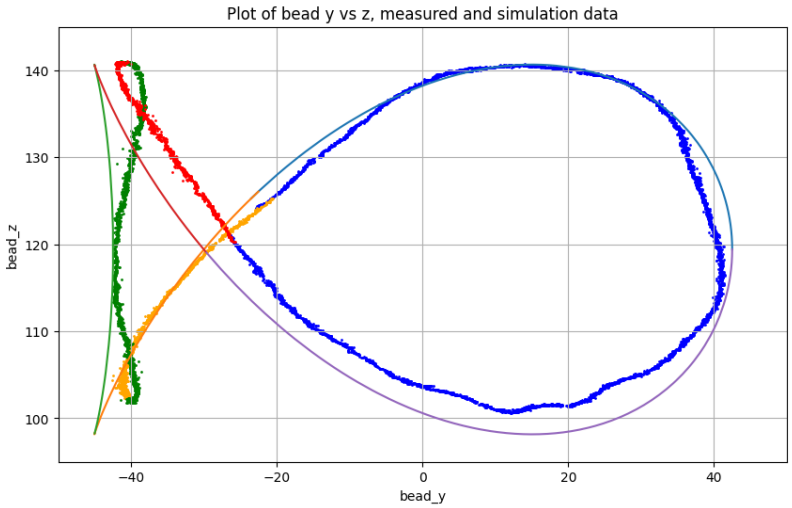


Fig. 4.14: Plot of measured data and their corresponding nearest points on the simulated dataset

A significant challenge encountered with the dataset pertained to the absence of simulated data points for the lower left tail loop of the fish shape. This discrepancy became apparent during the evaluation of Neural Network Model 1, as outlined in Section For a detailed explanation, refer to section 6.4. The resulting prediction data points are illustrated in 4.15

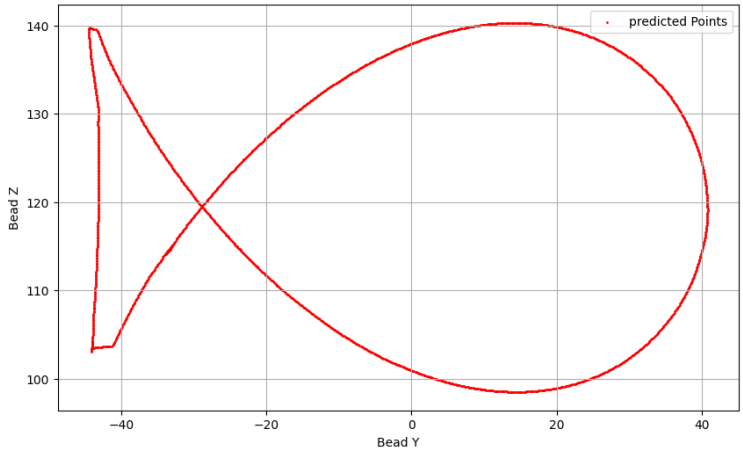


Fig. 4.15: The tail disjoint problem

To address this issue, a methodological approach was employed whereby the top part of the fish shape was mirrored along the initial position parallel to the X-axis. This technique effectively allowed the use of the upper tail loop data for the lower tail loop post-mirroring. The resultant dataset, depicted in Fig.4.16 ensures a comprehensive representation of the fish shape. This īs the final dataset and will be utilized in the subsequent chapters, ref Fig. 4.17.

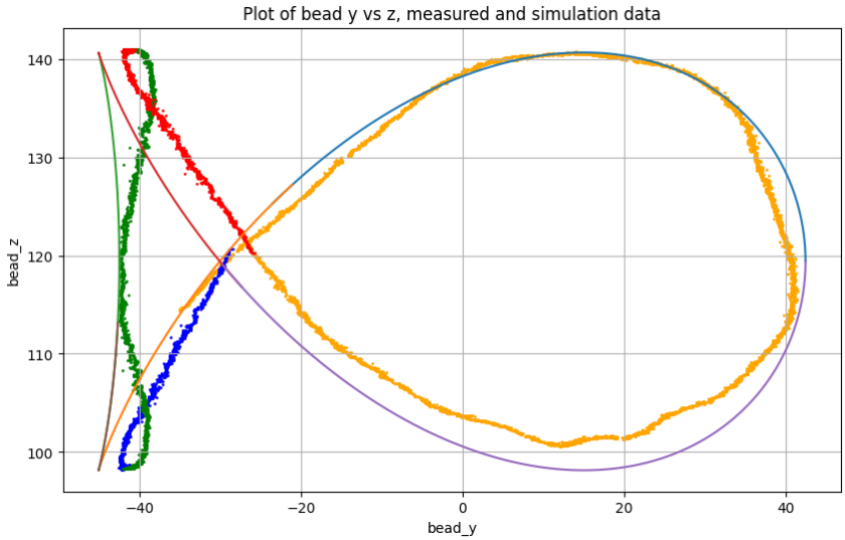


Fig. 4.16: Plot of measured and simulated data (mirrored tail loop)

measured_x	measured_y	measured_z	reference_x	reference_y	reference_z	traps_x	traps_y	traps_z	theta
1.26	40.655	119.407	0	42.5	119.4	0	40.7990404	119.40002	0
1.714	41.043	119.391	0	42.5	119.4	0	40.7990404	119.40002	0
1.254	40.605	119.42	0	42.4999835	119.4240863	0	40.7990238	119.4238182	0.0005667
1.269	40.567	119.452	0	42.4999341	119.4481726	0	40.7989739	119.4476164	0.0011335
1.5	40.479	119.494	0	42.4997364	119.4963451	0	40.7987745	119.4952126	0.002267
1.493	40.513	119.489	0	42.4997364	119.4963451	0	40.7987745	119.4952126	0.002267
1.225	40.519	119.496	0	42.4997364	119.4963451	0	40.7987745	119.4952126	0.002267
1.483	40.489	119.531	0	42.4994068	119.5445171	0	40.7984419	119.5428084	0.0034004
1.522	40.881	119.528	0	42.4994068	119.5445171	0	40.7984419	119.5428084	0.0034004
1.491	40.417	119.542	0	42.4991926	119.5686029	0	40.7982258	119.5666061	0.0039672
1.405	40.209	119.66	0	42.4976272	119.6890287	0	40.7966463	119.6855911	0.0068009
1.359	40.391	119.656	0	42.4976272	119.6890287	0	40.7966463	119.6855911	0.0068009
1.302	40.8	119.651	0	42.4976272	119.6890287	0	40.7966463	119.6855911	0.0068009
1.086	41.355	119.694	0	42.4972153	119.713113	0	40.7962306	119.7093873	0.0073676
1.455	40.353	119.741	0	42.4957817	119.7853639	0	40.794784	119.7807738	0.0090679
1.227	40.513	119.751	0	42.4957817	119.7853639	0	40.794784	119.7807738	0.0090679
1.517	41.182	119.751	0	42.4957817	119.7853639	0	40.794784	119.7807738	0.0090679
1.244	40.503	119.77	0	42.4952379	119.8094467	0	40.7942353	119.8045685	0.0096346
1.513	40.809	119.782	0	42.4946612	119.8335291	0	40.7936534	119.8283627	0.0102014
1.345	40.447	119.814	0	42.4940515	119.8576111	0	40.7930381	119.8521565	0.0107682
1.895	40.7	119.809	0	42.4940515	119.8576111	0	40.7930381	119.8521565	0.0107682
1.576	40.776	119.843	0	42.4934089	119.8816925	0	40.7923897	119.8759497	0.0113349
1.393	40.293	119.837	0	42.4927333	119.9057734	0	40.7917079	119.8997425	0.0119017
1.452	40.452	119.837	0	42.4927333	119.9057734	0	40.7917079	119.8997425	0.0119017
1.793	40.606	119.853	0	42.4927333	119.9057734	0	40.7917079	119.8997425	0.0119017
1.764	40.672	119.852	0	42.4927333	119.9057734	0	40.7917079	119.8997425	0.0119017

Fig. 4.17: Snapshot of final dataset

5 Experimental Setup

5.1 Hardware Components

The experimental setup for the OptiTrap system included several sophisticated hardware components designed to achieve precise acoustic levitation of particles. The primary components consisted of two opposed phased arrays, each containing 16×16 transducers. These arrays were positioned at a distance of 23.9 cm from each other, creating an acoustic field in the space between them. The transducers were controlled by a Field Programmable Gate Array (FPGA), which receives power from a dedicated power supply. The FPGA enables high-frequency updates and control signals that are essential for maintaining stable levitation and manipulation of particles.

The system includes three OptiTrack Prime 13 cameras, strategically positioned to cover the entire workspace where the particles are levitated. These cameras operate at a frequency of 240 Hz, providing high-resolution, real-time tracking of levitated particles. The position of a Styrofoam bead placed in the area between the two opposing transducers was captured by the cameras. The cameras send these positional data to the Motive application, which tracks and records each position in time. This setup ensured that the system could accurately monitor and adjust the position of the levitated particles and maintain precise control over their movements.

5.2 Software Environment

The software environment of the OptiTrap system is crucial for managing hardware components and ensuring precise control over the levitation process. The primary software tools used were Visual Studio and Motive.

Visual Studio was used to develop the control algorithms and interface with FPGA. Control software, written in C++, was developed and debugged within this integrated development environment (IDE). The C++ code in Visual Studio handles the

real-time control of transducer arrays, implementing algorithms that calculate the necessary acoustic fields to levitate and manipulate particles.

The Motive software application was connected to the three OptiTrack cameras. Motive is specialized software application designed for motion capture and real-time tracking. It seamlessly integrates with the OptiTrack system, providing advanced tools for capturing and analyzing the motion of levitated particles. Motive tracks the position of the styrofoam bead in real-time and records its movements. These positional data are then used to dynamically adjust the acoustic fields, ensuring that the particles follow the desired trajectories.

5.3 Calibration Procedures

Calibration is a vital step in the setup to ensure that the system operates accurately. The calibration process involves aligning the transducer arrays and synchronizing them with the motion capture system to ensure that the positional data are accurate and correspond correctly to the physical space.

Calibration in Motive:

Calibration in Motive is a multi-step process designed to ensure that the motion capture system can accurately track the positions of the particles within the acoustic levitation setup. The process begins with setting up OptiTrack cameras in the desired configuration, ensuring that they cover the entire working volume where the particles are levitated.

Through meticulous calibration using Motive, the OptiTrap system ensures that the positional data from the OptiTrack cameras are accurate and reliable. This accuracy is crucial for maintaining the stability of levitated particles and achieving precise control over their movements. The combination of hardware calibration, robust software environment, and thorough calibration procedures ensures that the OptiTrap system performs optimally, enabling advanced research and applications in acoustic levitation.

6 Methodology

6.1 Neural Network Models

Neural network models are powerful tools for modelling complex systems and improving their performance through data-driven approaches. In this study, various neural network architectures were considered for modelling the relationship between the trap positions, simulated positions, and measured positions.

Neural Network 1: Mapping Trap Positions to Measured Positions

The first neural network was designed to map current trap positions and θ to measured positions and θ. This network takes trap positions and theta as input and outputs the measured positions and theta, accurately modelling the system's behaviour and predicting bead positions based on current trap configurations.

The architecture of this neural network includes an input layer that receives trap positions (y,z) and theta. This is followed by several hidden layers to capture the complex relationships between the inputs and outputs. The output layer produces the measured positions (y,z) and theta.

The primary objective of this network is to accurately model the behaviour of the system and predict where the bead will be based on the current trap configuration.

Neural Network 2: Modifying Trap Positions to closely match Simulated Positions

The second neural network focuses on modifying the trap positions to minimize the deviation between the measured points (obtained from the first model) and simulated points. This network used a custom cost function to evaluate the loss between the measured and reference points and adjusted the trap positions accordingly. The

input to this network includes the measured positions and θ from Model 1, whereas the output includes the modified trap positions and θ.

The architecture of this neural network includes an input layer that receives the measured positions (y,z) and θ. This is followed by several hidden layers designed to capture the relationship between the inputs and desired trap modifications. The output layer produces modified trap positions (y,z) and θ.

The primary objective of this network was to minimize the deviation between the measured and simulated points and adjust the trap positions to bring the measured points closer to the simulated points. By optimizing this model, we aimed to reduce the loss function, which is defined as the deviation between the measured and simulated points.

Iterative Optimization Process

The optimization process involves updating the trap positions using two neural networks. The process starts with the current trap positions and θ. The first neural network, Model 1, predicted the measured positions based on the current trap positions and θ. The second neural network, Model 2, adjusts the trap positions as per the measured positions received from Model 1, by minimizing the loss between the predicted measured positions and reference positions. The modified trap positions and θ were then fed back into the Model 1, which gives us new measured positions which are the final measurements. As model 1 mimics the real world behaviour of the levitating particles on the OptiTrap, this final measurements gives us an idea of the path of levitated particles on the system if we run the levitator with the adjusted trap positions.

For adjusting the trap positions in Model 2, we have three approaches: the Baseline approach, the Static Theta Static Reference Positions (STSR), and the Static Theta Dynamic Reference Positions (STDR). The key differentiating factor among these approaches is how the deviation in the custom cost function is calculated.

Baseline Model:

The Baseline Model uses reference positions directly from features of Model 2 to calculate the deviation. This approach involves comparing these reference positions with the predicted measured positions from Model 1 to determine the deviation.

