University of Nevada, Reno

ENHANCING QUADRUPED ROBOT DESIGN WITH INTELLIGENT PHYSICS-INFORMED NEURAL NETWORK-ASSISTED DYNAMIC STATE ESTIMATION AND ACTIVE SPINE INTEGRATION

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of Master of Science in

Electrical Engineering

by

Yuqing Liu

Dr. Hao Xu / Thesis Advisor

May 2024

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Yuqing Liu

entitled

Enhancing Quadruped Robot Design with Intelligent Physics-Informed Neural Network-Assisted Dynamic State Estimation and Active Spine Integration

be accepted in partial fulfillment of the
requirements for the degree of

Master of Science

Hao Xu, Ph.D.
Advisor

M. Sami Fadali, Ph.D.
Committee Member

Hung La, Ph.D.
Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean
Graduate School

May, 2024

ABSTRACT

This dissertation represents my master's work on quadrupedal robot systems at the Autonomous System Laboratory, University of Nevada, Reno. It primarily focuses on the intelligent design of quadruped robots and unmanned vehicles. The intelligence in these designs stems from advanced state estimation techniques and the integration of an active spine, combining constraints from physical models with insights from learning-based studies. An advanced Robot State Estimation (RSE) methodology is introduced, which uses a combination of a Physics-Informed Neural Network (PINN) and an Unscented Kalman Filter (UKF) with proprioceptive sensory data to enhance state estimation accuracy. This approach effectively calibrates the Inertial Measurement Unit (IMU), mitigates IMU drift through constraints applied via Ordinary Differential Equations, and eliminates the need for external contact sensors by identifying terrain interactions, improving odometry, and operational reliability in real-world scenarios.

Next, the focus is on enhancing the physical limitations of traditional robotics platforms. To achieve this, we utilize a dynamic spine to enhance flexibility and absorb impacts, which necessitates reevaluating the robot's dynamic model. To manage these complexities, the physical model estimation is enhanced, and Reservoir Computing is employed for real-time adaptive control. This significantly improves robotic mobility and stability in challenging environments.

ACKNOWLEDGEMENTS

First and foremost, I extend my deepest gratitude to my advisor, Prof. Dr. Hao Xu, for his unwavering support throughout my undergraduate and master's studies and research. His patience, motivation, enthusiasm, and knowledge have been fundamental to my academic journey. Dr. Xu has not only guided me through the complexities of my research but has also been instrumental in helping me navigate and overcome numerous challenges. His academic insights and steadfast support have been invaluable in shaping both my thesis work and my overall educational experience. I am profoundly thankful for his mentorship and his significant impact on my personal and professional growth.

I am immensely thankful to the members of my defense committee for their invaluable insights and contributions to my work. Their expertise was crucial in refining and enhancing the ideas presented in this thesis. I would also like to express my appreciation to my friends, lab mates, and university librarians for their unwavering support and inspiration throughout the various stages of my research. Their collective assistance and camaraderie have been instrumental in my academic journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

This chapter introduces an advanced Robot State Estimation (RSE) methodology specifically tailored for legged robots. It features a learning-based contact estimation framework that eliminates the dependency on external physical contact sensors. Our approach substantially enhances state estimation accuracy by integrating multimodal proprioceptive sensory data with a Physics-Informed Neural Network (PINN) and an Unscented Kalman Filter (UKF). The primary aim is to effectively calibrate the Inertial Measurement Unit (IMU) and provide a detailed depiction of the robot's dynamic state.

The use of PINN is crucial for mitigating IMU drift. Ordinary differential equations (ODEs) apply constraints on the loss function. This method offers significant advantages over traditional vision-based systems by remaining unaffected by visual impairments and removing the need for hard-to-integrate dedicated contact sensors.

Despite the advances achieved with the PINN-UKF integration, the physical limitations of traditional robotic platforms pose significant challenges. To address these, this thesis introduces a pioneering strategy by incorporating a dynamic spine into a quadruped robot, which enhances flexibility and shock absorption. This modification necessitates a complete reformulation of the robot's dynamic model, as the spine significantly alters system dynamics during movement, rendering traditional modeling techniques inadequate.

Two strategic methodologies are proposed to adapt to these complex dynamics: 1) enhancing physical model estimation for operation in challenging environments, particularly focusing on the unique dynamics introduced by dynamic spines in quadruped models. This

strategy aims to optimize the robot's interaction with complex terrains and 2) employ Reservoir Computing to manage the chaotic dynamics that arise from the spine's interactions during movement. This approach provides a robust framework for real-time adaptive control, effectively addressing the modeling challenges.

1.2 Related Work

The section summarizes the three categories of state estimation, namely; model-based, data-driven and hybrid, combines the pros of the model- based and data-driven approaches. It improves the model- based accuracy. The model-based approach relies on the laws of physics. The robot's behavior can be described using the geometric constraints, i.e., the kinematic model, or considering the forces and moments acting on the robot, i.e., the dynamic model. The kinematic model requires only geometrical parameters and does not need extensive robot parametrization because its reliability depends mainly on sensing capabilities. The state-of-the-art kinematic observer[1] is based on a linear parameter-varying system, where the states are the object velocities, and the accelerations are the inputs. This approach leads to high accuracy in transient maneuvers, but the model is not observable in nearly steady-state conditions[2]. Hence, a heuristic function is applied to avoid unobservability and to lead the lateral velocity to zero when the object moves straight or nearly straight[1]. The downside is the amount of data necessary to define the heuristic function. Moreover, despite the performance improvement, it is still susceptible to integration errors and sensor drift. Thus, in recent publications[3][4][5][6], the measurements from the Inertial Measurement Unit (IMU) are coupled with those from a Global Navigation Satellite System (GNSS) to increase the amount of information available for the estimator. The velocities measured by the GNSS are integrated into an estimation-prediction framework, which estimates the sideslip angle and partially compensates for the error induced by the

low GNSS sampling rate. However, the GNSS/IMU fusion kinematic approach still suffers from low GNSS sampling rate[3]. Furthermore, a high-precision GNSS is too expensive as the standard sensor of robots, and signal reception cannot always be assured. Therefore, it is mainly applied to racing[7]. Thus,a solution is to consider dynamic models that rely less on the sensor signal quality. Dynamic models allow a more robust noise computation of the accelerations than kinematic models[2]. However, dynamic models require estimation of model parameters, which is a critical source of uncertainty[8]. EKF and UKF are state-of-the-art estimation techniques for the model-based approach, and the process and the observation noises are commonly assumed to be Gaussian and uncorrelated. The EKF uses a first-order Taylor series expansion to linearize around the current mean and covariance. It has excellent accuracy in nearly steady-state conditions, and when the robot behaves closely to linearity and when the robot behaves with strong nonlinearities, UKF assures a better estimation accuracy because it linearizes up to the second order of the Taylor series expansion[9]. However, both observers suffer from the mismatches between the physical and modeled object behavior. A possible solution is to combine the pros of dynamic and kinematic models to develop a hybrid kinematic-dynamic observer[10][11]. This family combines the accuracy in transient maneuvers of the kinematic models and the better robustness to sensor noise of the dynamic models. The kinematic and the dynamic filters work simultaneously, and the final side slip angle estimation is a weighted average of the two approaches. The weights are chosen according to the lateral acceleration signal. However, tuning the weighting coefficients' tuning is difficult, and the optimum solution varies according to the considered maneuvers. Another solution to combine dynamic and kinematic models is the development of a modular scheme to estimate forces and longitudinal and lateral velocities in sequential steps[12]. The approach is experimentally validated in different road conditions, but the results do not show its performance when the robot is driven at the limit of handling. Thus, the approach's applicability to evasive maneuvers is

limited.

1.3 Thesis contributions and Outline

Motivated by discussion above, the thesis contributes to the literature by the following:

- Leveraging a Physics-Informed Neural Network (PINN) to address IMU drift issues through the integration of constraints within the loss function using Ordinary Differential Equations (ODEs).
- Enhancing state estimation by integrating a Physics-Informed Neural Network (PINN) with an Unscented Kalman Filter (UKF).
- Developed a contact estimator through learning-based training that is designed to accurately identify discontinuous events across diverse and complex environments.
- Replacing the traditional rigid body with a dynamic spine implementation, which improves flexibility and shock absorption, critical for maneuvering through challenging terrains.
- Introducing reservoir computing to provide a robust solution for managing the chaotic dynamics introduced by the complexity of the spine dynamic model.
- Implementing Lyapunov control algorithm to ensure system stability with spine implementation.

The rest of this thesis is organized as follows: Chapter 1 serves as an introduction, providing a brief overview of the Robot State Estimation (RSE) methodology specifically tailored for legged robots. It introduces the basic concepts of Inertial Measurement Unit

(IMU) drift and the solutions available to address this issue. The chapter also discusses the state estimation and related research, setting the stage for the motivation behind this study.

Chapter 2 addresses the Robot State Estimation (RSE) methodology, which employs a Physics-Informed Neural Network (PINN) in conjunction with an Unscented Kalman Filter (UKF). This combination enhances state estimation and effectively mitigates Inertial Measurement Unit (IMU) drift, resulting in a robust state estimation process.