Static Theta Static Reference Positions (STSR):

In the STSR approach, the reference positions are calculated from the θ using predefined formulae for the shape of the fish, refer 6.1 and 6.2. The deviation is then calculated using these reference positions against the predicted measured positions

from Model 1. In this model, the calculation of the reference positions is performed outside the custom cost function.

$$q_y = \cos(\theta) - \frac{\sin^2(\theta)}{\sqrt{2}} \tag{6.1}$$

$$q_z = \sin(\theta)\cos(\theta) \tag{6.2}$$

Static Theta Dynamic Reference Positions (STDR):
The Static Theta Dynamic Reference Positions approach also calculates reference positions from the theta using the formulae for the shape of the fish. However, the key difference is that the calculation of reference positions is performed inside the custom cost function. This allows for a dynamic adjustment of reference positions during the optimization process, leading to potentially more accurate adjustments of trap positions.

We also incorporate a scaled version for each of the above approaches.

6.2 Feature Extraction

Neural Network Models Overview:

- **Model 1:**
  Input Features: traps_y, traps_z, θ
  Output Features: measured_y, measured_z, θ
  Objective: Train the network to predict the measured positions based on the trap positions and theta, modelling the real-world behaviour of the particles.

- **Model 2:**
  Input Features: measured_y, measured_z, θ
  Output Features: traps_y, traps_z, reference_y, reference_z, θ
  Objective: Train the network to adjust the trap positions to minimize the deviation between the measured and reference points, thereby ensuring that the particles follow the desired paths accurately.

We decided to omit the inclusion of x-coordinates from all positions, as its value is always 0. The volumetric display of fish on the levitator is always on the yz-plane; furthermore, this limitation also applies to any other 2D shape.

6.3 Evaluation Metrics

Evaluation metrics are crucial for assessing the performance and effectiveness of the neural network models developed in this thesis. This section details the metrics used to evaluate both Model 1 and Model 2, providing insights into their accuracy, efficiency, and robustness.

- **Mean Squared Error (MSE):** MSE measures the average squared difference between the predicted and actual values. It provides a measure of the model's accuracy, with lower values indicating better performance.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6.3}$$

  MSE in Model 1 is used to evaluate the difference between the predicted bead positions and the actual measured positions. MSE in Model 2 helps evaluate the effectiveness of the trap position adjustments in reducing deviations.

- **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted and actual values. It is less sensitive to outliers compared to MSE.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.4}$$

  MAE provides a straightforward interpretation of prediction accuracy by indicating the average error magnitude. MAE provides an intuitive measure of the optimization accuracy.

- **R-squared (R^2):** R^2 indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating better model performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6.5}$$

  R^2 in Model 1 is used to assess how well the predicted bead positions match the actual measured positions. R^2 in Model 2 helps assess the model's capability to produce trap positions that closely follow the reference trajectories.

- **Training Time:** The total time taken to train the model provides an indication of the computational efficiency. Training time helps evaluate the practicality of the model, especially for large datasets and real-time applications.

- **Inference Time:** The time taken to make predictions on the test set, measured per sample. Inference time is critical for applications requiring real-time predictions.

- **Custom Cost Function:** The custom cost function is designed to minimize the deviation between the predicted bead positions (from Model 1) and the true reference positions. It focuses on optimizing trap positions to achieve accurate particle trajectories. This cost function directly influences the training process of Model 2, guiding it to adjust the trap positions for better alignment with the reference positions.

- **Visualization of Predicted Particles:** Visualizations provide an intuitive understanding of the model's performance, highlighting areas of success and regions requiring further improvement.

6.4 Neural Network 1: Mapping Trap Positions to Measured Positions

6.4.1 Model Selection:

The selection of neural network models was based on their ability to handle time-series data and capture complex relationships between input features. Several models were considered, each offering unique advantages:

- LSTM: Known for its ability to capture long-term dependencies and handle sequential data, making it suitable for time-series prediction tasks.
- CNN: Effective in capturing spatial relationships and patterns within the data, often used in image recognition but also applicable to time-series data.
- Feedforward NN: A simpler architecture that is easier to train and interpret, suitable for tasks where complex temporal dependencies are not critical.
- GRU: Similar to LSTM but with a simpler structure, often leading to faster training times while maintaining the ability to capture dependencies in sequential data.

To evaluate the performance of different neural network models for predicting particle positions in acoustophoretic volumetric displays, we employed a decision matrix approach. This method allowed us to systematically compare the models based on several key criteria, ensuring a comprehensive assessment of each model's strengths and weaknesses. The criteria used in our decision matrix include accuracy, training time, and inference time.

The following table 6.1 summarizes the performance metrics for the four neural network architectures evaluated: Long Short-Term Memory (LSTM), Feedforward Neural Network (Feedforward NN), Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN) :

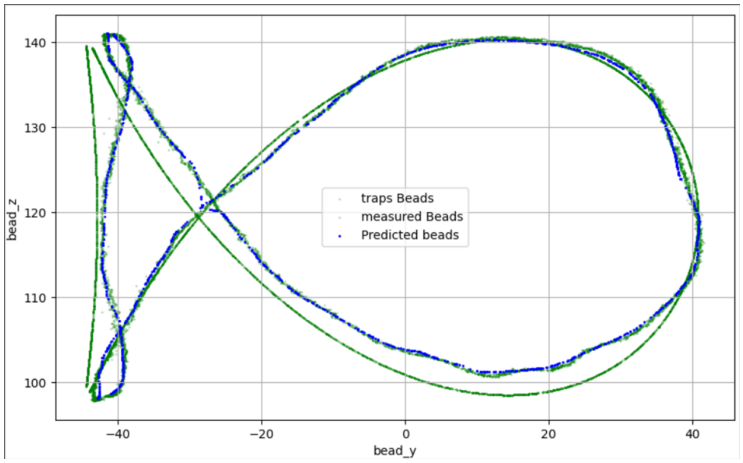
Model	Training Time (s)	Inference Time(s/sample)	R-squared	MSE	MAE
LSTM	82.58	0.2715	0.99936	0.0062	0.0410
Feedforward	94.70	0.2557	0.9930	0.0065	0.0404
GRU	85.21	0.2586	0.9930	0.0065	0.0430
CNN	57.70	0.1790	0.9931	0.0065	0.0392

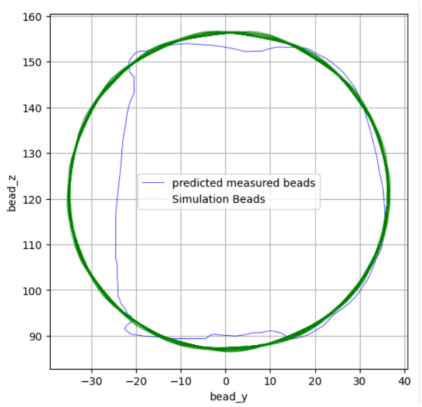
Tab. 6.1: Performance metrics of various neural network models for the Model 1

The visual analysis of the models' predictions compared to the actual positions of the levitated particles for different shapes (fish, circle, heart) further highlights the models' robustness and ability to generalize beyond the training data.

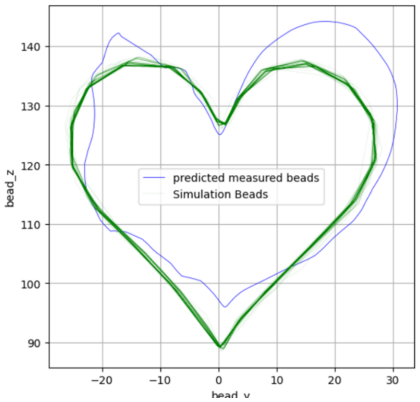
LSTM (Long Short-Term Memory): LSTM networks are known for their ability to capture long-term dependencies and handle sequential data, making them suitable for time-series prediction tasks. The LSTM model required 82.58 seconds for training, reflecting its complexity and capability to learn intricate temporal patterns effectively. It had an inference time of 0.2715 seconds per sample. The LSTM achieved the highest R-squared value of 0.99936, indicating that it explains nearly all the variance in the target variable. Its MSE was 0.0062, and the MAE was 0.0410, showing excellent accuracy and minimal deviation from actual values.

Fig. 6.1: Traps predictions of fish shape from LSTM model





(a) Circle



(b) Heart

Fig. 6.2: Plots of Circle and Heart from LSTM model

In terms of shape prediction:

- Fish Shape: Predicted positions closely follow the actual positions with minor deviations.

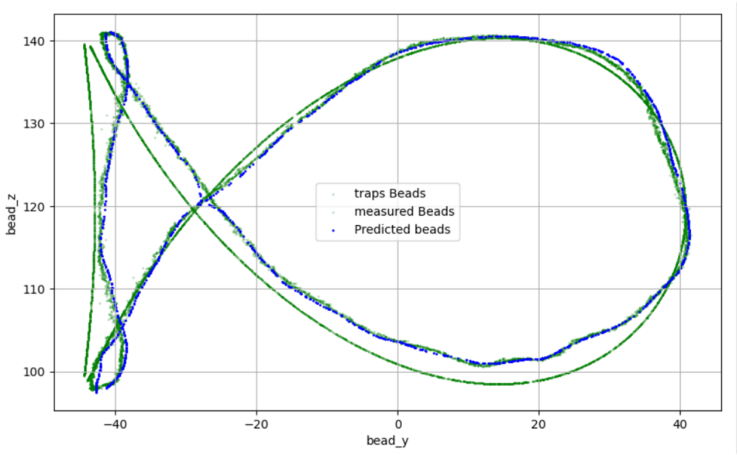
- Circle Shape: Some deviations from actual positions, indicating potential challenges in generalization.

- Heart Shape: Close alignment with actual positions, but noticeable deviations in some areas.

Feedforward Neural Network (FFNN)

Feedforward NNs have a simpler architecture that is easier to train and interpret, suitable for tasks where complex temporal dependencies are not critical. The Feedforward NN took 94.70 seconds for training, indicating a higher computational requirement due to its architecture. It had an inference time of 0.2557 seconds per sample. The model achieved an R-squared value of 0.9930, suggesting strong performance. Its MSE was 0.0065, and it had an MAE of 0.0404, indicating very accurate predictions.

Fig. 6.3: Traps predictions of fish shape from FFN model



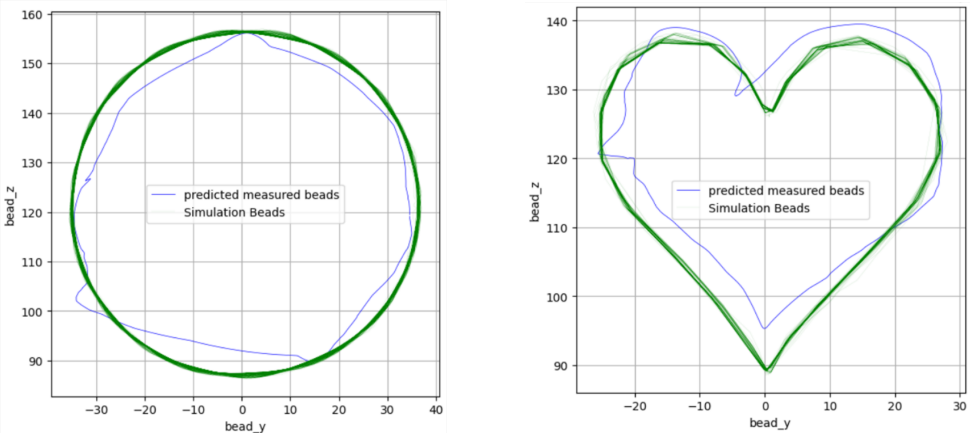
In terms of shape prediction:

- Fish Shape: High accuracy with predictions closely matching actual positions.

- Circle Shape: Relatively accurate but shows some deviations.

- Heart Shape: Good alignment with actual positions, minor deviations present.

GRU (Gated Recurrent Unit)

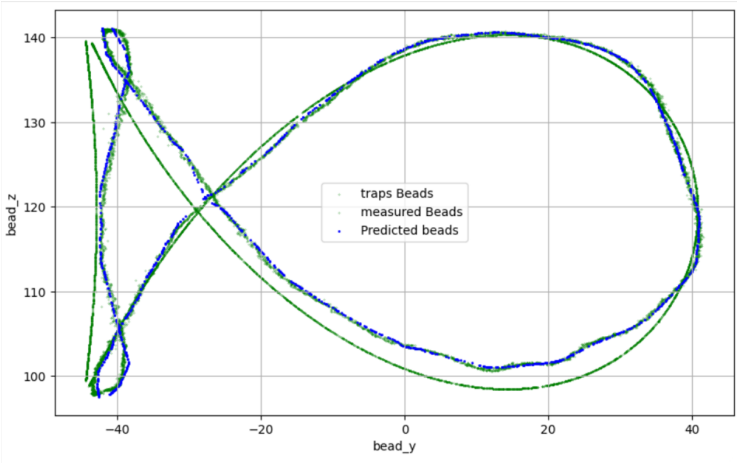


(a) Circle **(b)** Heart

Fig. 6.4: Plots of Circle and Heart from FNN model

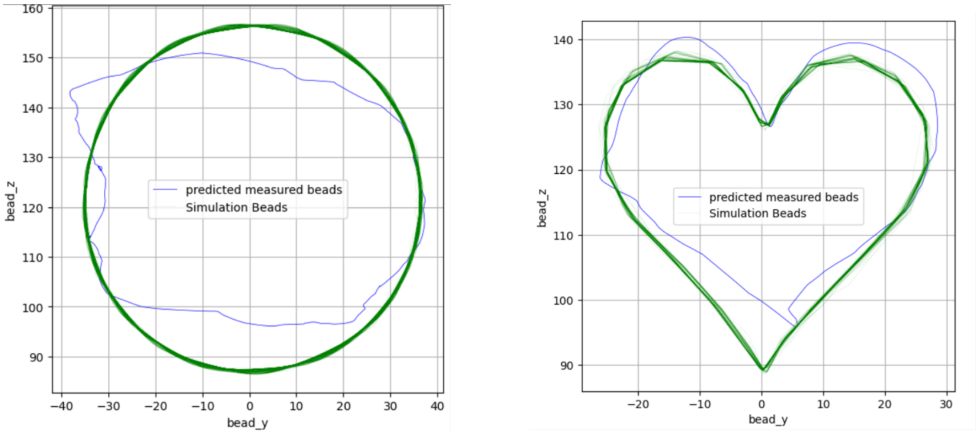
GRUs are similar to LSTMs but have a simpler structure, often leading to faster training times while maintaining the ability to capture dependencies in sequential data. The GRU model required 85.21 seconds for training, showing a balance between complexity and efficiency. It had an inference time of 0.2586 seconds per sample. The GRU achieved an R-squared value of 0.9930, similar to the Feedforward NN. Its MSE was 0.0065, and the MAE was 0.0430, indicating high accuracy.

Fig. 6.5: Traps predictions of fish shape from GRU model



In terms of shape prediction:

- Fish Shape: High accuracy with minor deviations from actual positions.
- Circle Shape: Accurate but with some noticeable deviations.
- Heart Shape: Good alignment with actual positions, few deviations.



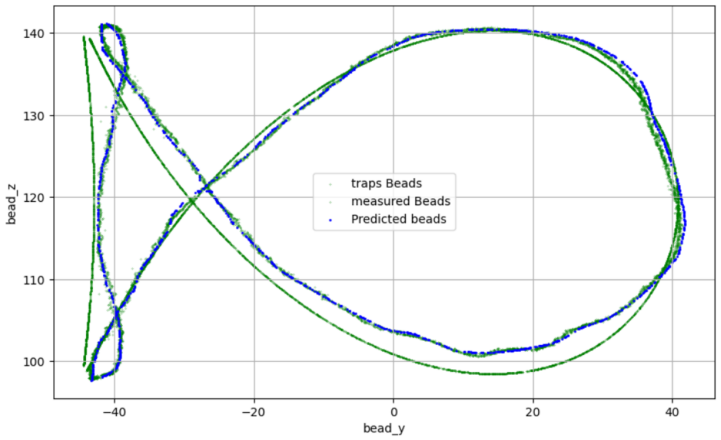
(a) Circle (b) Heart

Fig. 6.6: Plots of Circle and Heart from GRU model

CNN (Convolutional Neural Network)

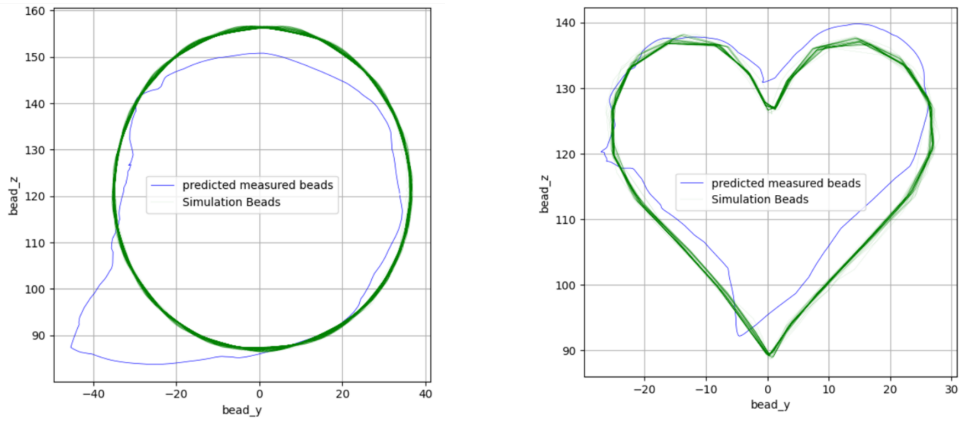
CNNs are effective in capturing spatial relationships and patterns within the data, often used in image recognition but also applicable to time-series data. The CNN model was the fastest in terms of training time at 57.70 seconds. It also had the fastest inference time at 0.1790 seconds per sample, making it suitable for real-time predictions. The CNN achieved an R-squared value of 0.9931, indicating strong performance. Its MSE was 0.0065, and it had the lowest MAE among all models at 0.0392, indicating the best performance in terms of prediction accuracy.

Fig. 6.7: Traps predictions of fish shape from CNN model



In terms of shape prediction:

- Fish Shape: Predicted positions closely follow actual positions with minor deviations.



(a) Circle **(b)** Heart

Fig. 6.8: Plots of Circle and Heart from CNN model

- Circle Shape: Accurate but shows some deviations.
- Heart Shape: Good alignment with actual positions, minor deviations present.

Based on the decision matrix, the LSTM model emerges as the most effective for capturing complex temporal dependencies and achieving high accuracy. However, the CNN model is notable for its efficiency, demonstrating the fastest training and inference times, and the lowest MAE. The Feedforward Neural Network and GRU models also show strong performance, with slightly higher training and inference times.

After evaluating the strengths and weaknesses of each model, the LSTM model was selected for its superior performance in handling temporal dependencies, which is crucial for this task.

6.4.2 Model Training:

The training process involves several steps, from data preparation to model evaluation. The following subsections detail these steps:

Data Preparation:
The dataset consists of measured points, trap positions, reference points, and theta values. For Model 1, the input features are traps_y, traps_z, and θ, while the output features are measured_y, measured_z, and θ.

To prepare the data for training, the following steps are taken:

- Scaling: StandardScaler was considered for scaling the input and output features to ensure consistency in feature ranges.

- Reshaping: The input features are reshaped to match the required input shape for the LSTM layers, which expect a 3D input (batch_size, time steps, features). The appropriate shape for the input to model1 is (batch_size, 1, 3), where batch_size is the number of samples in the batch, 1 represents the time step dimension for the LSTM, and 3 corresponds to the three features: traps_y, traps_z, and θ.

- Train-Test Split: The dataset is divided into training and testing sets, ensuring that both the input and output data are appropriately separated for robust model evaluation. We consider a test dataset size of 0.30 for both - the unscaled and scaled versions of model 1.

```
scaler1 = StandardScaler()

input_train_scaled = scaler1.fit_transform(input_train)
output_train_scaled = scaler1.transform(output_train)

input_train_scaled = input_train_scaled.reshape((input_train_scaled.
    shape[0], 1, input_train_scaled.shape[1]))
```

Model Architecture:

1. Input Layer: The LSTM model expects input in the form of 3D tensors with shape (batch_size, timesteps, features). For model1, the input shape is specified as (1, 3), meaning each input sequence consists of a single timestep with three features - traps_y, traps_z, θ. This setup is suited for time-series data where each input sample is represented by a vector of three features.

2. LSTM Layer:

   - Type: Long Short-Term Memory (LSTM) layer.

   - Units: 64 units. The number of units in an LSTM layer determines the dimensionality of the output space. Each unit is responsible for capturing and maintaining information over time, allowing the model to learn from sequential dependencies in the data.

   - Activation Function: ReLU (Rectified Linear Unit). ReLU introduces non-linearity to the model, helping it to learn complex patterns. Although LSTMs have their own activation functions for gating mechanisms, the ReLU activation function is used in the dense layers that follow.

   - Return Sequences: False. This parameter is set to False, indicating that the LSTM layer will only output the final state of the sequence rather than the output for each timestep. This configuration is suitable when the model needs only the final representation of the sequence to make predictions.

3. Dense Layers:

   - First Dense Layer:

     – Units: 64. This dense layer has 64 units, allowing the model to learn more complex representations of the features extracted by the LSTM layer.

     – Activation Function: ReLU. The ReLU activation function is again used here to introduce non-linearity and enable the model to capture complex patterns in the data.

   - Output Dense Layer:

     – Units: The number of units in the output dense layer equals the number of target variables,i.e.3 - measured_y, measured_z, θ. In this case, it matches the number of columns in output_train, which represents the number of features in the output. This layer directly maps the learned features to the target space.

– Activation Function: No activation function is applied. This is typical in regression tasks where the output layer is linear to ensure predictions are continuous and not bounded by any activation function.

```
model1 = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(64, activation='relu', input_shape=(1, 3),
    return_sequences=False),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(output_train.shape[1])
])
```

Compilation and Training:

- Loss Function: Mean Squared Error (MSE). MSE is used as the loss function for regression tasks. The mean squared error is chosen for its effectiveness in regression tasks, where the goal is to minimize the difference between the predicted and actual values.

- Optimizer: Adam. The Adam optimizer is employed to minimize the loss function. The Adam optimizer is selected for its adaptive learning rate capabilities, which help in achieving faster convergence.

```
model1.compile(loss='mean_squared_error', optimizer='adam')
```

Training Process:

- Epochs: The model is trained for 200 epochs. An epoch is one complete pass through the entire training dataset. Training for multiple epochs allows the model to iteratively improve its performance by adjusting its parameters.

- Validation Split: 0.1. During training, 10% of the data is reserved for validation. This data is used to evaluate the model's performance on unseen data after each epoch. It helps to monitor the model's performance and prevent overfitting.

- Early Stopping: The early stopping callback monitors the validation loss. If no improvement is observed for 50 consecutive epochs (patience=20), training is stopped early to avoid overfitting and save computational resources. The best weights from the epoch with the lowest validation loss are restored.

```
early_stopping_callback = callbacks.EarlyStopping(monitor='val_loss',
    patience=20, restore_best_weights=True)

history = model1.fit(X_train, y_train, epochs=200, verbose=1,
   validation_split=0.1, callbacks=[early_stopping_callback])
```

Prediction and Post-Processing

```
predictions = model1.predict(X_test)
predictions_inverse_scaled = scaler1.inverse_transform(predictions)
```

- Predictions: After training, the model generates predictions on new input data. The model1.predict method is used for this purpose.

- Inverse Transformation: To convert the scaled predictions back to the original scale of the data, an inverse transformation is performed using the scaler1.inverse_transform method. This step is crucial for interpreting the predictions in the context of the original data.

6.5 Neural Network 2: Modifying Trap Positions to closely match Simulated Positions

6.5.1 Model Selection:

The selection of Model 2 focuses on leveraging the strengths of Long Short-Term Memory (LSTM) networks to handle sequential data and capture complex temporal relationships. The LSTM model is particularly suitable for this task due to its ability to learn from sequences and maintain information over long periods, which is essential for understanding the dynamics of particle positions over time. The interdependency and data flow between Model 1 and Model 2 further strengthens the case for using the LSTM network in Model 2. The following points justify this:

- **Consistency in Sequential Data Handling**

  Given that Model 1 employs an LSTM network to predict measured positions, it is prudent to ensure consistency in handling sequential data by utilizing an LSTM in Model 2 as well. The inherent sequential nature of the data, encompassing trap positions, measured positions, and theta, is adeptly managed by LSTM networks in both models, thereby fostering improved overall performance and coherence in predictions.

- **Interdependency of Predictions**

  Model 2 is inherently dependent on the predictions generated by Model 1, as these predictions serve as inputs for Model 2. Furthermore, the custom cost function in Model 2 invokes the ‘predict‘ method on Model 1 during its operation. This interdependency highlights the significance of Model 1’s prediction accuracy and quality.

- **Feedback Loop**

  A crucial aspect of the data flow between these models is the feedback loop, where the predictions from Model 2 are utilised again in Model 1 to derive the final measured points predictions. This recursive interaction between the two models implies that any enhancements made to Model 1’s architecture will directly benefit Model 2 and vice versa. Utilising LSTM networks in both models guarantees that improvements in sequential data handling and prediction accuracy are perpetuated through the entire pipeline, thereby enhancing overall performance.

- **Complex Pattern Recognition**

  Both Model 1 and Model 2 are tasked with recognizing and modelling complex, non-linear relationships within the data. The LSTM network's proficiency in handling such complexity makes it an ideal candidate for use in Model 2. By employing LSTM networks in both models, the intricate dependencies between the input features and the output predictions are captured more accurately, which is pivotal for minimizing the deviation between measured and simulated positions.

- **Robustness and Generalization**

  The implementation of LSTM networks in both models ensures robustness and improved generalization across various datasets. The LSTM's ability to effectively manage noisy and irregular data enhances the reliability and accuracy of Model 2's predictions, which is critical given that these predictions are subsequently utilized in Model 1.

Given these considerations, it is evident that using LSTM networks in both Model 1 and Model 2 is not only justified but also advantageous. The LSTM's proven performance in managing sequential data, modelling complex patterns, and robustness to noise makes it an ideal choice, ensuring consistency and improved overall performance in predicting particle positions within acoustophoretic volumetric displays.

6.5.2 Model Training:

The training process for Model 2 involves several steps, including data preparation, model architecture configuration, and the implementation of a custom cost function. These steps ensure that the model can effectively learn to adjust the trap positions to minimize the deviation from the simulated positions.

Data Preparation: For Model 2, the input features are measured_y, measured_z, and theta, with the output features being traps_y, traps_z, reference_y, reference_z, and θ.

To prepare the data for training, the following steps are taken:

- Scaling: The input and output features are scaled using StandardScaler to ensure consistency in feature ranges. This scaling helps the model to learn effectively by normalizing the data.

- Reshaping: The input features are reshaped to match the required input shape for the LSTM layer, which expects a 3D input (samples, time steps, features).

- Train-Test Split: We consider a test dataset size of 0.15 for both scaled and unscaled versions of Model 2.

Model Architecture:

The architecture of each model consists of several layers of 64 neurons each, designed to process the input features and produce accurate predictions:

- LSTM Layer: The initial LSTM layer processes the input features (‘measured_y‘, ‘measured_z‘, and ‘θ‘), capturing temporal dependencies and relationships.

- Dense Layers: Two subsequent dense layers, each with 64 neurons, further refine the features extracted by the LSTM layer. These layers help in learning complex mappings between the input features and the desired output features.

- Output Layer: The final dense layer outputs the predicted values for ‘traps_y‘, ‘traps_z‘, ‘simulated_y‘, ‘simulated_z‘, and ‘theta‘. Out of which, we are interested in the columns ‘traps_y‘, ‘traps_z‘ and θ.