Chapter 3 addresses the limitations of the previously discussed PINN-UKF methodology, particularly the physical constraints and rigidity encountered when navigating complex terrain. It offers an innovative modification by replacing the traditional rigid body with a dynamic spine implementation. The use of reservoir computing provides a robust solution for handling the complex dynamics induced by the spine model. Additionally, the chapter explores a causal Lyapunov control algorithm to ensure stability.

Chapter 4 summarized the research work and achievements of the entire dissertation and outlines directions for future research.

CHAPTER 2

ENHANCED ROBOT STATE ESTIMATION USING PHYSICS-INFORMED NEURAL NETWORKS AND MULTIMODAL PROPRIOCEPTIVE DATA

2.1 Abstract

In this chapter, we introduce an innovative Robot State Estimation (RSE) methodology incorporating a learning-based contact estimation framework for legged robots, which obviates the need for external physical contact sensors. This approach integrates multimodal proprioceptive sensory data, employing a Physics-Informed Neural Network (PINN) in conjunction with an Unscented Kalman Filter (UKF) to enhance the state estimation process. The primary objective of this RSE technique is to calibrate the Inertial Measurement Unit (IMU) effectively and furnish a detailed representation of the robot's dynamic state.

Our methodology exploits the PINN to mitigate IMU drift issues by imposing constraints on the loss function via Ordinary Differential Equations (ODEs). The advantages of utilizing a contact estimator based on proprioceptive sensory data are multifold. Unlike vision-based state estimators, our proprioceptive approach is immune to visual impairments such as obscured or ambiguous environments. Moreover, it circumvents the necessity for dedicated contact sensors—components not universally present on robotic platforms and challenging to integrate without substantial hardware modifications.

The contact estimator within our network is trained to discern contact events across various terrains, thereby facilitating resilient proprioceptive odometry. This enables the contact-aided invariant Kalman Filter to produce precise odometric trajectories. Subsequently, the UKF algorithm estimates the robot's three-dimensional attitude, velocity, and position.

Experimental validation of our proposed PINN-based method illustrates its capacity to assimilate data from multiple sensors, effectively reducing the influence of sensor biases by enforcing ODE constraints, all while preserving intrinsic sensor characteristics. When juxtaposed with the employment of the UKF algorithm in isolation, our integrated RSE model demonstrates superior performance in state estimation. This enhanced capability automatically reduces sensor drift impacts during operational deployment, rendering our proposed solution applicable to real-world scenarios.

2.1.1 Introduction

The field of robotics is continually advancing towards more autonomous and robust systems[13], driven by rapid enhancements in sensor technology and intelligent algorithms[14]. Among the sensors employed, the Inertial Measurement Unit (IMU) is fundamental for providing vital motion data. However, a significant challenge that persists is the IMU drift — an inherent error accumulation over time that can severely skew the state estimation of robots, leading to degraded operational accuracy and potential system malfunctions.

To address this critical issue, exploring innovative methodologies that counteract the effects of IMU drift and enhance overall robot state estimation is imperative. This research introduces a dual approach that integrates a Physics-Informed Neural Network (PINN) with an Unscented Kalman Filter (UKF) to forge a novel Robot State Estimation (RSE) methodology. Furthermore, the utilization of proprioceptive sensory data via a learning-based contact estimator constitutes a significant advancement in bolstering the robustness of state estimation amidst diverse environmental interactions. Conventional estimators rely heavily on external contact sensors, which may be rendered ineffective in multifaceted environments. By harnessing on proprioceptive data, the proposed RSE framework diminishes

reliance on such external factors, offering a robust alternative that is particularly beneficial in environments where external contact sensing is impractical.

2.1.2 Method Rationale

The targeted strategy of the Physics-Informed Neural Network (PINN) focuses on mitigating IMU drift by minimizing θ^*, which is determined based on physics-based constraints and the residual of the measurement data. By integrating physical laws directly into the learning process, PINNs effectively control drift, ensuring that state estimation is more accurate and reflective of the robot's true dynamical behavior.

In parallel, our approach incorporates a sophisticated contact estimation mechanism using Convolutional Neural Networks (CNNs). The architecture of the contact estimation network consists of two sets of convolutional operations followed by three fully connected layers. Each set features two one-dimensional convolution layers and a one-dimensional max pooling layer. These convolution layers are engineered to extract detailed features from proprioceptive sensory data, while the one-dimensional kernel, which operates across the time domain, optimizes computational efficiency by minimizing memory usage and processing time. Padding is applied to preserve data dimensions throughout the convolutions. The rectified linear unit (ReLU) serves as the activation function, enhancing nonlinear processing capabilities. Additionally, a dropout layer is incorporated in the second convolution set to prevent overfitting. Max pooling subsequently reduces the data dimensionality, further refining the feature extraction process.

The synergy between the PINN and the Unscented Kalman Filter (UKF) enhances the robustness of the estimation process. This integration improves the precision of state estimations and adeptly manages the non-linearities and uncertainties characteristic of dynamic

robotic environments. Unlike traditional methods that may require frequent recalibration, this combined approach dynamically adjusts to sensor errors and operational variances, providing a consistent, reliable framework. Experimental validations of the PINN-UKF model have demonstrated its superior ability to diminish the effects of sensor biases and maintain accurate trajectory estimations. Such capabilities are pivotal for enhancing the operational integrity of robots, particularly in complex real-world applications where precision and reliability are paramount.

2.2 Preliminaries

2.2.1 The Robot Model

The PINN UKF algorithm use a six-degrees-of-freedom(6-DoF) kinematic robot model to obtain simplification state estimates. As shown in Figure 2.1, the model includes the navigation and robot body coordinates. The starting point of the navigation coordinates is defined as the track start. The navigation coordinates consisted of three variables: E(east), N(north), and U(upward).

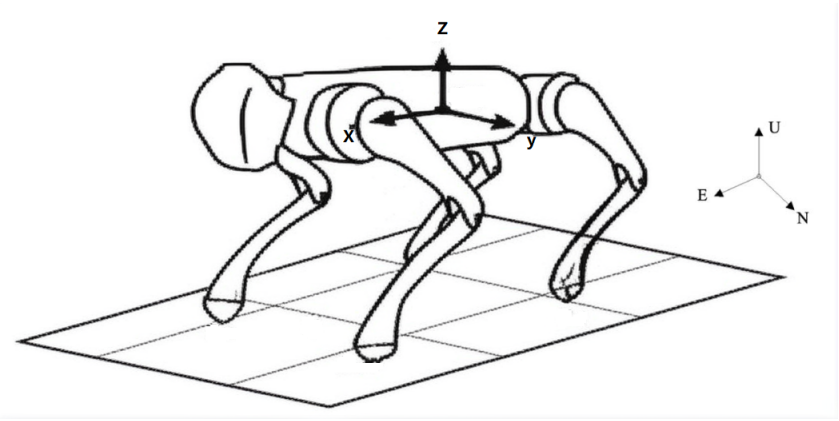

Figure 2.1: The 6-DoF robot model coordinates

The quadrupedal robot body's coordinate origin point was located at its center of mass, and the right-hand rule was used. The x-, y-, and z-directions are pointed forward, left, and upward, respectively. Acceleration and velocity could be broken down into longitudinal acceleration a_x /velocity v_x, lateral acceleration a_y/ velocity v_y, and vertical acceleration a_z/ velocity v_z. The robot's direction angle is defined as the rolling angle (around x, roll rate ω_x), pitching angle (around y, pitch rate ω_y), and heading angle (around z, yaw rate ω_z).

2.2.2 The Sensor States

A quadrupedal robot was equipped with an IMU sensor capable of measuring roll, pitch, and yaw angles and rates and tracking longitudinal, lateral, and vertical velocities and accelerations. The robot's proprioceptive state estimators typically combine IMU data with leg odometry for enhanced accuracy. Leg odometry relies on kinematic and contact data for state updates, making precise measurements of these variables essential. However, it's noteworthy that not all legged robots have specialized contact sensors or springs for contact detection. Integrating such dedicated contact sensors can be complex, often necessitating significant changes to the robot's hardware design.

In our research, we implemented a deep learning-based contact estimator that forgoes the need for specialized sensors. Instead, the system utilizes joint encoders, kinematics, and IMU data. To gather contact datasets, we employed the Unitree A1 robot. This approach involved the creation of a 'virtual sensor' that collects data from joint encoders, kinematics, and the IMU. The contact estimator then processes this data to simulate virtual leg torque and force outcomes. This innovative method replaces traditional force and torque sensors, enhancing the robot's functionality without the complexity of additional

hardware, demonstrating the versatility and effectiveness of the virtual sensor approach in varying environments.

Using the universal transverse mercator (UTM) and the navigation starting point, easting and northing are transformed from the GNSS coordinates into the navigation coordinates. Let the symbols E and N represented easting and nothing in the navigation coordinates, respectively. The UTM velocity is calculated from navigation coordinates as:

$$v_{\text{gps}} = \frac{\sqrt{(\Delta E)^2 + (\Delta N)^2 + (\Delta U)^2}}{\Delta t} \tag{2.1}$$

where Δt is a single GNSS interval, and ΔE, ΔN, and ΔU are the differences in Easting, Northing, and Altitude coordinates between two consecutive measurements.

The robot had multiple sensors, including the GNSS, IMU, and virtual sensor based on contact estimator. The research platform structure is present in chapter II. Due to real-time computation requirement, the monocular vision sensor are not used in the PINN-UKF algorithm. The sensor input of PINN-UKF are introduced in Table 2.1.

Table 2.1: The sensor inputs of PINN UKFM.

Sensor Types	Signal Name	Symbol	Units
GNSS	Easting	E	m
GNSS	Northing	N	m
GNSS	Altitude	U	m
GNSS	UTM velocity	V_{GPS}	m/s
IMU	Roll angle	ϕ	rad
IMU	Pitch angle	Θ	rad
IMU	Yaw angle	ψ	rad
IMU	Roll rate	ω_x	rad/s
IMU	Pitch rate	ω_y	rad/s
IMU	Yaw rate	ω_z	rad/s
IMU	Longitudinal velocity	v_x	m/s
IMU	Lateral velocity	v_y	m/s
IMU	Vertical velocity	v_z	m/s
IMU	Longitudinal acceleration	a_x	m/s^2
IMU	Lateral acceleration	a_y	m/s^2
IMU	Vertical acceleration	a_z	m/s^2
Virtual Sensor	Leg torque	M_x, M_y, M_z	kN·m
Virtual Sensor	Leg force	F_x, F_y, F_z	kN

2.2.3 Problem Formulation

The PINN-based is defined as a time-series forecasting model in which sensor signals are taken as discrete variables. Assuming that " n " represents the current timestamp, the PINN module input states are defined as:

$$\begin{aligned} X &= [X_{n-N}, X_{n-N+1}, \ldots, X_n] \\ X_n &= \left[X_n^{(1)}, X_n^{(2)}, \ldots, X_n^{(\vartheta)}\right] \end{aligned} \tag{2.2}$$

where $n-N$ is the starting time step, X represents the sensor signals shown in Table 1 , and ϑ is the number of sensor signals.

PINN is used as a universal function approximator to achieve IMU calibration. Build-

ing upon previous artificial intelligence-based techniques and the integration of the Kalman filter for estimation, the calibrated values are called "proxy-states." By applying the conservation principles derived from the robot dynamics, the proxy-states satisfy the conservation principles originating from the robot dynamics. Therefore, the PINN module output states is defined as:

$$\hat{u}_\theta = \left[\hat{u}_\theta^{(1)}, \hat{u}_\theta^{(2)}, \ldots, \hat{u}_\theta^{(\kappa)}\right] \tag{2.3}$$

where $\hat{u}_\theta$ represents the proxy-states, and κ is the number of proxy-states. ased on the Linear Time-Invariant (LTI) state-space assumption, these proxy-states are used to compute the integrated states. The integrated states represent the combined or processed information derived from the proxy-states over a certain time period. The sensor measurements of these integrated states z are represented as:

$$\begin{aligned} z &= [z_{n+1}, z_{n+2}, \ldots, z_{n+N}] \\ z_{n+1} &= \left[z_{n+1}^{(1)}, z_{n+1}^{(2)}, \ldots, z_{n+1}^{(\kappa)}\right] \end{aligned} \tag{2.4}$$

where $[n+1, n+2, \ldots, n+N]$ represent the output timestamps, and $n+N$ is the ending time step.

For better integration with the robot control, we hypothesized that the robot's physical model satisfies the linear time-invariant (LTI) state-space model over the interval $[n+1, n+2, \ldots, n+N]$:

$$\begin{cases} \dot{x} = Ax_t + \mathscr{B}_t \\ y_t = Cx_t + \mathscr{D}_t \end{cases} \tag{2.5}$$

where x_{t+1} is the state variable at next time step, x_t is the state variable at current time step, y_t represents the output variable at current time step, $\mathscr{B}_t$ and $\mathscr{D}_t$ are disturbances or noise, and $\dot{x}$ denotes the rate of change of the state variable.

By utilizing the LTI state-space model, the output of the PINN module could be used to compute the other robot states. Based on the ODE constraints, PINN ensured the estimated IMU states satisfied the physical relationships among sensor measurements.

Next, the proxy-states were inputted into the PINN UKF-based sensor-fusion model. Using these proxy-states, the proposed UKF module could estimate the robot's velocity and position.

2.3 Methodology

2.3.1 Structure of the Proposed RSE

In recent years, there has been a growing focus on the advancement of multi-sensor systems for robot state estimation, driven by their potential to enhance accuracy and robustness in intricate environments. The proposed PINN-UKF is an example of such a system, integrating multiple sensors to facilitate real-time estimation of robot dynamics. The architecture of the proposed PINN-UKF comprises three modules: the sensors module, the PINN module, and the UKF module, as shown in Figure 2.2. The sensors module within PINN-UKF incorporates various sensors, including GNSS, IMU, and Virtual Sensor, by combining the Kinematics Model and IMU sensor fusion, which provide comprehensive information about the robot's motion, encompassing position, velocity, acceleration, and orientation, as shown in Figure 2.3. The PINN module was utilized to adaptively reduce data noise to learn the intricate, nonlinear relationship between the sensor signals and the filtered robot states. Specifically, the PINN module employed the time-series sensor signals as input and generated the corresponding proxy-states. These proxy-states were subsequently inputted into the UKF module, which employed a state-space model to estimate

the robot's position, velocity, and other parameters. Through the synergistic utilization of both PINN and UKF, the proposed PINN-UKF accomplished precise and robust robot state estimation, as shown in Figure 2.2.

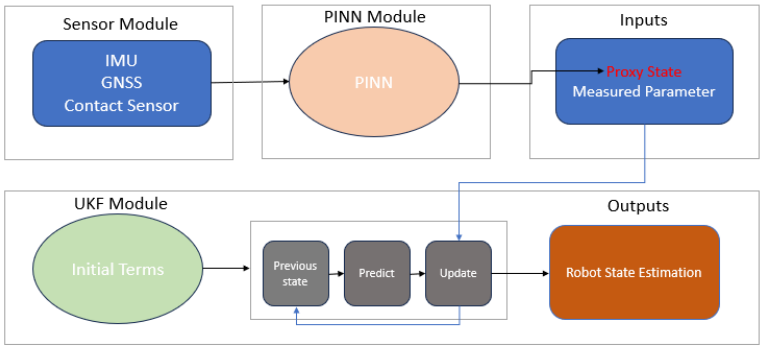

Figure 2.2: The structure of the proposed RSE

2.3.2 The PINN Module

In practical scenarios, diverse sensors may produce data with noise and drift due to their distinct characteristics and operational environments. To achieve accurate robot state estimation, we employed Physics-Informed Neural Network (PINN) to integrate the robot dynamics into a neural network architecture. The PINN module penalized the loss function with ordinary differential equations (ODEs) and algebraic equations to align the sensor data with the robot dynamics. Furthermore, the PINN module amalgamated data from multiple sensors to mitigate measurement errors and enhance the consistency of the data with actual robot dynamics. Consequently, the PINN module derived the angle and velocity by integrating the acceleration and angular velocity. Temporal interaction is extensively employed in the establishment of data-driven robot models, as it can capture complex temporal and hidden correlations for improved state prediction. Thus, a temporal model comprising an

encoder layer, a temporal interaction layer, and a decoder layer was proposed, as shown in Figure 2.3.

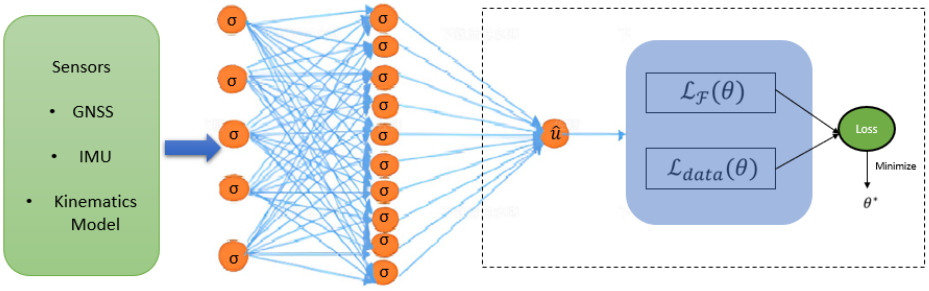

Figure 2.3: The proposed PINN module.

The encoder layer embeds and encodes the sensor signal X. Data are mapped into the high dimension through the multilayer perceptron(MLP). The encoder layer is defined as:

$$e_t = \sigma\left(W_{\text{enc}} X(t) + b_{\text{enc}}\right) \tag{2.6}$$

where e_t denotes the embedded feature vector. Additionally, the MLP includes a weighting matrix W_{enc}, bias term b_{enc}, and the rectified linear unit function (ReLU) activation function σ.