Compilation and Training:

The model is compiled using a custom cost function specifically designed for this task. The custom cost function calculates the deviation between the predicted bead positions from Model 1 and the true reference positions. This deviation is minimized to ensure that the optimized trap positions closely match the desired reference positions.

- Custom Cost Function: The cost function extracts the predicted trap positions and theta from the output of Model 2, then uses these values as input to Model 1 to predict the bead positions. The loss is calculated as the mean absolute deviation between the predicted bead positions and the true reference positions. For the 3 approaches of Model 2, the method of calculating deviation differs. The below code represents the custom cost function for the Baseline model.

```
def custom_cost_function(y_true_nn2, y_pred_nn2):
    batch_size = tf.shape(y_pred_nn2)[0]

    # Extract theta and predicted positions from y_pred
    traps_pred = y_pred_nn2[:, 0:2]
```

```
    theta_pred = y_pred_nn2[:, 4:5]

    # Reshape to match the input shape for model1
    input_for_nn1 = tf.concat([traps_pred, theta_pred], axis=1)
    input_for_nn1 = tf.expand_dims(input_for_nn1, axis=1)  # Reshape

    # Get predicted bead positions from model1
    predicted_bead_positions = model1(input_for_nn1, training=False)

    # Extract true reference_y and reference_z from y_true
    true_reference_positions = y_true_nn2[:, 2:5]

    # Calculate loss
    deviation = tf.reduce_mean(tf.reduce_sum(tf.abs(predicted_bead_positions - true_reference_positions), axis=1))

    return deviation
```

- Early Stopping: Early stopping is implemented to prevent overfitting by monitoring the validation loss and stopping the training process if the loss does not improve for a specified number of epochs. This ensures that the model maintains its generalization capability.

The model is trained using the Adam optimizer, known for its efficient gradient-based optimization. The training process involves multiple epochs, during which the model continuously learns to adjust the trap positions to minimize the deviation from the reference positions. Once we receive the trap predictions from Model 2 for the corresponding measured positions (from Model 1), we plot them with the Original Trap Positions. In the next stage, we also use these Predicted Traps in Model 1 to finally obtain the Prediction for Measured Positions, which depicts the real-world behaviour of the bead for the predicted traps. We also plot these predicted measured positions against actual reference positions to analyse them. Visualizations of all the tried models are explained in the next chapter.

7 Results and Analysis

7.1 Developmental Stages of Neural Network Models - Qualitative Performance Analysis

The outline of Model 1 remains same for the various developmental stages of Model 2, we have used scaled and unscaled versions of Model 1 depending on the the type of Model 2 network. For Model 2, we have presented the different advancements in the Neural Network Configurations.

7.1.1 Baseline Model

Model Training and Validation Loss for Model 1

In Fig.7.1, both training and validation losses decrease significantly over the epochs, indicating effective learning. The convergence of the training and validation losses towards lower values suggests good generalization. Minor fluctuations in the validation loss may be due to overfitting, but overall, model 1 shows stable training performance.

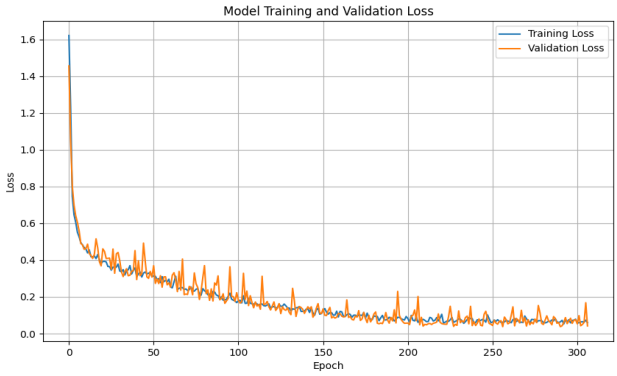


Fig. 7.1: Model Training and Validation Loss (Baseline Model)

Model Training and Validation Loss for Model 2

The plot in fig 7.2 demonstrates that Model 2 effectively learns the underlying patterns in the data without significant overfitting. The initial rapid decrease in loss followed by stabilization indicates that the model converges well. The close alignment of training and validation loss curves suggests good generalization capability. These observations collectively point to the robustness and reliability of Model 2 in predicting particle positions accurately.

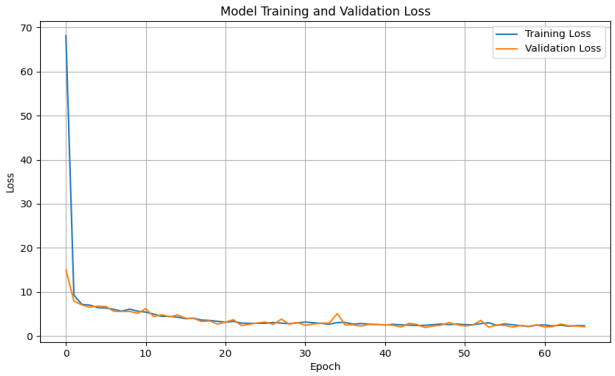


Fig. 7.2: Model Training and Validation Loss Model 2 (Baseline Model)

Predicted vs Actual Traps

In the Predicted Vs Actual Traps, refer fig 7.3, the alignment of the predicted traps (blue) with the actual traps (orange) and the actual reference positions (black) indicates that the model's predictions closely match the actual trap positions and reference positions. The close overlap of these points demonstrates the model's high accuracy in predicting the trap positions.

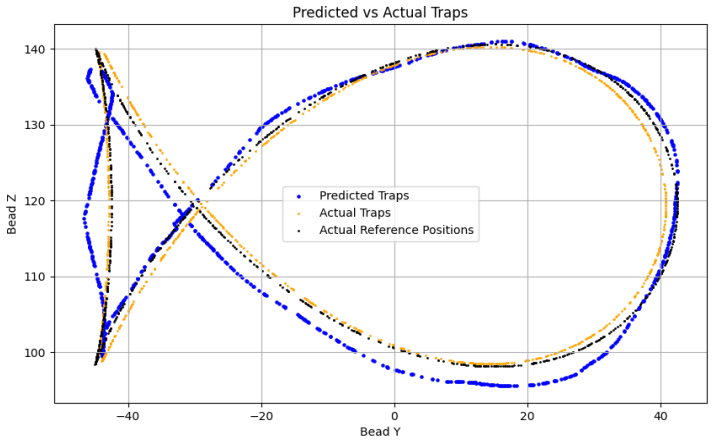


Fig. 7.3: Predicted vs Actual Traps (Baseline Model)

Predicted Final Measured Positions

The close alignment of the predicted final measured positions (green) and predicted trap positions (blue) with the actual reference positions (black) suggests that the model accurately predicts both the final measured positions and the trap positions relative to the reference positions.

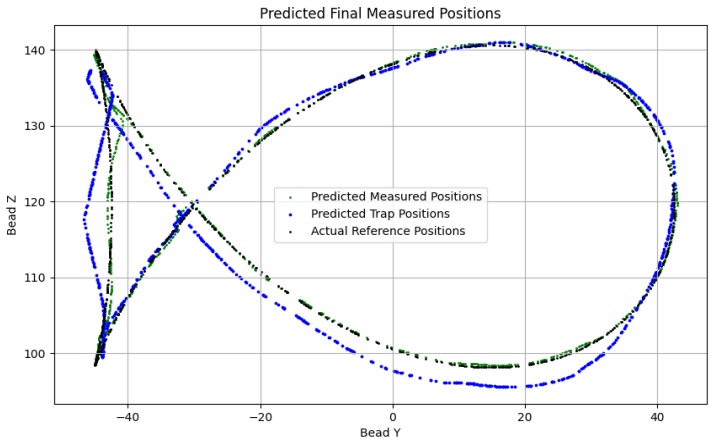
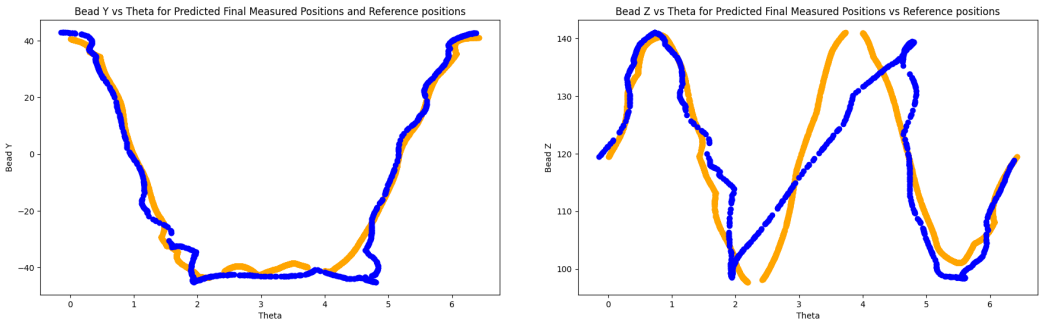


Fig. 7.4: Predicted Final Measured Positions (Baseline Model)

Plot of Bead Y and Z vs theta

The predicted beads closely follow the reference positions, demonstrating the model's ability to accurately predict the bead Y positions based on the theta values. Minor deviations between the lines indicate areas where the model's predictions could be further refined, refer fig 7.5



(a) Bead Y vs Theta **(b)** Bead Z vs Theta

Fig. 7.5: Plot of Bead Y and Z vs Theta (Baseline Model)

Similar to the Bead Y vs theta plot, the predicted positions align closely with the reference positions, indicating that the model effectively captures the relationship

between Bead Z positions and theta. Deviations between the lines highlight areas where model performance can be improved.

7.1.2 Scaled Baseline Model

Model Training and Validation Loss

The loss for both training and validation sets decreases rapidly during the initial epochs and stabilizes at a low value, refer fig 7.6. This indicates that Model 1 is effectively learning the relationships in the data without overfitting as there is close alignment of thee training and validation loss curves.

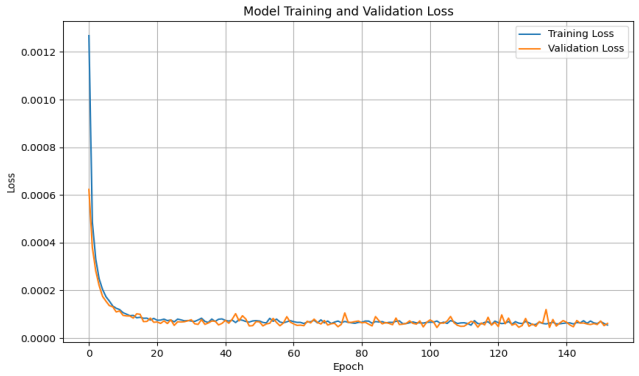


Fig. 7.6: Model Training and Validation Loss (Scaled Baseline Model)

Model Training and Validation Loss Model 2

Similar to Model 1, the training and validation loss curves for Model 2 show a rapid decline initially and then stabilize, maintaining low values. The close proximity of the training and validation loss curves indicates that Model 2 is well-optimized and not overfitting.

Fig. 7.7: Model Training and Validation Loss Model 2 (Scaled Baseline Model)

Predicted vs Actual Traps

The close alignment of the predicted trap positions (blue) with the actual trap positions (orange) and the reference positions (black) suggests that the model is accurately predicting the trap positions.

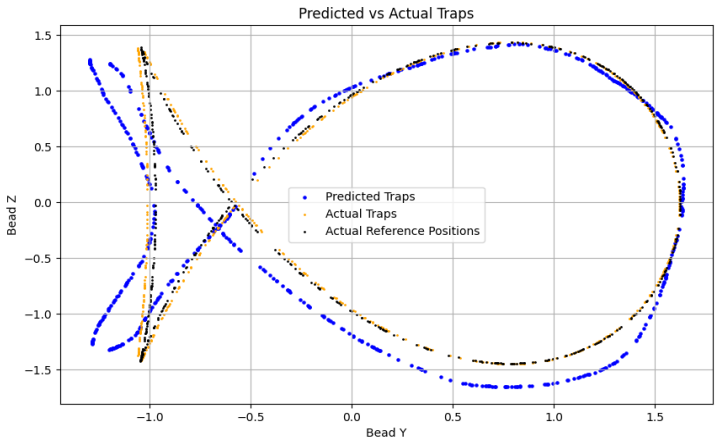


Fig. 7.8: Predicted vs Actual Traps (Scaled Baseline Model)

Predicted Final Measured Positions

This plot visualizes the final measured positions predicted by the model against the actual reference positions. The predicted measured positions (green) closely follow the actual reference positions (black), indicating high accuracy in the model's predictions.

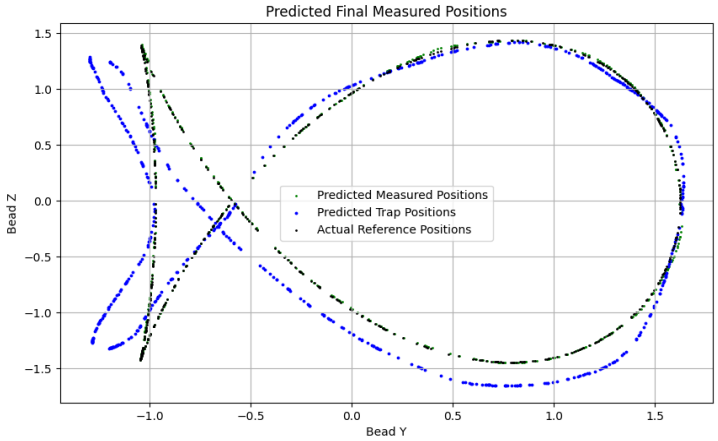
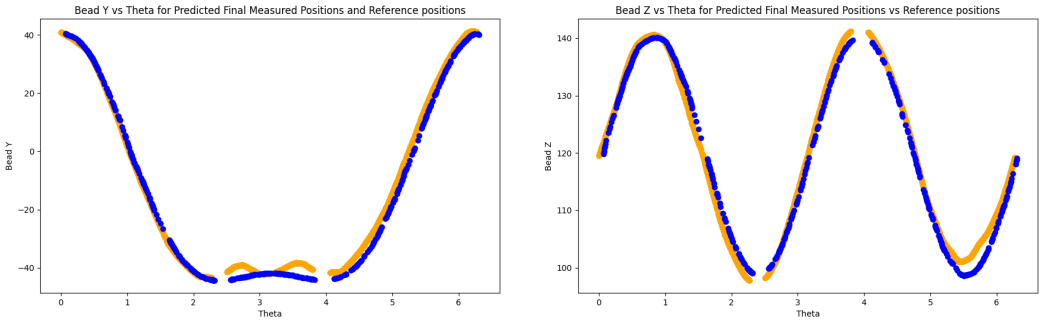


Fig. 7.9: Predicted Final Measured Positions (Scaled Baseline Model)

Plot of Bead Y and Z vs Theta The predicted positions closely follow the reference positions, suggesting that the model accurately predicts Bead Y across the

range of Theta values. The consistency of the data points along the reference line demonstrates the model's reliability in this aspect. Similar to the previous plot, the predicted positions align well with the reference positions. The model successfully captures the Bead Z values across different Theta values. The close adherence of the predicted points to the reference line indicates that the model is effective in predicting Bead Z as well.



(a) Bead Y vs Theta **(b)** Bead Z vs Theta

Fig. 7.10: Plot of Bead Y and Z vs Theta (Scaled Baseline Model)

7.1.3 Static Theta Static Reference Positions

Model Training and Validation Loss

This plot illustrates the training and validation loss over 250 epochs. The rapid initial decline in loss values indicates that the model quickly learns the underlying patterns in the data. Both training and validation losses continue to decrease gradually, with minimal divergence between them, suggesting the model generalizes well to unseen data.

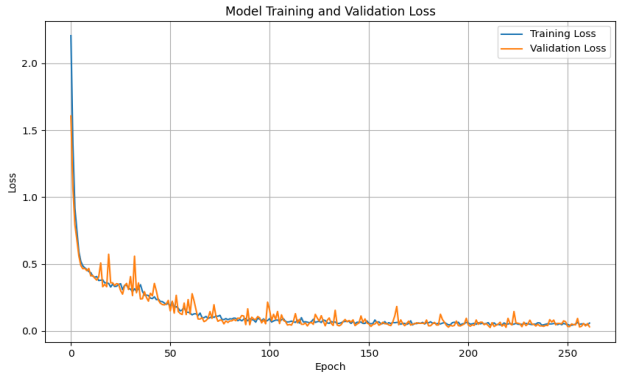


Fig. 7.11: Model Training and Validation Loss (Static Theta Static Reference Positions)

Model Training and Validation Loss Model 2

The plot indicates that both the training and validation loss decreased rapidly within the first few epochs, stabilizing around epoch 10. The losses continue to decrease at a slower rate, showing a consistent trend with slight fluctuations. This suggests that the model is learning effectively and achieving a good fit to the training data, with the validation loss closely tracking the training loss, indicating minimal overfitting.

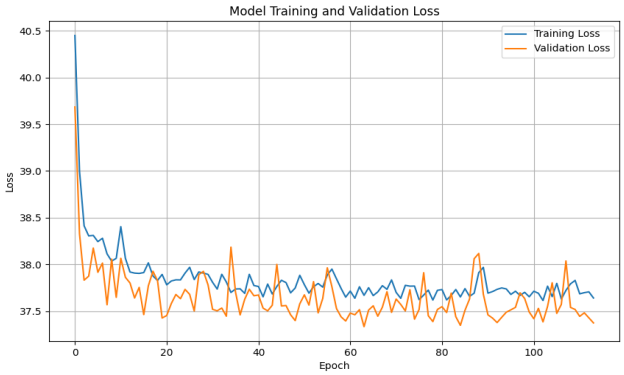


Fig. 7.12: Model Training and Validation Loss Model 2 (Static Theta Static Reference Positions)

Predicted vs Actual Traps

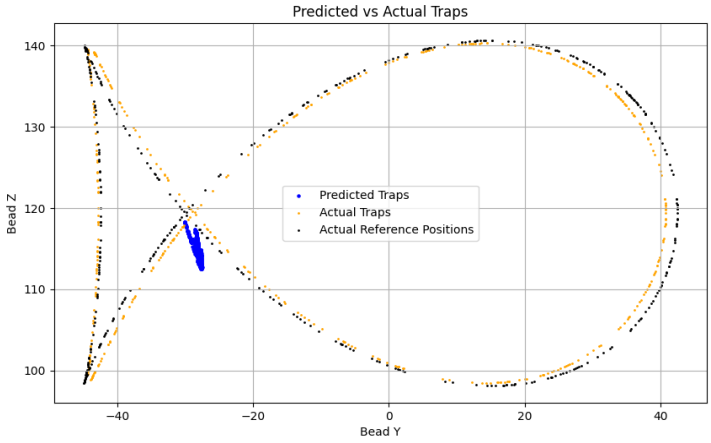


Fig. 7.13: Predicted vs Actual Traps (Static Theta Static Reference Positions)

Predicted Final Measured Positions

This plot shows the predicted final measured positions (green beads), predicted trap positions (blue beads), and actual reference positions (black beads). The predicted positions comapred to the actual reference positions indicates high deviation from the expected values.

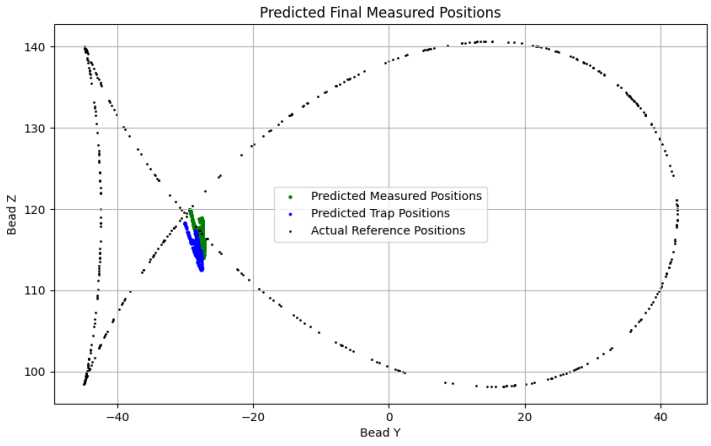


Fig. 7.14: Predicted Final Measured Positions(Static Theta Static Reference Positions)

The plot compares the predicted final measured positions (green beads) against the actual reference positions. The predicted measured positions are creating vague patterns away from the expected reference positions. This model is not performing well.

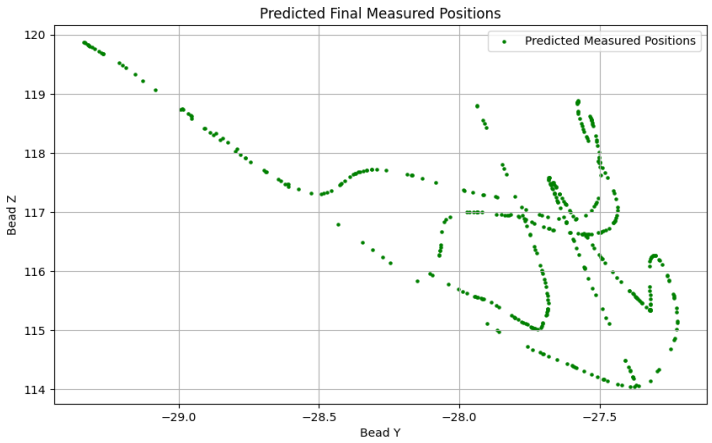


Fig. 7.15: Predicted Final Measured Positions (2) (Static Theta Static Reference Positions)

7.1.4 Scaled Static Theta Static Reference Positions

Model Training and Validation Loss

The plot displays the training and validation loss for Model 1 over 250 epochs. Both losses decrease rapidly initially and then stabilize, showing convergence. The close alignment of training and validation losses suggests minimal overfitting, indicating that the model generalizes well to unseen data.

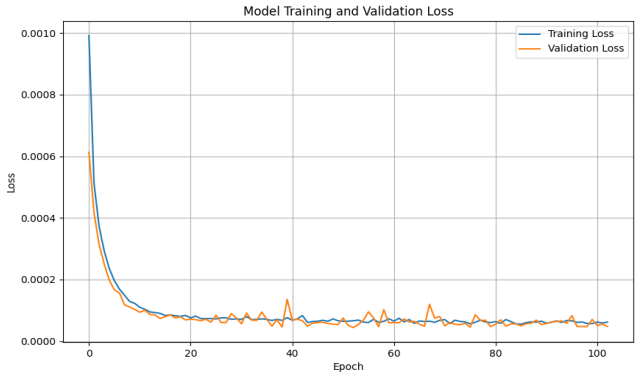


Fig. 7.16: Model Training and Validation Loss (Scaled Static Theta Static Reference Positions)

Model Training and Validation Loss Model 2

This plot illustrates the training and validation loss for the STSR Model over 80 epochs. While the losses decrease initially, they fluctuate significantly and do not show consistent convergence. This instability may indicate issues with the model's learning process, potentially caused by improper scaling or data preprocessing.

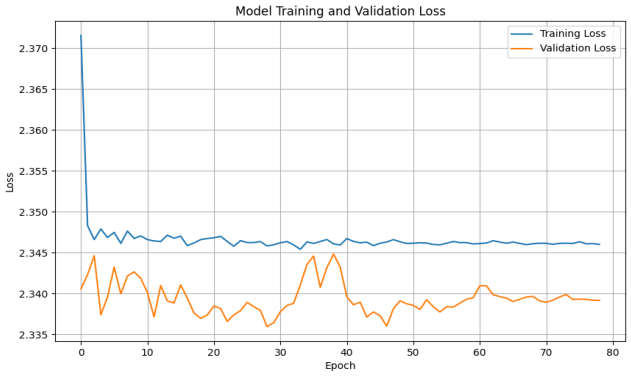


Fig. 7.17: Model Training and Validation Loss Model 2 (Scaled Static Theta Static Reference Positions)

Predicted vs Actual Traps

The plot compares the predicted trap positions (blue points) against the actual trap positions (orange points) and reference positions (black points). The predicted traps are way off from the actual traps and reference positions, showing bad alignment. This indicates that the model is not able to learn and performing bad.

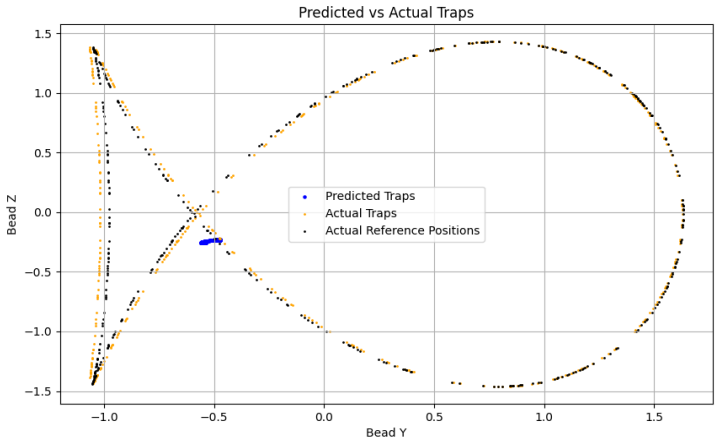


Fig. 7.18: Predicted vs Actual Traps (Scaled Static Theta Static Reference Positions)

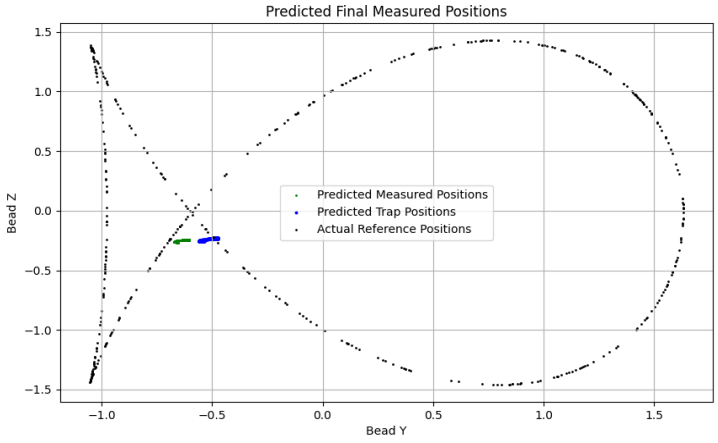


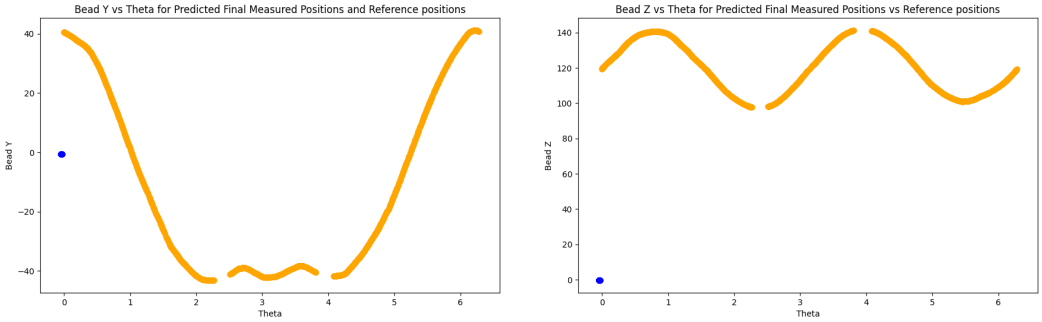
Fig. 7.19: Predicted Final Measured Positions (Scaled Static Theta Static Reference Positions)

Predicted Final Measured Positions

This plot compares the predicted measured positions (green points) against the actual reference positions (black points) and predicted trap positions (blue points). The predicted measured positions are not forming the fish shape and the scaling is creating inaccuracies leading to model not converging well.