Since robot states have temporal interactions, the temporal interaction layer represents the temporal interactions of different hidden states. The time dimension of e_t is connected:

$$e^0 = \text{Concat}\left(e_t\right), \quad t \in [n-L, n] \tag{2.7}$$

where the embedded feature vectors are concatenated to form e^0. Then, the PINN module learns the temporal interaction:

$$e^l = \sigma\left(W^l e^{l-1} + b^l\right), \quad l = 1, 2, \ldots, s \tag{2.8}$$

where e^{l-1} and e^{l} are the input and output of layer l, respectively. The decoder layer predicted the proxy-states as follows:

$$\hat{u}_\theta = (W^s e^s + b^s) + u_p \tag{2.9}$$

where u_p represents the past measurement of the IMU, and $\hat{u}_\theta$ refers to the proxy states that represent the IMU calibration values. We defined the residual between the proxy states and past measurements of the IMU as the drift of the IMU. This structure was similar to the residual block in a residual network. The term $(W^s e^s + b^s) = \hat{u}_\theta - u_{\text{past}}$ represents the latent (hidden) solution of the drift of IMU. The PINN determines the parameter θ of the NN by minimizing the loss function:

$$\theta = \text{argmin}\, \mathscr{L}(\theta) \tag{2.10}$$

$$\mathscr{L}(\theta) = \mathscr{L}_{\mathscr{F}}(\theta) + \mathscr{L}_{\text{data}}(\theta) \tag{2.11}$$

$$\mathscr{L}_{\text{data}}(\theta) = \frac{1}{N_d} \sum_{i=1}^{N_d} |\hat{u}_\theta(X;\theta) - u(X)|^2 \tag{2.12}$$

$$\mathscr{L}_{\mathscr{F}_1}(\theta) = \sum_{k=1}^{7} \omega_k \left[\frac{1}{FN_{\mathscr{F}}} \sum_{t=n+1}^{n+N} \sum_{i=1}^{N_{\mathscr{F}}} |g_k\left(\hat{u}_0(X;\theta), X_n, z_t\right)|^2 \right] \tag{2.13}$$

$$\mathscr{L}_{\mathscr{F}_2}(\theta) = \sum_{k=8}^{12} \omega_k \left[\frac{1}{N_{\mathscr{F}}} \sum_{i=1}^{N_{\mathscr{F}}} |f_k\left(\hat{u}_0(X;\theta), x_n\right)|^2 \right] \tag{2.14}$$

$$\mathscr{L}_{\mathscr{F}}(\theta) = \mathscr{L}_{\mathscr{F}_1}(\theta) + \mathscr{L}_{\mathscr{F}_2}(\theta) \tag{2.15}$$

The objective function $\mathscr{L}_{\mathscr{F}}(\theta)$ quantifies the mean square error between the residuals of the physics-informed equations, whereas $\mathscr{L}_{\text{data}}(\theta)$ quantifies the same for the residuals of the observation I data. Weights $\omega_1, \omega_2, \ldots, \omega_{12}$ are assigned to the physical constraints, and N_d and $N_{\mathscr{F}}$ indicate the respective batch sizes. The function $u(X)$ denotes the anticipated future measurements from the IMU, with t representing the corresponding output timestamps and F denoting the forecast horizon. Furthermore, $g\left(\hat{u}_0(X;\theta), X_n, t\right)$ and $f\left(\hat{u}_0(X;\theta), X_n, t\right)$ correspond to ordinary and algebraic equations respectively [15]. These equations were employed as regularization terms, enhancing the neural network by enforcing the laws of physics by minimizing the physics-based loss function.

In the context of partial differential equations (PDEs), the physics-informed neural network (PINN) typically uses the loss function $\mathscr{L}_{\mathscr{F}}(\theta)$ to penalize deviations from physical principles. However, for the continuous time modeling and prognostic challenges addressed herein, the PDE-centric $\mathscr{L}_{\mathscr{F}}(\theta)$ was deemed unsuitable because robot dynamics are generally described by ODEs and algebraic equations. To overcome this, we adapted the method employed by physics-constrained neural networks (PCNN) and neural ordinary differential equations (NODEs)[16]. A PCNN is a specialized form of PINN that implements a regularization coefficient to balance the emphasis between data-driven and knowledge-driven regularization. In the NODEs framework, the neural network's hidden states are treated analogously to states in an ODE, and an ODE solver is utilized to compute the temporal evolution of these states. In our approach, we employ ODEs to capture the dynamic progression of the system and utilize algebraic equations to depict the robot dynamics model accurately. The PINN module generate signals corresponding to proxy-states, which signifies the derivatives of the state variable $\dot{x}$ within the LTI state-space representation. The inputs and outputs pertaining to the PINN module are designated as follows:

The PINN module was responsible for yielding signals indicative of proxy-states. These proxy-states encapsulated the derivative of the state variable $\dot{x}$ in relation to the LTI state space. The variables fed into and emerging from the PINN module were accordingly labeled as:

$$X_n^{(1-15)} = \left[a_{x_n}, a_{y_n}, a_{z_n}, \omega_{x_n}, \omega_{y_n}, \omega_{z_n}, v_{x_n}, v_{y_n}, v_{z_n}, \varphi_n, \Theta_n, psi_n, d_{E_n}, d_{N_n}, d_{U_n}\right] \tag{2.16}$$

$$\hat{u}_\theta(X;\theta)^{(1-6)} = \left[a_x, a_y, a_z, \omega_x, \omega_y, \omega_z\right] \tag{2.17}$$

The ODEs and the state $z_t^{(7)}$ represent the position change of the robot, which were represented as:

$$z_t^{(7)} = \sqrt{(E^t - E^n)^2 + (N^t - N^n)^2 + (U^t - U^n)^2} = \sqrt{(d_{E_t})^2 + (d_{N_t})^2 + (d_{U_t})^2} \tag{2.18}$$

$$g_q\left(\hat{u}_\theta(X;\theta), X_n, z_t\right) = X_n^{(q+6)} + \hat{u}_\theta(X;\theta)^{(q)} \times d_t - z_t^{(q)}, \quad q = 1, \ldots, 6 \tag{2.19}$$

$$g_7\left(\hat{u}_\theta(X;\theta), X_n, z_t\right) = \sum_{i=1}^{3}\left[X_n^{(i+6)} \times d_t + \frac{\hat{u}_\theta(X;\theta)^{(i)} \times d_t^2}{2}\right] - z_t^{(7)} \tag{2.20}$$

$$z_t^{(1-6)} = \left[v_{x_t}, v_{y_t}, v_{z_t}, \varphi_t, \Theta_t, \psi_t\right] \tag{2.21}$$

where $X_n^{(7-12)}$ are the initial states of ODE outputs; $z_t^{(1-6)}$ are the measurements of ODE outputs at timestamp t; d_t is the time interval between n and t. By minimizing $g_1 \sim g_6$, the proxy-states could incorporate information from the related robot states. The state $z_t^{(7)}$ represents the position change of the robot, $z_t^{(8,9,10)} = \left[d_{E_t}, d_{N_{t'}} d_{U_t}\right]$,where $[E, N, U]$ are the outputs of variable y_t in the LTI state-space robot model. By minimizing g_7, the physical

knowledge of y_t was incorporated into the physics-informed loss function. The formula for position updating of the robot dynamics may impact the learning effectiveness of the neural network, and hence, the Euler integral is employed to calculate the displacement of the robot. Furthermore, utilizing the loss calculation method of Physics-Informed Neural Networks (PINN) allows for combining ODE robot dynamics with a data-driven model, considering multiple sources.

The simple dependence relationships among robot states are captured through algebraic equations. The algebraic equations representing the linear dynamics in the LTI state space are expressed as shown in equation 2.5. Minimizing the loss of $\hat{u}_\theta(X;\theta) - \dot{x}$ facilitates the incorporation of the physical dynamics model into the physics-informed loss function. The referenced robot dynamics model is based on the two-degree-of-freedom (2-DOF) robot dynamics model. To simplify the robot dynamics, the longitudinal and latitudinal dynamics are decoupled by neglecting the influence of the latitudinal and longitudinal forces.

In the Euler-Lagrange formulation, the torque required at each joint can be computed as:

$$\tau = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}}\right) - \frac{\partial P}{\partial q} + \tau_{\text{friction}} + \tau_{\text{external}} \tag{2.22}$$

where is the kinetic energy of the robot, which depends on $\dot{q}$. P is the robot's potential energy, which depends on $q.\tau_{\text{friction}}$ represents the torques due to friction in the joints. τ_{external} includes torques due to external forces (like interaction with the environment).

$$\tau_n^{(1-4)} = H(q_n)\,\ddot{q}_n + C(q_n, \dot{q}_n)\,\dot{q}_n + G(q_n) + \tau_{\text{friction}}(q_n, \dot{q}_n) + \tau_{\text{external}} \tag{2.23}$$

where: $H(q_n)$ is the inertia matrix at the current configuration q_n. $\ddot{q}_n$ is the joint acceleration, which can be derived from $\dot{q}_n$ through numerical differentiation or estimation. $C(q_n, \dot{q}_n)$ is the Coriolis and centrifugal forces matrix. $G(q_n)$ represents the gravitational

torque vector. $\tau_{\text{friction}}\ (q_n, \dot{q}_n)$ includes torques due to friction, which can be a function of both q and $\dot{q}$.

In our approach, we use IMU sensor and joint encoder kinematics to replace τ_{friction} and τ_{external} , hence the updated new torque equation:

$$\begin{aligned}\tau_n^{(1-4)} =& H(q_n)\,\ddot{q}_n + C(q_n, \dot{q}_n)\,\dot{q}_n + G(q_n) \\ &+ J^T(q_n)\,F_{\text{interaction}}\left(q_n, \dot{q}_n, p_{f_n}, v_{f_n}, \alpha_n, \omega_n\right)\end{aligned} \tag{2.24}$$

In this updated equation: $H(q_n)$ is the inertia matrix at the joint configuration q_n. $\ddot{q}_n$ is the joint acceleration, potentially computed from $\dot{q}_n$. $C(q_n, \dot{q}_n)$ is the matrix of Coriolis and centrifugal forces. $G(q_n)$ represents the gravitational torques, which depend on the configuration q_n. $J(q_n)$ is the Jacobian matrix converting forces at the foot to torques at the joints. $F_{\text{interaction}}$ is now a function that includes the linear acceleration (α_n) and angular velocity (ω_n) from the IMU, along with joint kinematics to estimate the interaction forces at the robot's feet. which combined joint torque we have new equation:

$$\tau_{\text{total},n} = \tau_{\text{Hip Roll},n} + \tau_{\text{Hip Pitch},n} + \tau_{\text{Knee Pitch},n} \tag{2.25}$$

Total Torque About the Center of Mass (τ_{total}) :

$$\tau_{\text{total}} = \sum_{i=1}^{4} \sum_{j=1}^{n_j} J_{ij}^T\left(q_{ij}\right)\tau_{ij} \tag{2.26}$$

where i indexes the legs of the robot (1 to 4 for a quadruped), j indexes the joints in each leg (n_j is the number of joints per leg). J_{ij} is the Jacobian matrix for joint j in leg i, q_{ij} are the joint angles for joint j in leg i. τ_{ij} is the torque at joint j in leg i.

Longitudinal Force Component From Total Torque:

$$F_{x_{\text{total}}} = \frac{1}{r_{\text{eff}}}\left(\sum_{i=1}^{4} \tau_{\text{total},i} \cdot \cos(\theta_i)\right) \tag{2.27}$$

where $F_{x_{\text{total}}}$ is the total longitudinal force acting on the robot, $\tau_{\text{total},i}$ is the component of the total torque from leg i contributing to longitudinal force, r_{eff} is the effective distance from the ground contact point to the center of mass where the torque acts to produce force, θ_i is the angle of application of the force relative to the horizontal axis, affecting its longitudinal component.

Latitudinal Force Component From Total Torque:

$$F_{y_{\text{total}}} = \frac{1}{r_{\text{eff}}} \left(\sum_{j=1}^{4} \tau_{\text{total},j} \cdot \sin\left(\theta_j\right) \right) \tag{2.28}$$

where $F_{y_{\text{total}}}$ is the total longitudinal force acting on the robot, $\tau_{\text{total},j}$ is the component of the total torque from leg j contributing to latitudinal force, r_{eff} is the effective distance from the ground contact point to the center of mass where the torque acts to produce force, θ_j is the force's application angle relative to the horizontal axis, affecting its longitudinal component.

The torques τ_{ij} can be derived from motor commands, and the Jacobian matrices J_{ij} translate these torques to forces at the robot's feet. The effective radius r_{eff} and angles θ_i would be determined based on the robot's geometry and the feet' position during the stance phase of locomotion.

The terms α_n and ω_n come from the IMU sensor data at time n, which provide the necessary dynamic information about the robot's acceleration and rotational movement, contributing to estimating forces due to environmental interactions. As determined by the contact estimation network using these inputs, the contact state will inform the $F_{\text{interaction}}$ term. For instance, if the contact state indicates a foot is in the air, the corresponding elements in $F_{\text{interaction}}$ related to that foot should be 0, reflecting no ground interaction forces for that limb.

2.3.3 Contact Estimation

This section discusses our deep learning-based approach for estimating contact states. We define the contact state for each leg, labeled as $l \in \{RF, LF, RH, LH\}$, using the vector $C = \begin{bmatrix} c_{RF} & c_{LF} & c_{RH} & c_{LH} \end{bmatrix}$, where $c_l \in \{0, 1\}$. Here, '0' represents no contact, and '1' indicates firm contact with the ground. Given the input data, our goal is to correctly estimate the contact state C, treating this as a classification problem within our deep neural network. The contact state vector C can assume one of 16 possible states, ranging from all feet in the air to all feet making contact with the ground. We convert the binary vector C to a decimal state $S \in \{0, 1, \ldots, 15\}$ for easier processing. For instance, a contact state $C_i = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$ is converted to $S_i = 6$.

The contact estimation network is composed of two convolutional blocks and three fully connected layers, as shown in Figure 2.4. Each block includes two one-dimensional convolutional layers followed by a one-dimensional max pooling layer. The convolutional layers are designed to extract deep features from the input data. To enhance computational efficiency in terms of memory usage and run time, a one-dimensional kernel is employed. This kernel traverses the time domain, with padding applied to maintain data dimensions. The ReLU activation function is used for nonlinearity in the convolutional layers. To prevent overfitting, a dropout mechanism is applied to the second convolutional layer. Finally, a one-dimensional max-pooling layer is added at the end of each block to downsample the data.

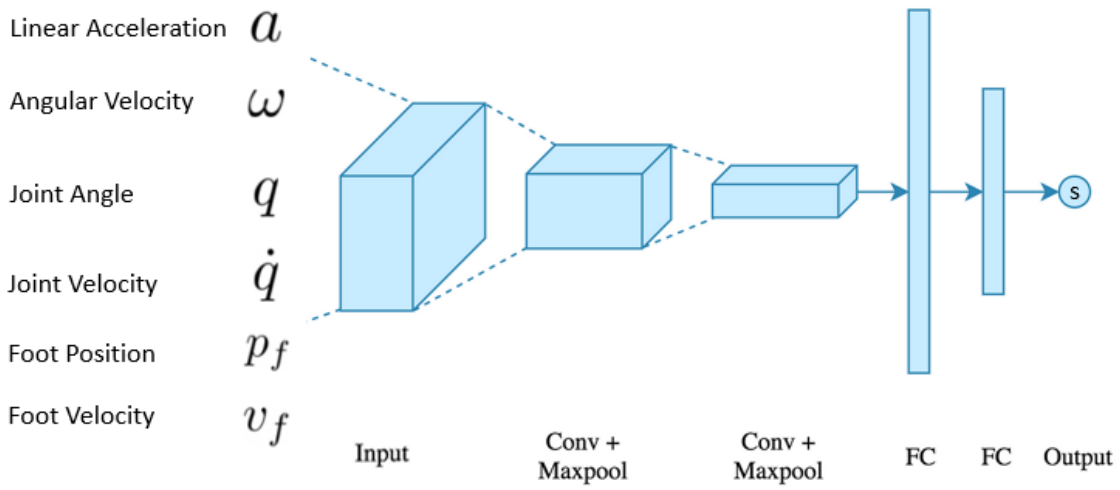

Figure 2.4: The architecture of the contact estimation network

2.3.4 Processing Contact Estimation Input Data

The contact estimation network takes sensor measurements from the joint encoder, IMU, and kinematic, avoiding the force and torque sensor as input. For a synchronized time n, the sensor measurements are concatenated as $z_n^{(1-6)} = [q_n \quad \dot{q}_n \quad a_n \quad \omega_n \quad P_{fn} \quad v_{fn}]$

$$a_n^{(1-3)} = \begin{bmatrix} a_{xn} & a_{yn} & a_{zn} \end{bmatrix} \tag{2.29}$$

$$w_n^{(1-3)} = \begin{bmatrix} w_{xn} & w_{yn} & w_{zn} \end{bmatrix} \tag{2.30}$$

$$q_n^{(1-12)} = \begin{bmatrix} q_{RF1n} & q_{RF2n} & q_{RF3n} & q_{LF1n} & \cdots & q_{LH3n} \end{bmatrix} \tag{2.31}$$

$$\dot{q}_n^{(1-12)} = \begin{bmatrix} \dot{q}_{RF1n} & \dot{q}_{RF2n} & \dot{q}_{RF3n} & \dot{q}_{LF1n} & \dots & \dot{q}_{LH3n} \end{bmatrix} \tag{2.32}$$

$$p_{f_n^{(1-12)}} = \begin{bmatrix} p_{PRFxn} & p_{PRFyn} & p_{PRFzn} & p_{PLFxn} & \dots & p_{PLHzn} \end{bmatrix} \tag{2.33}$$

$$v_{f_n^{(1-12)}} = \begin{bmatrix} v_{VRFxn} & v_{VRFyn} & v_{VRFzn} & v_{VLFxn} & \dots & v_{VLHzn} \end{bmatrix} \tag{2.34}$$

where $q_n^\top$ is a 12×1 vector of joint encoder measurements (rad) at time n, $\dot{q}_n^\top$ is a 12×1 vector of joint angular velocity (rad/sec), a_n is the linear accelerations (m/sec) from the IMU in the IMU frame, ω_n is the angular velocity (rad/sec) in the IMU frame, $p_{fn}^\top$ is a 12×1 vector with foot positions calculated from forward kinematics, and $v_{fn}^\top$ is a 12×1 vector that carries the linear velocities of each foot. Both p_{fn} and v_{fn} are represented in the robot's hip frame. To infer the relationship between data across the time domain, we create a window with size w and append previous measurements within this window into a 2D array $D_n = \begin{bmatrix} z_{n-w}^\top & z_{n-w+1}^\top & \dots & z_n^\top \end{bmatrix}^\top$ D_n is a $w \times 54$ array. The network takes D_n as input and estimates the contact state S_n as output each time.