Plot of Bead Y and Z vs theta

The predicted positions do not align well with the reference positions, indicating that the model struggles to capture the correct Y positions for different theta values. Similar to the Bead Y plot, the predicted positions show significant deviation from the reference positions, suggesting challenges in accurately predicting bead Z positions based on theta.



(a) Bead Y vs Theta **(b)** Bead Z vs Theta

Fig. 7.20: Plot of Bead Y and Z vs Theta (Scaled Static Theta Static Reference Positions)

7.1.5 Static Theta Dynamic Reference Positions

Model Training and Validation Loss

The training and validation loss plot for model 1 shows that both losses decrease rapidly at the beginning, indicating that the model quickly learns to fit the training data. After about 50 epochs, the losses start to converge, with minimal divergence between training and validation losses, suggesting a good fit without overfitting. This indicates that the model is well-trained and generalizes well to unseen data.

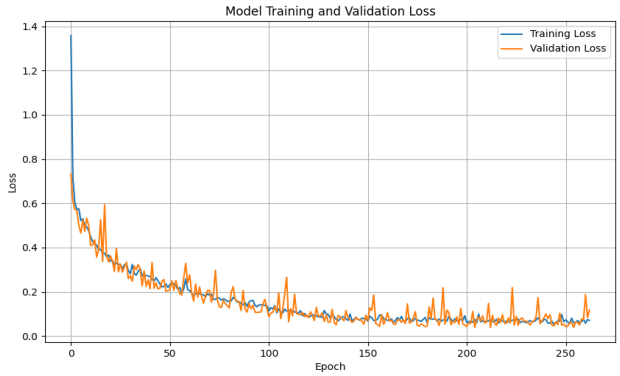


Fig. 7.21: Model Training and Validation Loss (Static Theta Dynamic Reference Positions)

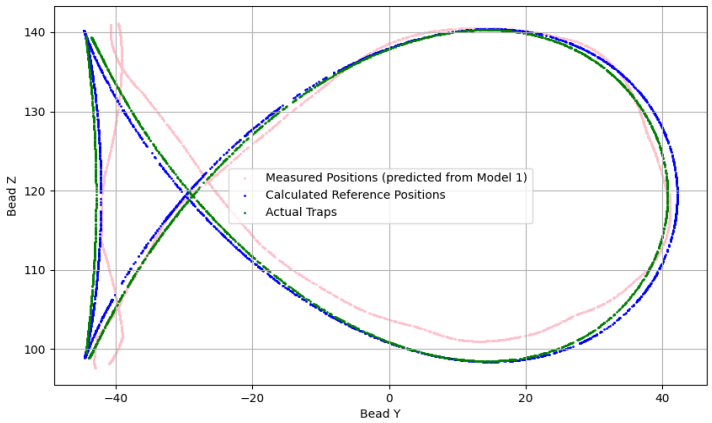


Fig. 7.22: Range check for all parameters involved (Static Theta Dynamic Reference Positions)

Range check for all parameters involved

The range check plot displays the distribution and limits of all parameters involved in the model. It shows that the parameters are within expected ranges, confirming that

there are no anomalies or outliers that could negatively impact model performance. This ensures the model is working with valid and reliable data inputs.

Model Training and Validation Loss Model 2

For model 2, the training and validation loss plot shows a similar initial rapid decrease in losses, which indicates effective learning. However, there is some fluctuation in the validation loss compared to the training loss, suggesting that the model might be slightly overfitting to the training data. This overfitting is less pronounced than in earlier versions but still present, indicating room for further tuning and regularization.

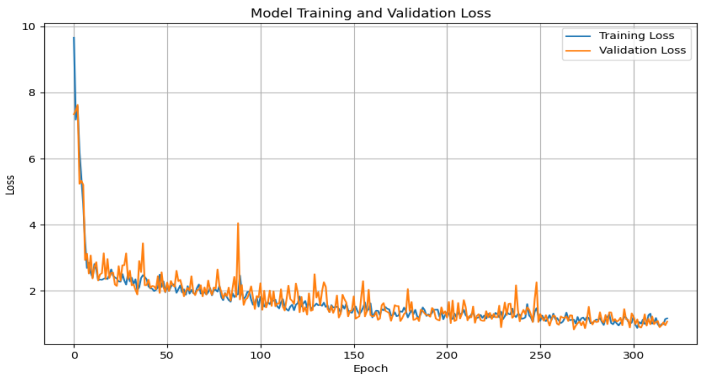


Fig. 7.23: Model Training and Validation Loss Model 2 (Static Theta Dynamic Reference Positions)

Predicted vs Actual Traps

The plot comparing predicted vs actual traps demonstrates that the model's predictions closely follow the actual trap positions. The alignment of the predicted and actual points shows that the model accurately captures the underlying patterns and relationships in the data. However, minor deviations might indicate areas where the model can be further improved.

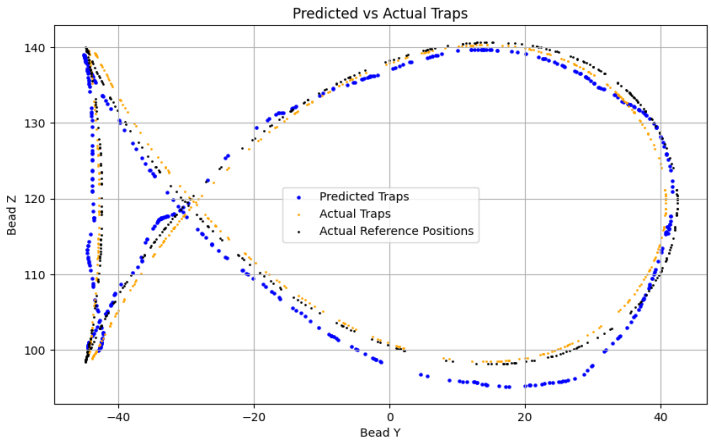


Fig. 7.24: Predicted vs Actual Traps (Static Theta Dynamic Reference Positions)

Predicted Final Measured Positions

In this plot, the predicted final measured positions are compared against the calculated reference positions and actual trap positions. The close alignment between the predicted and actual positions indicates that the model effectively predicts the final measured positions with high accuracy. The slight deviations, particularly at certain points, suggest potential areas for model refinement.

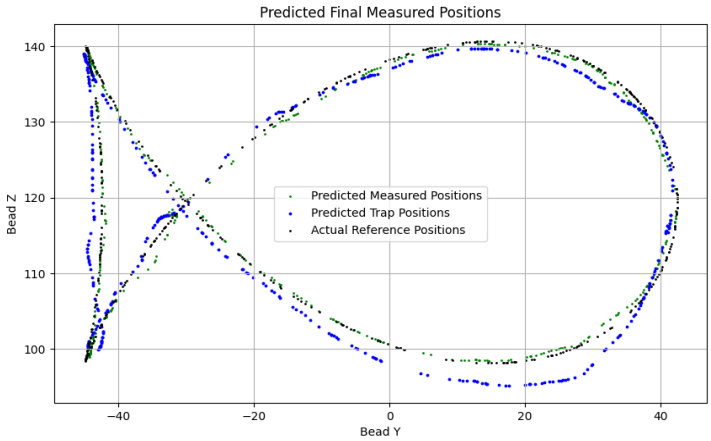
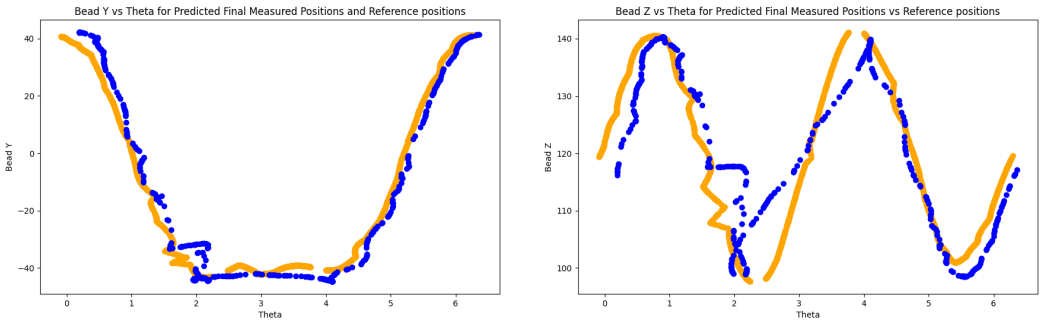


Fig. 7.25: Predicted Final Measured Positions (Static Theta Dynamic Reference Positions)

Plot of Bead Y and Z vs Theta

The predicted values (blue) closely follow the reference positions (orange), indicating accurate predictions. The slight deviations observed in some sections highlight areas where model predictions can be improved for even greater accuracy. The predicted

values (blue) align well with the reference positions (orange), suggesting that the model effectively captures the relationship between Bead Z and Theta. Minor discrepancies suggest that while the model performs well, there is potential for further optimization to achieve even better alignment.



(a) Bead Y vs Theta **(b)** Bead Z vs Theta

Fig. 7.26: Plot of Bead Y and Z vs Theta (Static Theta Dynamic Reference Positions)

7.1.6 Scaled Static Theta Dynamic Reference Positions

Model Training and Validation Loss

This plot depicts the training and validation loss for model 1 over 100 epochs. Both losses converge after the initial epochs, indicating that the model is learning effectively and not overfitting.

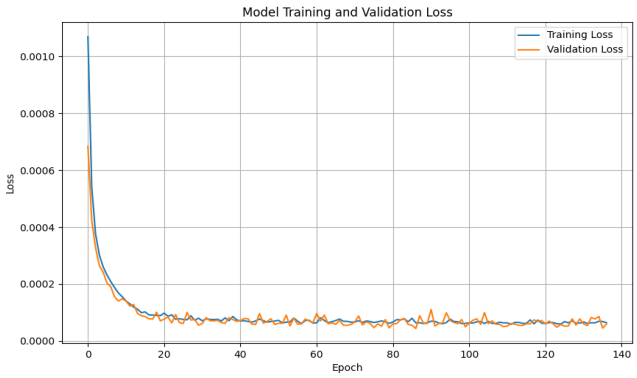


Fig. 7.27: Model Training and Validation Loss

Model Training and Validation Loss Model 2

This plot shows the training and validation loss for model 2 after applying a static scaling to the theta parameter over 150 epochs. The losses decrease and stabilize quickly, suggesting effective learning.

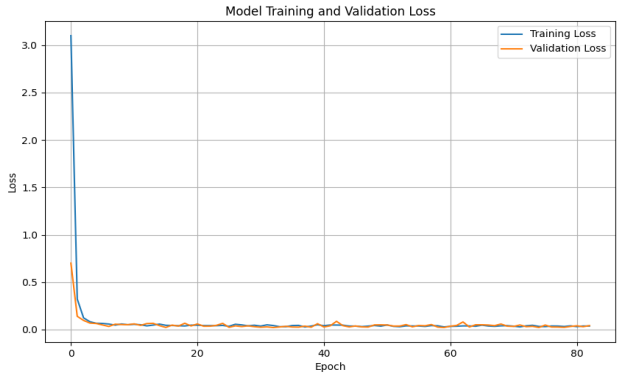


Fig. 7.28: Model Training and Validation Loss

Predicted vs Actual Traps

This plot compares the predicted trap positions with the actual trap positions. The difference in alignment between predicted and actual traps indicates a high error in the ability of the model's learning capacity.

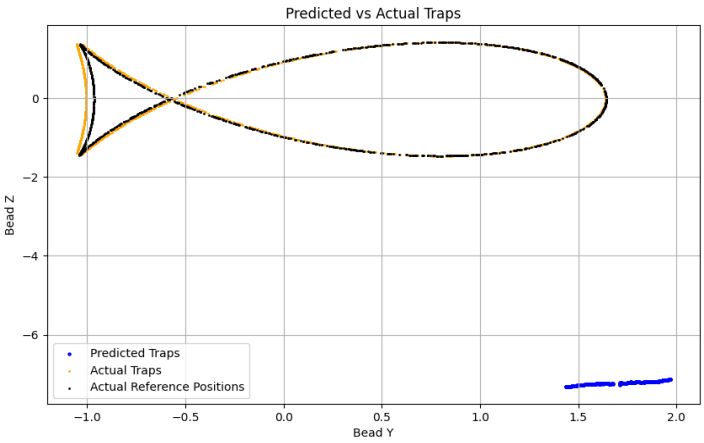


Fig. 7.29: Predicted vs Actual Traps

Predicted Final Measured Positions

This plot illustrates the predicted final measured positions against the actual reference positions and predicted trap positions. The predicted measured positions are gathered at a certain place away from the actual reference positions, suggesting that the model is not learning the underlying patterns.