2.3.5 The UKF Module

The UKF-Manifolds is a specialized variant of the UKF designed for environments where the state space is characterized by manifold structure. This adaptation is particularly significant in complex applications such as quadrupedal robot localization and state estimation. Unlike traditional approaches where state variables are simple Euclidean vectors, in these robots, the state variables often reside on intricate manifolds, such as rotation matrices within the special orthogonal group. UKF-Manifolds strategically selects sigma points

on these manifolds and utilizes their geometric properties for more precise state estimation. This methodology, not only maintains the validity and accuracy of state estimates, but also prevents the emergence of invalid state values—like the loss of orthogonality in rotation matrices—which are common pitfalls in standard UKF implementations. This precision is crucial in quadrupedal robots, where accurate state estimation directly influences stability and maneuverability, enhancing the robot's ability to navigate and interact with complex environments effectively.

For stochastic process on Riemannian manifolds, the theory of Lie groups [17] is used to define the robot's state estimation; the IMU gyro sensor-fusion model is a UKF-M based filter, which is a standard 3D kinematic model based on internal inputs. The PINN-UKF is based on the methodology proposed in [18]. The modification of the PINN UKF involved uses the outputs of the PINN module to mitigate IMU drift.

The states of a moving robot in a discrete dynamic system are represented as:

$$\chi_n \in \mathcal{M} = \left\{ C_n \in \mathbb{R}^{3\times3}, v_n \in \mathbb{R}^3, \mathcal{P}_n \in \mathbb{R}^3, b_{g_n} \in \mathbb{R}^3, b_{a_n} \in \mathbb{R}^3 \right\} \tag{2.35}$$

where χ_n denotes the state of a robot belonging to a parallelizable manifold $\mathcal{M}$; n is the current timestamp; $v_n = (v_{E_n}, v_{N_n}, v_{U_n})$ is the velocity vector v_{E_n} velocity east, v_{N_n} velocity north; $\mathcal{P}_n = (E_n, N_n, U_n)$ is robot coordinate in the navigation coordinates; $b_{g_n} = \left(b_{\omega_{x,n}}, b_{\omega_{y,n}}, b_{\omega_{z,n}}\right)$ is the gyro bias; $b_{a_n} = \left(b_{a_{x,n}}, b_{a_{y,n}}, b_{a_{z,n}}\right)$ is the accelerometer bias; and C_n is a special orthogonal group that represents 3D rotation [17].

$$\mathrm{SO(3)} := \left\{ C_n \in \mathbb{R}^{3\times3} \mid C_n C_n^{\mathrm{T}} = I_3, \det C_n = 1 \right\} \tag{2.36}$$

where I_3 is the identity matrix. Based on the time derivative of $C_n C_n^{\mathrm{T}} = I_3$, a skew-symmetric matrix $C_n^{\mathrm{T}} \dot{C}_n$ was obtained:

$$C_n^{\mathrm{T}}\dot{C}_n + \dot{C}_n^{\mathrm{T}}C_n = 0 \tag{2.37}$$

The $C_n^T\dot{C}_n$ as a skew-symmetric matrix is often noted as $[\boldsymbol{\omega}]_\times$:

$$C_n^{\mathrm{T}}\dot{C}_n = [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\boldsymbol{\omega}_z & \boldsymbol{\omega}_y \\ \boldsymbol{\omega}_z & 0 & -\boldsymbol{\omega}_x \\ -\boldsymbol{\omega}_y & \boldsymbol{\omega}_x & 0 \end{bmatrix} \tag{2.38}$$

where $[\boldsymbol{\omega}]_\times$is in the Lie algebra of $SO(3)$. The Lie algebra is a vector space and can be decomposed into:

$$[\boldsymbol{\omega}]_\times = \boldsymbol{\omega}_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} + \boldsymbol{\omega}_y \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} + \boldsymbol{\omega}_z \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.39}$$

where $\omega = \left[\omega_x, \omega_y, \omega_z\right]$ is in the vector of angular velocities. For the ω constant, the solution of the ODE is:

$$C_n = \exp\left([\boldsymbol{\omega}]_\times n\right) \tag{2.40}$$

where exp() is the exponential map on the SO(3) and $C_0 = I$. The exp() map was derived from the time derivatives of $\chi_n \in \mathcal{M}$

Analogous to the Gaussian belief in the Kalman filter, the UKF-M algorithm constructs a probability distribution as $\chi_n \sim \mathcal{N}\left(\hat{\chi}_n, P_n\right)$ for the random variable $\chi_n \in \mathcal{M}$ as:

$$\chi_n = \varphi\left(\hat{\chi}_n, \xi_n\right), \quad \xi_n \sim \mathcal{N}\left(0, \mathrm{P}_n\right) \tag{2.41}$$

where $\hat{\chi}_n$ is viewed as the mean estimate at timestep n; φ is the propagation function; $\xi_n \in \mathbb{R}^d$ is a random Gaussian vector; $\mathcal{N}$ is the Gaussian distribution; and $\mathrm{P}_n \in \mathbb{R}^{d\times d}$ is the

covariance matrix. $\varphi(\hat{\chi}_n, \xi_n) \in \mathcal{M}$ is derived by initiating from $\hat{\chi}_n$ and integrating $\sum_{i=1}^{d} \xi_n^i V_i$ where d represent the dimension of the associated vector fields.

Given that the probability distribution of χ_n is denoted as $p(\chi_n)$. further information regarding χ_n is acquired through the measurement y_n as:

$$y_n = h(\chi_n) + v_n \tag{2.42}$$

where h represents the observation function, and $v_n \sim \mathcal{N}(0, \mathrm{R}_n)$ signifies the white Gaussian noise. The UKF module utilized gyro measurements and acceleration as inputs to update the random variable χ. The measurements of this standard 3D kinematic model were expressed as:

$$y_n = \left\{\mu_n \in \mathbb{R}^3, a_{b_n} \in \mathbb{R}^3\right\} \tag{2.43}$$

where $\mu_n = \left(\omega_{x_n}, \omega_{y_n}, \omega_{z_n}\right)$ represents the gyroscope, and $a_{b_n} = \left(a_{x_n}, a_{y_n}, a_{z_n}\right)$ denotes the accelerometer. The UKF-M algorithm[18] updated the state and covariance by combining measurements y_n and system states χ_n.

In PINN UKFM, the output of the PINN module was utilized to filter the noise and minimize the norm errors. Proxy-states $\hat{u}_\theta$ serves as the calibrated IMU measurements for the filtering process. The states of the proxy-states are represented as:

$$y_n = y_n' = \hat{a}_\theta = \left\{\mu_n \in \mathbb{R}^3, a_{b_n} \in \mathbb{R}^3\right\} \tag{2.44}$$

where $y_n{}' = \hat{u}_\theta$ represents the PINN module output as the proxy-states. Using the propagation function[18], PINN UKF built the robot model. First, the gyro measurements are used to calculate the rotation matrix:

$$C_{n+1} = C_n \exp\left(\left(\mu_n - b_{g_n} + w_n^{(0:3)}\right) \times d_t\right) \tag{2.45}$$

Then the robot acceleration is used to calculate the calibrated robot acceleration.

$$a = C_n \left(a_{b_n} - b_{a_n} + w^{(3:6)} \right) + g \tag{2.46}$$

where $g = [0, 0, -9.8]$ represents the gravitational constant and a represents the calibrated acceleration. Based on the calibrated robot acceleration, the model updated the speed as:

$$v_{n+1} = v_n + a \times d_t \tag{2.47}$$

where $v_n = (v_{E_n}, v_{N_n}, v_{U_n})$ denotes the velocity vector. Based on the velocity vector, the model updated the position vector as:

$$\mathcal{P}_{n+1} = \mathcal{P}_n + v_n \times d_t + \frac{a d_t^2}{2} \tag{2.48}$$

where $\mathcal{P}_n = (E_n, N_n, U_n)$ is the coordinates in the navigation coordinates. Finally, the model uploaded the gyro and accelerometer biases as:

$$b_{a,n+1} = b_{a,n} + w^{(6:9)} d_t \tag{2.49}$$

$$b_{g,n+1} = b_{g,n} + w^{(9:12)} d_t \tag{2.50}$$

where $b_{g,n}$ represents the gyro bias, and $b_{a,n}$ represents the accelerometer bias.

The probability distribution of χ and the propagation function remain constant; thus, the posterior distribution $p(\chi_n \mid y_n')$ is calculated as $p(\chi_n \mid y_n)$. The proxy-states y_n' provids information about the sigma point ξ_n. First, the sigma points ξ_n are computed as:

$$\chi_{k|n}^{(i)} = \begin{cases} \mu_n & \text{if } i = 0, \\ \mu_n + \left(\sqrt{(d+\lambda) P_n} \right)_i & \text{if } 1 \le i \le d, \\ \mu_n - \left(\sqrt{(d+\lambda) P_n} \right)_{i-d} & \text{if } d+1 \le i \le 2d, \end{cases} \tag{2.51}$$

where $\lambda = (\alpha^2 - 1)d,$ and d is the dimensionality of the state.

where λ is the scale parameter [19]; α is a tuning parameter chosen by the practitioner [20] (typically small to ensure stability); d is the dimensionality of the state space; P_n is the covariance matrix at time step n; and $\text{col}(\cdot)_j$ extracts the j^{th} column from the matrix, representing the weight associated with the j^{th} sigma point.

These sigma points are then propagated through the robot model to yield the set of transformed sigma points y_{j_n}.

Next, the PINN-UKFM utilizes the Kalman update equations to refine the state estimate and covariance as follows:

$$K_n = P_{\xi_n y_n} P_{y_n y_n}^{-1} \tag{2.52}$$

$$\hat{x}_n^+ = \varphi\left(\hat{x}_n, K_n(y_n^{'} - \bar{y}_n)\right) \tag{2.53}$$

$$P_n^+ = P_n - K_n P_{y_n y_n} K_n^T \tag{2.54}$$

where $y_n^{'}$ is the output from the PINN module (proxy-states), K_n is the Kalman gain matrix, $\hat{x}_n^+$ is the updated state estimate, P_n is the prior covariance matrix of the state, and P_n^+ is the updated covariance matrix.

The unscented transformation[21] approximates the posterior $p\,(\zeta_n \mid y_n{}')$ for ξ_n as:

$$p\,(\xi_n \mid y_n^{'}) \sim \mathcal{N}\left(\overline{\xi_n}, \mathrm{P}_n^+\right) \tag{2.55}$$

$$\overline{\xi_n} = \mathrm{K}_n \left(y_n' - \overline{y_n} \right) \tag{2.56}$$

where $\overline{\xi_n}$ represents the gaussian white noise mean.

The unscented approximation to the posterior $p\left(\xi_n \mid y_n'\right)$ is thus the distribution of a Gaussian $\overline{\xi_n} + \bar{\xi}_n +$ with $\bar{\xi}_n{}^+ \sim \mathcal{N}\left(0, \mathrm{P}_n{}^+\right)$[18]. Then, PINN UKF approximates the posterior distribution $p\left(\chi_n \mid y_n'\right)$ as:

$$\chi_n \approx \varphi\left(\hat{\chi}_n^+, \bar{\zeta}_n^+\right), \quad \bar{\zeta}_n^+ \sim \mathcal{N}\left(0, \mathrm{P}_n^+\right) \tag{2.57}$$

$$\hat{\chi}_n^+ = \varphi\left(\hat{\chi}_n, \overline{\xi_n}\right) \tag{2.58}$$

$$\chi_n \approx \varphi\left(\varphi\left(\hat{\chi}_n, \overline{\xi_n}\right), \xi_n^+\right) \tag{2.59}$$

where $\xi_n{}^+$represents the posterior noise. The posterior distribution $p\left(\chi_n \mid y_n'\right)$ reduces to a Bayesian estimation problem [17] that incorporated the information from the PINN module.

2.4 Experimental Result

2.4.1 Validation of Contact Estimator

In validating our proposed estimation framework, we analyze the estimated ground reaction forces and foot velocities overlapped with a single leg's estimated and ground

truth contacts in a forest dataset. As shown in figure 2.5, the foot velocity is captured in two subplots: the upper subplot presents the foot velocity V_z with ground truth contact phases indicated in yellow, while the lower subplot demonstrates V_z with the estimated contact phases shaded in blue. The congruence between the shaded regions and the foot velocity fluctuations reveals the accuracy of contact phase estimation, affirming the model's capability to discern contact events.

Similarly, the ground reaction forces represented as F_x, F_y, and F_z, are depicted in the two lower graphs, again with ground truth contacts and estimated contacts. The estimated contact periods align closely with spikes and troughs in F_x, F_y, and F_z, which correspond to the dynamics of actual foot-ground interactions, underlining the proficiency of our contact estimator.

The estimated contacts exhibit a high degree of overlap with the ground truth contacts, suggesting that the estimation method can reliably predict contact events, even in the forest dataset's inherently unpredictable terrain. This overlap is indicative of the model's potential to adaptively learn and accurately represent the leg's interaction with a complex environment. Moreover, comparing the estimated ground reaction forces and the ground truth indicates that our method can effectively capture the nuanced physical interactions between the robot leg and the forest floor.

2.4.2 Validation of PINN-UKF

This subsection presents the demonstration of the UKF module, which is built upon the output of the PINN module. The main objective of this module is to reduce the effect of state noise using the Gaussian noise hypothesis. The outcomes achieved with PINN-UKFM surpassed those obtained with the only virtual sensor algorithm. Without coordinate input,

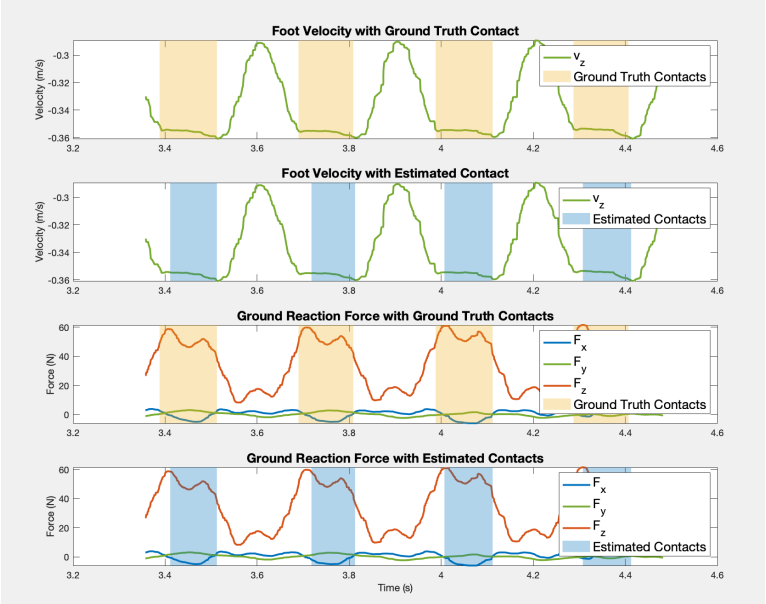

Figure 2.5: Contact Estimation and Ground Reaction Force Prediction

the robot coordinates are updated in the robot model using the UKF module. The PINN module effectively mitigated the impact of robot sensor drift and yielded a dependable state estimation that closely approximate the true state of the robot. Thus, the updated robot position of PINN UKFM is also closer to the true state.

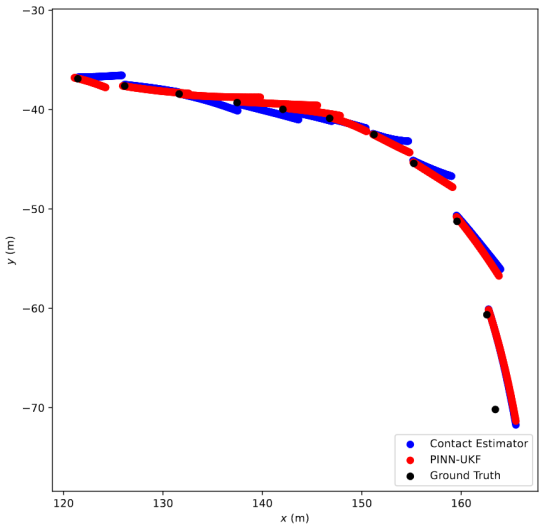

Figure 2.6: Compare trajectories between virtual sensor only and PINN-UKF

Additionally, the UKF-M module could estimate the robot's speed, which included [ve, vn, vu]. Next, the modeling results were compared to the measurement states to verify PINN-UKF correctness. As shown in Figure 2.7, compared to the UKF algorithm, PINN-UKF provides a better estimate of vu.