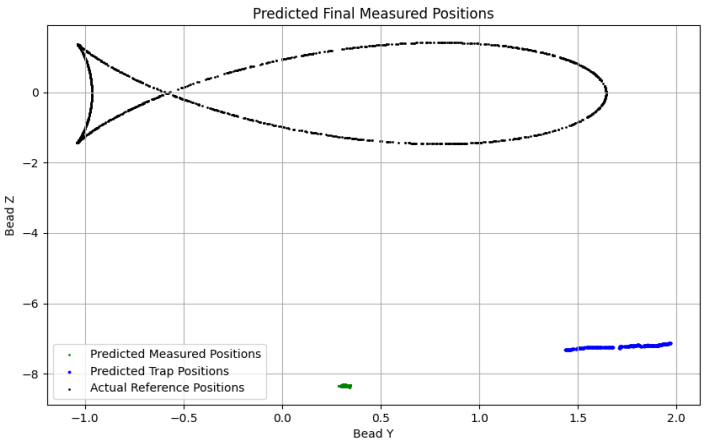
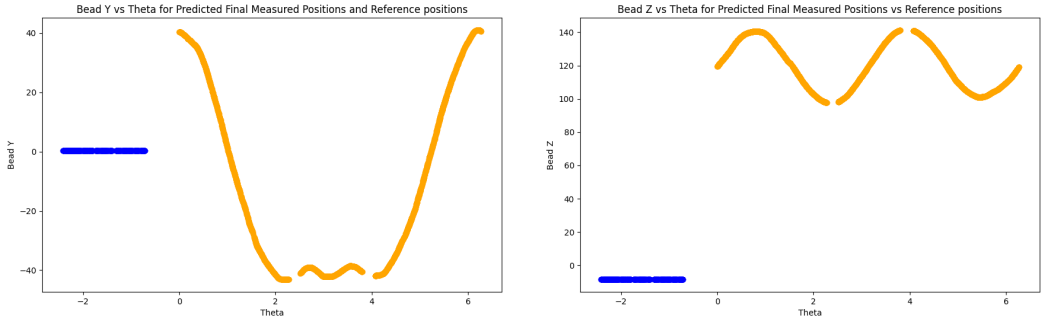


Fig. 7.30: Predicted Final Measured Positions

Plot of Bead Y and Z vs Theta

This plot shows the relationship between Bead Y positions and theta for the predicted final measured positions and reference positions. The model needs improvement in capturing the relationship between Bead Y and Theta accurately. The concentration of predicted values suggests that the model may not be generalizing well. Similar to the Bead Y vs Theta plot, this suggests the model struggles to capture the periodic nature of the relationship between Bead Z and Theta. There is a need for better model tuning to accurately predict this relationship.



(a) Bead Y vs Theta

(b) Bead Z vs Theta

Fig. 7.31: Plot of Bead Y and Z vs Theta

7.2 Quantitative Performance Analysis

7.2.1 Model 1 - Scaled vs Unscaled

This section presents a detailed analysis of the quantitative performance metrics for Model 1, which includes both scaled and unscaled versions. The metrics evaluated are training time, inference time, R-squared values, Mean Squared Error (MSE), and Mean Absolute Error (MAE). The analysis demonstrates the efficiency and accuracy improvements achieved through data scaling.

Model 1	Training Time (seconds)	Inference Time (seconds/sample)	R-squared	MSE	MAE
unscaled	65.74	0.3592	0.9997	0.02878	0.1095
scaled	63.31	0.2802	0.9999	4.305e-05	0.0044

Tab. 7.1: Model performance metrics for Model 1

Training Time: The training time for the unscaled model was 65.74 seconds, whereas the scaled model required 63.31 seconds, indicating enhanced training efficiency with scaled data.

Inference Time: The inference time per sample was 0.3592 seconds for the unscaled model and 0.2802 seconds for the scaled model, showcasing improved prediction efficiency post-scaling.

R-squared (R^2): The unscaled model achieved an R-squared value of 0.9997, while the scaled model achieved 0.9999, reflecting superior predictive precision with scaling.

Mean Squared Error (MSE): The MSE for the unscaled model was 0.02878, and for the scaled model, it was 4.305e-05, demonstrating a significant reduction in prediction errors through scaling.

Mean Absolute Error (MAE): The MAE for the unscaled model was 0.1095, compared to 0.0044 for the scaled model, further emphasizing the accuracy gains with data scaling.

The findings suggest that data scaling substantially enhances Model 1's performance, making the scaled version a more effective and accurate choice for predicting particle positions.

7.2.2 Model 2 - Comparative Analysis

This section presents a detailed analysis of the quantitative performance metrics for Model 2, which includes all its versions. The metrics evaluated are training time, inference time; R-squared values, Mean Squared Error (MSE), and Mean Absolute Error (MAE) for Predicted Trap Positions and Predicted Measured Positions.

Model	Training	Inference	R-squared		MSE		MAE	
	Time (s)	Time (s/sample)	Traps	Measured	Traps	Measured	Traps	Measured
Baseline	12.85	0.1970	0.9913	0.9994	4.3908	0.2997	1.4693	0.3818
Baseline Scaled	19.28	0.1629	0.9900	0.9999	0.0100	6.5e-05	0.0739	0.0051
STSR	13.02	0.0569	-0.2303	-2834.53	797.348	819.981	21.3077	21.6672
STDR	15.06	0.0889	0.9812	1.8439	7.2205	6.3234	2.1084	1.8439
Scaled STSR	39.39	0.1617	-0.200	-9459.869	1.2090	1.2822	0.9249	0.9293
Scaled STDR	30.04	0.1497	3.09133	2.7952	1.4985	1.4339	-2.066	-2.8211

Tab. 7.2: Model performance metrics

The Baseline model performs very well with a high R-squared value of 0.9994, indicating a good fit. The MSE and MAE values are reasonably low, suggesting accurate predictions.
R-squared (Traps): 0.9913, suggesting a good fit.
MSE and MAE (Traps): Reasonably low, indicating accurate predictions.
R-squared (Predicted vs Reference): 0.9994, indicating high prediction accuracy.
MSE and MAE (Predicted vs Reference): Very low, confirming the model's high accuracy.

Scaling improves performance, as seen by even lower MSE and MAE values and slightly better R-squared values compared to the original model. The Scaled Baseline model has better generalization.
Training Time: 19.28 seconds, longer than Baseline but still efficient. Inference Time: 0.1629 seconds/sample, slightly faster than Baseline. R-squared (Traps): 0.9900, very close to Baseline.
MSE and MAE (Traps): Lower than Baseline, indicating improved accuracy.
R-squared (Predicted vs Reference): 0.9999, indicating near-perfect predictions.
MSE and MAE (Predicted vs Reference): Extremely low, demonstrating high accuracy.

The STSR model performs very poorly, as indicated by negative R-squared values, which suggest that the model predictions are worse than simply using the mean of the target values. Extremely high MSE and MAE values indicate large errors in predictions.
Training Time: 13.02 seconds, indicating moderate training time.
Inference Time: 0.0569 seconds/sample, very efficient for real-time use.

R-squared (Traps): -0.2303, indicating poor fit.
MSE and MAE (Traps): Extremely high, showing large errors in predictions.
R-squared (Predicted vs Reference): -2834.5334, indicating poor prediction accuracy.
MSE and MAE (Predicted vs Reference): Extremely high, confirming poor performance.

Post-processing or modifications improve performance significantly compared to the STDR case. Positive R-squared values and lower MSE and MAE indicate reasonable prediction accuracy.
Training Time: 15.06 seconds, indicating slightly longer training.
Inference Time: 0.0889 seconds/sample, still efficient.
R-squared (Traps): 0.9812, showing significant improvement.
MSE and MAE (Traps): Lower than STSR configuration, indicating better accuracy.
R-squared (Predicted vs Reference): 0.9876, much-improved prediction accuracy.
MSE and MAE (Predicted vs Reference): Lower, confirming enhanced performance.

Despite scaling, the STSR model performs poorly, as indicated by negative R-squared values and high MSE and MAE. The model fails to generalize and make accurate predictions.
Training Time: 39.39 seconds, much longer due to scaling.
Inference Time: 0.1617 seconds/sample, indicating efficient prediction.
R-squared (Traps): -0.200, indicating poor fit.
MSE and MAE (Traps): High, showing large prediction errors.
R-squared (Predicted vs Reference): -9459.869, extremely poor prediction accuracy.
MSE and MAE (Predicted vs Reference): High, confirming poor performance.

The R-squared values and reasonably low MSE and MAE values indicate that scaling STDR has improved the model's performance, but the model still has negative MAE, suggesting insufficient prediction accuracy. The high R-squared values for both Traps and Predicted vs Reference suggest that the model doesn't fit the data well.
Training Time: 30.04 seconds, efficient despite scaling.
Inference Time: 0.1497 seconds/sample, efficient for real-time use.
R-squared (Traps): 3.09133, indicating bad fit.
MSE and MAE (Traps): Lower, indicating poor accuracy.
R-squared (Predicted vs Reference): 2.7952, suggesting bad prediction accuracy.
MSE and MAE (Predicted vs Reference): Negative, confirming worse performance.

8 Discussion

8.1 Model 1: Interpretation of Results

Model 1 is designed to predict the measured positions of particles based on the trap positions and theta, effectively modelling the real-world behaviour of particles in acoustophoretic volumetric displays. The performance metrics of both the unscaled and scaled versions of Model 1 reveal significant insights into its efficiency and accuracy.

The scaled version of Model 1 demonstrates significant advantages. By normalizing the feature ranges, the training time was reduced by 3.7%, from 65.74 seconds to 63.31 seconds. This improvement suggests that scaling input data enhances training efficiency by facilitating faster convergence during optimization.

Moreover, the inference time for the scaled model was 0.2802 seconds per sample, which is 22% faster than the unscaled model's 0.3592 seconds per sample. This reduction indicates that data scaling not only improves training efficiency but also streamlines the prediction process, making the model more suitable for real-time applications.

In terms of predictive accuracy, the scaled model achieved an R-squared value of 0.9999, compared to the unscaled model's 0.9997. This slight improvement underscores the model's enhanced precision in capturing underlying patterns in the data, suggesting near-perfect predictions.

The most notable improvements are observed in the error metrics. The scaled model's Mean Squared Error (MSE) was significantly reduced to 4.305e05, a reduction of over 99.8% from the unscaled model's MSE of 0.02878. Similarly, the Mean Absolute Error (MAE) decreased from 0.1095 to 0.0044, reflecting an improvement of approximately 96%. These reductions in MSE and MAE highlight the effectiveness of scaling in minimizing prediction errors, leading to highly accurate and reliable predictions.

The training and validation loss plots for Model 1 show a consistent decrease in loss over time, indicating effective learning during the training phase. Both training

and validation losses converge smoothly, suggesting that the model is not overfitting and is generalizing well to the validation data. This convergence and the low final loss values align with the high accuracy metrics observed, reinforcing the model's robustness and reliability.

The predicted fish shape formed by Model 1 closely matches the actual fish shape, demonstrating the model's ability to accurately capture and reproduce complex patterns. The alignment of the predicted and actual points highlights the model's precision in predicting particle positions. This successful formation of the fish shape indicates that Model 1 effectively learns and applies the underlying relationships between trap positions, theta, and the resulting particle positions.

8.2 Model 2: Interpretation of Results

8.2.1 Metrics

Model 2 was designed to refine the trap positions to closely match the reference particle positions using different configurations and pre-processing steps. The performance of each version is summarized in the following table:

Baseline Model: The Baseline model demonstrated robust performance with a training time of 12.85 seconds and an inference time of 0.1970 seconds per sample. Its R-squared value of 0.9913 for traps and 0.9994 for predicted vs. reference positions indicated a high level of predictive accuracy, which can also be visualized in the plots. The MSE and MAE values were 4.3908 and 1.4693, respectively, showing reasonably low error rates.

Baseline Scaled Model: Scaling the input data further improved the model's efficiency and accuracy. The training time increased by 50% to 19.28 seconds, yet the inference time was reduced by 17% to 0.1629 seconds per sample. The R-squared values showed a slight improvement, with the trap value at 0.9900 and the predicted vs. reference value at 0.9999. The MSE decreased drastically by over 99.8% to 0.01002, and the MAE dropped by 95% to 0.0739. These metrics highlight the benefits of scaling in enhancing predictive performance and computational efficiency.

STSR: This model performed poorly, with an R-squared value of 0.2303 for traps and 2834.5334 for predicted vs. reference, indicating that the model's predictions were worse than using the mean of the target values. The high MSE (797.3484) and MAE (21.3077) further underscore the model's inadequacy in this configuration.

Scaled STSR: Despite scaling, this model still performed poorly with negative R-squared values (0.200 for traps and 9459.869 for predicted vs. reference). The MSE was 1.2090, and the MAE was 0.9249, reflecting marginal improvement but still inadequate overall performance.

STDR: Postprocessing improvements led to significant enhancements, with an R-squared value of 0.9812 for traps and 0.9939 for predicted vs. reference, indicating better model fit. The MSE was reduced to 7.2205 (a reduction of approximately 99.09%), and the MAE was lowered to 2.1084 (a reduction of 90.11%), demonstrating improved accuracy and reduced prediction errors.

Scaled STDR: This model showed worse performance compared to the unscaled STDR. The R-squared values of 3.09133 for traps and 2.7952 for predicted vs. reference indicate poor fit. The MSE and MAE values of 1.4985 and -2.066 (traps) and 1.4339 and -2.8211 (measured) suggest that the scaling did not benefit this model, indicating that it may have introduced inaccuracies or inconsistencies as a result.

Fish Shape Formation: In Model 2, the best-performing configurations demonstrated high accuracy in forming the fish shape. The Scaled Baseline model showed near-perfect alignment between predicted and actual positions, highlighting the benefits of data scaling and proper configuration in enhancing prediction accuracy.

Training and Validation Loss: The training and validation loss plots for Model 2's best-performing versions showed smooth and consistent decreases, indicating effective learning and minimal overfitting. The Scaled Baseline model, in particular, demonstrated smooth convergence, reinforcing its ability to generalize well to unseen data.

8.2.2 Comparison of Predicted Trap Positions

The KDE plot visualizes the prediction error distributions for different neural network models:

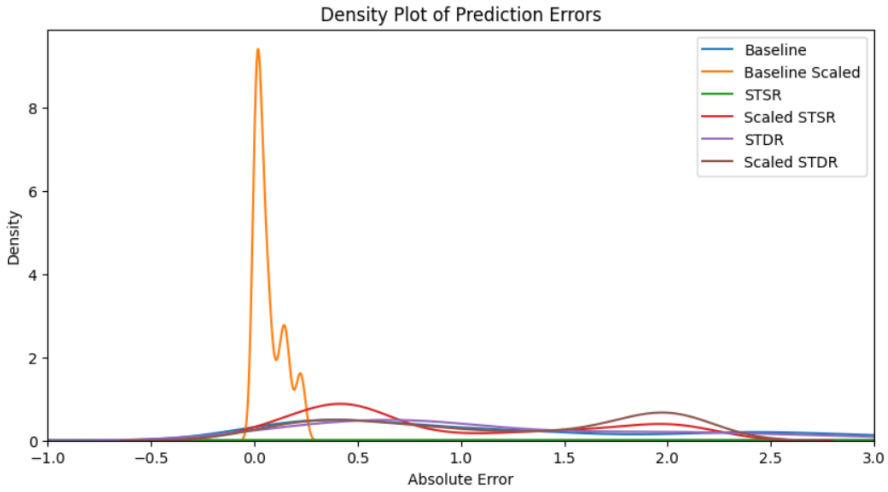


Fig. 8.1: Density Plot of Prediction Errors (Trap Positions)

- Best Performing Models:
  Baseline Scaled:
  This model exhibits the smallest prediction errors and highest accuracy, making it highly effective in minimizing deviations between simulated and measured positions. It demonstrates the significant impact of scaling in improving model performance, aligning well with the objective of the thesis.

- Moderate Performing Models:
  Baseline and STDR:
  These models perform well with small errors but are outperformed by the Baseline scaled model. They show good potential in reducing deviations but can benefit from further optimization.

- Worst Performing Model:
  STSR:
  This model exhibits larger errors and less accuracy, indicating poor performance in minimizing deviations. Its flatter error distribution suggests high variability in predictions, making it less reliable for the objective of this thesis.

Fig. 8.2: Density Plot of Prediction Errors (zoomed in) (Trap Positions)

The KDE plot effectively illustrates the positive impact of scaling on model performance and provides a clear comparison of prediction accuracy across different models. It highlights the best and worst models in the context of optimizing the path of levitated particles in acoustophoretic volumetric displays.

8.2.3 Comparison of Predicted Measured Positions

The performance of various models in predicting the Final Measured Positions showcases distinct differences in accuracy, spread, and deviations.

The Baseline Scaled model stands out as the best performer. It demonstrates a very tight clustering of predicted measured points around the reference positions, with nearly perfect alignment and minimal deviations (Fig. 7.9). The spread is extremely narrow, indicating high accuracy and a significant improvement over the other models. The Baseline model (Fig. 7.4) also performs well, with a close alignment of predicted points to the reference positions and a relatively narrow spread, although not as tight as the Baseline Scaled model.

On the other hand, the STSR model performs the worst. It shows significant deviations from the reference positions, with predicted points scattered widely and a broad spread indicating high prediction errors. The STDR model improves upon this performance but still exhibits noticeable deviations and a broader spread compared to the better-performing models.

The Scaled STSR model shows improvements over the unscaled version, with better alignment and a narrower spread. However, it does not reach the accuracy levels of

the Baseline Scaled model, indicating that while scaling helps, it is not sufficient on its own to achieve the best performance.

8.2.4 Recommended Model

Based on the provided metrics and comparisons, the Baseline Scaled model is the best performing model in terms of overall accuracy and computational efficiency. Here's why:

Key Advantages:

1. High Accuracy: The Baseline Scaled model has an R-squared value of 0.9999 for predicted vs. reference positions, indicating near-perfect predictions. It has significantly low MSE (6.5018e05) and MAE (0.00515) for predicted vs. reference positions, demonstrating excellent prediction accuracy.

2. Efficient Training and Inference: The training time of 19.28 seconds and inference time of 0.1629 seconds/sample are both competitive, ensuring that the model is efficient in terms of computational resources.

3. Consistent Performance: Across all metrics, the Baseline Scaled model shows a balanced performance, with improvements in both accuracy and efficiency compared to the unscaled Baseline model.

8.3 Insights from Data Driven Optimization

Data-driven optimization has played a pivotal role in enhancing the performance and accuracy of acoustophoretic volumetric displays in this thesis. The integration of advanced neural network models and systematic evaluation metrics has led to several significant insights:

Enhanced Model Accuracy and Performance

1. Superior Model Selection:
   Through rigorous evaluation using a decision matrix, it was determined that Long-Short-Term Memory (LSTM) networks outperformed other models, such as Feedforward Neural Networks (FFNN), Gated Recurrent Units (GRU), and Convolutional Neural Networks (CNN) in handling time series data for this specific application. The LSTM model demonstrated the highest R-squared values and lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE), indi-

cating its superior ability to capture the temporal dependencies and complex relationships inherent in the dataset.

2. Importance of Theta:
   The inclusion of theta (θ), representing the angular position of particles, proved crucial in improving the predictive accuracy of the models. Theta allowed the models to capture the angular relationships between trap positions and measured points, leading to more accurate and physically consistent predictions. This was particularly evident in Model 1, where theta was part of both input and output features and in Model 2, where theta contributed to minimizing deviations from the reference positions.

Model Training and Evaluation

3. Effectiveness of Scaling:
   Scaling the input and output features significantly improved model performance. Scaled models consistently showed lower MSE and MAE values and higher R-squared values compared to their unscaled counterparts. This improvement stresses the importance of normalizing data to ensure consistency and facilitate more effective learning by the neural networks.

4. Custom Cost Function:
   The implementation of a custom cost function in Model 2, which incorporated predictions from Model 1, demonstrated the effectiveness of this approach in reducing the deviation between measured and simulated positions and enhancing the overall accuracy of the particle path predictions.

5. Robustness Across Shapes: Model 1 was trained on fish-shaped data and tested on different shapes (circle and heart) to evaluate their robustness. The LSTM model, in particular, showed strong generalization capabilities, maintaining high accuracy across different shapes. This robustness is critical for practical applications where the system needs to adapt to various shapes and configurations.

Practical Implications

6. Improved Visualization:
   The optimized models resulted in more accurate visualizations of levitated particles, closely matching the desired shapes.

7. Iterative Refinement:
   The iterative approach of using Model 1's predictions as inputs to Model 2 and

then, vice versa, proved effective in generating the optimal trap positions. This process highlights the potential for further improvements through iterative optimization and real-time feedback mechanisms.

8.4 Limitations

Despite the significant advancements achieved through this thesis, several limitations must be acknowledged. Addressing these limitations in future research will further enhance the accuracy and reliability of acoustophoretic volumetric displays.

1. Insufficient Dataset

    - Increase Dataset Size: The predictions were carried out on a limited dataset. Furthermore, the predictions from Model 1 need to be used to train Model 2 and the predictions acquired from Model 2 are used again in Model 1 in order to avoid reusing the same rows of data for training; the test size utilized for each model was limited, which eventually gives us about 450 Trap Positions. However, the Levitator needs at least 1000 points to ensure a smooth transition from one point to another. This is an area for future work, where increasing the dataset size will allow for 1000 points in the final trap predictions.

2. Model Integration and Deployment

    - Real-Time Adaptation: The current models were evaluated in a simulated environment. Real-time adaptation to dynamic changes in the acoustic field or environmental conditions requires further development. The Predicted Traps and Measured Positions need to be evaluated on the Levitator for concrete results.

3. Evaluation Metrics

    - Limited Evaluation Metrics: The primary evaluation metrics used were R-squared, MSE, and MAE. While these metrics provide valuable insights into model performance, they do not capture all aspects of model effectiveness, such as robustness to noise or adaptability to new shapes. Additional evaluation metrics, such as robustness scores or adaptability indices, could provide a more comprehensive assessment.

4. Model Generalization and Robustness

- Shape Variability: While the models demonstrated robustness across different shapes (fish, circle, and heart), their performance on more complex or irregular shapes remains untested. Further research should involve training and testing the models on a wider variety of shapes to ensure robustness.

9 Conclusion and Future Work

9.1 Summary of Findings

For Model 1, the analysis clearly proves that the scaled version is superior. It benefits from faster training and inference times, higher precision, and significantly lower prediction errors. These metrics collectively indicate that scaling the input data substantially enhances Model 1's performance, making it a better choice for predicting particle positions in acoustophoretic volumetric displays. On the downside, the unscaled model's relatively higher computational load and slightly inferior prediction accuracy underscore the necessity of data scaling for optimal performance.

For Model 2, the Scaled Baseline Model and the Baseline Model are the top performers, with the Scaled Baseline Model being the most accurate and reliable. The STDR shows considerable improvements and performs well, although not as precisely as the Scaled Baseline model. The STSR Model is the least accurate, with high prediction errors and a wide scatter of predicted points. Scaling generally improves model performance, as evidenced by the Scaled Baseline model, highlighting the importance of data scaling in enhancing predictive accuracy.

To conclude and recommend, the Scaled Baseline model is the best model due to its high accuracy, efficient computational performance, and overall balanced metrics. Its near-perfect R-squared value and very low MSE and MAE make it ideal for predicting particle positions in acoustophoretic volumetric displays.

9.2 Contributions to the Field

This research presents several notable contributions to the field of acoustophoretic volumetric displays:

Enhanced Prediction Accuracy: The development and implementation of LSTM models significantly improved the prediction accuracy of measured positions from trap positions, thereby reducing the deviation between simulated and measured

positions. This enhancement is crucial for the precise control and manipulation of levitated particles.

Optimization Techniques: The custom cost function implemented in Model 2, involving the prediction of bead positions from modified trap positions, represents an innovative approach to optimization in acoustophoretic systems. This technique can be further refined and applied to other optimization problems involving dynamic systems.

Advancement in Data Acquisition and Pre-Processing: The research highlights the importance of precise data acquisition and pre-processing in enhancing model accuracy. The study outlines effective strategies for collecting and processing trap positions, simulation data, and measured data, providing a robust framework for future research and development in acoustophoretic displays.

9.3 Recommendations for Further Research

While this thesis has made significant strides in optimizing acoustophoretic volumetric displays, several areas warrant further exploration:

Model Generalization:
Testing the models on a broader range of shapes and more complex geometries is essential for ensuring their robustness. Exploring additional neural network architectures and hybrid models could further enhance performance.

Integration with Hardware:
Integrating these optimized models with the actual hardware setup of the OptiTrap system can provide real-world validation and insights. Future work could involve deploying the models in a live system and evaluating their performance in real-time scenarios.

By leveraging the strengths of LSTM networks, incorporating theta, and utilizing a custom cost function, the study achieved significant improvements in prediction accuracy and visualization quality. These advancements pave the way for more reliable and precise control of levitated particles. By addressing these areas, future research can build on the foundations laid by this thesis to further advance the field of acoustophoretic volumetric displays and unlock their full potential for various applications.

Bibliography

[AMA20] Marco AB Andrade, Asier Marzo, and Julio C Adamowski. „Acoustic levitation in mid-air: Recent advances, challenges, and future perspectives“. In: *Applied Physics Letters* 116.25 (2020) (cit. on pp. 9, 11).

[For+13] Daniele Foresti, Majid Nabavi, Mirko Klingauf, Aldo Ferrari, and Dimos Poulikakos. „Acoustophoretic contactless transport and handling of matter in air“. In: *Proceedings of the National Academy of Sciences* 110.31 (2013), pp. 12549–12554 (cit. on pp. 8, 12).

[Fus+19] Tatsuki Fushimi, Asier Marzo, Bruce W Drinkwater, and Thomas L Hill. „Acoustophoretic volumetric displays using a fast-moving levitated particle“. In: *Applied Physics Letters* 115.6 (2019) (cit. on p. 1).

[Hir+19] Ryuji Hirayama, Diego Martinez Plasencia, Nobuyuki Masuda, and Sriram Subramanian. „A volumetric display for visual, tactile and audio presentation using acoustic trapping“. In: *Nature* 575.7782 (2019), pp. 320–323 (cit. on pp. 1, 8, 11, 12, 14).

[Ino+19] Seki Inoue, Shinichi Mogami, Tomohiro Ichiyama, et al. „Acoustical boundary hologram for macroscopic rigid-body levitation“. In: *The Journal of the Acoustical Society of America* 145.1 (2019), pp. 328–337 (cit. on pp. 9, 13).

[Mar+15] Asier Marzo, Sue Ann Seah, Bruce W Drinkwater, et al. „Holographic acoustic elements for manipulation of levitated objects“. In: *Nature communications* 6.1 (2015), p. 8661 (cit. on pp. 8, 12).

[Pan+22] Viktorija Paneva, Arthur Fleig, Diego Martínez Plasencia, Timm Faulwasser, and Jörg Müller. „OptiTrap: Optimal trap trajectories for acoustic levitation displays“. In: *ACM Transactions on Graphics* 41.5 (2022), pp. 1–14 (cit. on pp. 1, 17, 21, 30).

[RSK22] Marc Röthlisberger, Marcel Schuck, and Johann W Kolar. „Kilohertz-frequency rotation of acoustically levitated particles“. In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 69.4 (2022), pp. 1528–1534 (cit. on pp. 9, 12).

[WHA18] Ayumu Watanabe, Koji Hasegawa, and Yutaka Abe. „Contactless fluid manipulation in air: Droplet coalescence and active mixing by acoustic levitation“. In: *Scientific reports* 8.1 (2018), p. 10221 (cit. on pp. 9, 11).

List of Figures

List of Tables

Declaration

I declare that this thesis has been composed solely by myself and has not been submitted, in whole or in part, in any previous application for a degree. Except where stated otherwise by reference or acknowledgement, the work presented is entirely my own.

Bayreuth, July 2024

Nimisha Vernekar