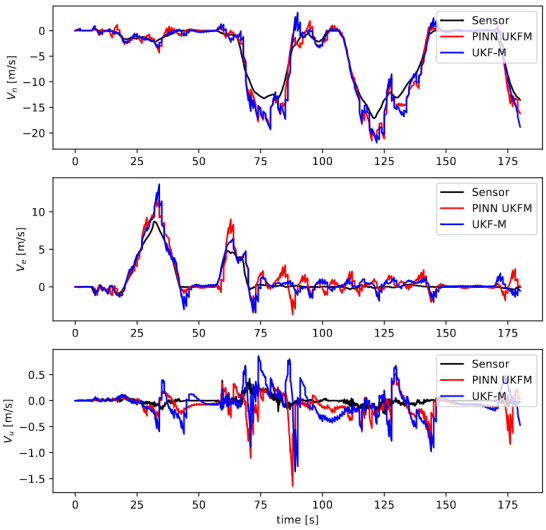

Figure 2.7: Compare 3D velocities implement with PINN

2.5 Conclusion

This study has demonstrated a significant advancement in the field of robotic state estimation through the strategic integration of an Unscented Kalman Filter (UKF) with a Physics-Informed Neural Network (PINN). Our research has showcased how this combination can rigorously constrain the dynamic model of a robot, thereby enhancing the accuracy and reliability of the state estimation process. The UKF, renowned for its capability to handle non-linearities and uncertainties inherent in robotic navigation, when paired with the PINN, ensures that the dynamic state of the robot adheres closely to the underlying physical laws. This synergy mitigates the risk of model divergence and significantly

reduces the detrimental effects of IMU sensor drift over time.

Further strengthening our methodology, implementing a learning-based contact estimator using proprioceptive sensory data has proven to be a pivotal innovation. Unlike traditional methods that rely heavily on external sensors or visual data, our approach utilizes intrinsic sensory inputs to detect contact events. This method has shown exceptional promise in environments where external sensors fail or are impractical, providing a robust alternative that maintains accuracy and operational integrity in obscured or complex settings. The contact estimator plays a crucial role in reinforcing the robot's navigation system, effectively reducing reliance on potentially erratic IMU data and thus, curbing sensor drifting.

In conclusion, the combined use of UKF and PINN forms a robust framework for robot state estimation, ensuring high fidelity in dynamic modeling and reducing sensor drift. Meanwhile, the contact estimator emerges as a crucial component, enhancing the system's resilience to environmental variabilities and sensor inaccuracies. Our findings affirm the potential of this integrated approach to significantly improve robotic systems' operational reliability and efficiency, particularly in challenging real-world applications. Future work will focus on refining these techniques and exploring their applicability across different robotic platforms and more diverse operational contexts.

The integration of Physics-Informed Neural Networks (PINN) with Unscented Kalman Filters (UKF) has demonstrated significant advances in robotic state estimation. However, the limitations inherent in the physical constraint models used within this PINN-UKF framework, alongside the challenges posed by the rigid dynamics of traditional robotic platforms, highlight critical areas for improvement. The next chapter introduces an innovative spine implementation to address these issues. This development moves away from traditional rigid body constraints, embracing a more flexible dynamic system that better ac-

commodates the complex movements of quadruped robots. Furthermore, the approach incorporates learning-based methods to effectively manage the chaotic dynamics introduced by the spine implementation, thereby offering a robust solution to the limitations identified in the initial methodology.

CHAPTER 3

INTELLIGENT DESIGN FOR QUADRUPED ROBOT ON A DYNAMIC, FLEXIBLE SURFACE WITH AN ACTIVE SPINE

3.1 Introduction

This chapter addresses the limitations of the previously discussed Physics-Informed Neural Network (PINN) and Unscented Kalman Filter (UKF) methodology. While these tools effectively enhance robot state estimation, they struggle with physical constraints and rigidity when navigating complex terrains. The conventional rigid body model typically employed offers limited flexibility and shock absorption, posing significant challenges in dynamic environments. To counter these issues, this study introduces an innovative modification: replacing the traditional rigid body with a dynamic spine implementation. This change significantly improves flexibility and shock absorption, which are critical for maneuvering through challenging terrains. However, integrating a spine introduces complexity by altering the robot's entire dynamic system during movement, making traditional modeling techniques less effective due to the chaotic nature of the new dynamics. Reservoir Computing is employed to manage this complexity, providing a robust solution for handling the chaotic dynamics induced by the spine, thereby ensuring more accurate and adaptable state estimation in dynamic environments.

The approach incorporates the design of a quadruped robot equipped with a dynamic spine, utilizing Reservoir Computing to enhance performance. Four bending sensors are integrated within the spine to modulate spinal forces for balance and enable closed-loop control of the Center of Mass (CoM) relative to dynamic surfaces. This passive system modeling allows the robot with a flexible spine to traverse rough terrain more efficiently, with increased agility and reduced impact forces, thereby improving overall mechanical

performance. Moreover, the paper introduces the development of a novel, intelligent control mechanism tailored for quadrupedal robots operating on dynamic, flexible surfaces. This system relies on a hierarchical distributed control mechanism that enables the legs to adjust centralized frames for optimal functional performance cooperatively. This approach not only circumvents the physical limitations of traditional methods by leveraging learning-based Reservoir Computing, it also avoids the need for conventional contact estimators in state estimation, establishing a new paradigm in robotic design and functionality.

3.1.1 Motivation

The design and analysis of dynamic systems, especially chaotic ones, have always been challenging tasks [22] [23]. To address these issues, we proposes a novel methodology that combines together the use of Reservoir Computing, Lyapunov control algorithm, and bending sensors integrated with a continuous dynamics spine. The central objective is to design a quadruped robot with a dynamic spine capable of navigating uneven terrain while minimizing energy costs and enhancing agility [24] [25]. We also prove the stability of the proposed system to ensure a successful implementation, addressing the issues of chaotic behavior[26]. The developed model has possible applications in fields such as search and rescue operations, the military sector, mining, construction, and robotics for autonomous exploration and monitoring [27]. Additionally, the results can help researchers better understand the dynamic behavior of legged robots, contributing to the improvement of future designs.

3.1.2 Method Rationale

The proposed approach utilizes Reservoir Computing to achieve our goal and integrates a bending sensor into a continuous dynamics spine.[26] [28] This allows the reservoir to adjust spine force balancing and use closed-loop control to continuously map the Center of Mass (CoM) onto the dynamic surface [29]. Additionally, hierarchical distributed control mechanisms developed for each leg are utilized to change centralized frames for better functional reflection cooperatively [29]. Employing these methods enables the training of the quadruped robot on uneven surfaces in simulation and deployment in the real world with the ability to adapt to changing environments [24].

3.1.3 Applications

The developed model has significant applications in the field of robotics for better understanding the dynamic behavior of legged robots and soft robots better, leading to the design of more effective and stable robots. In soft robotics, the proposed methodology could contributes to overcoming the challenges related to flexible structures' control. By integrating bending sensors into the continuous dynamics spine and employing reservoir computing to adjust spine force balancing, soft robots with flexible spines could navigate uneven terrain while reducing energy costs and improving agility [24]. The results are applicable in various fields, including search and rescue operations and the medical sector, where soft robots' ability to operate around delicate structures are essential [25].

3.2 Methodology

This section introduces a state-space model for the coupled mechanical system of a quadrupedal robot with a spine. This decomposition results in two bipedal systems, into which Reservoir Computing is integrated to model the dynamics of the quadrupedal robot via spine state measurement [30]. Once the dynamic behavior of the quadruped robot with the spine is known, the Lyapunov Control algorithm is employed to ensure stability [31]. Additionally, a feedback loop is utilized to verify whether the robot center of mass is mapping onto the dynamic surface accurately [29].

3.2.1 Problem Formulation

Consider a dynamical system consisting of two subsystems connected to a spine indexed by $\varepsilon \triangleq \{(1, s), (2, s), (s, 1), (s, 2)\}$. The notation (i, s) refers to the connection between subsystem i (either 1 or 2, representing the legs) and the spine. The compatibility constraint between the subsystems and the spine is given by $\lambda_e + \lambda_{\bar{e}} = -F_s$, where λ_e and $\lambda_{\bar{e}}$ represent the coupling forces between the subsystems and the spine, and F_s denotes the total external force applied to the spine. [31]Therefore, there are four variables for the coupling forces, corresponding to the four connections in the system.

The coupled mechanical system considered in this work is composed of two subsystems, with subsystem 1 representing the front legs and subsystem 2 the hind legs, where $q_i \in \mathbb{Q}_i \subset \mathbb{R}^{n_i}$ for $i = 1, 2$. The spine connects the two subsystems. The dynamical equations of motion for the coupled mechanical system[31] are given by:

$$\begin{cases} D_1\ddot{q}_1 + H_1 = B_1u_1 + J_e^\top \lambda_e + J_s^\top F_s \\ D_2\ddot{q}_2 + H_2 = B_2u_2 + J_{\bar{e}}^\top \lambda_{\bar{e}} - J_s^\top F_s \\ D_s\ddot{x}_s + H_s = B_su_s + J_e^\top \lambda_e + J_{\bar{e}}^\top \lambda_{\bar{e}} + F_s \\ \text{s.t. } c_{e,q}\left(q_1, q_2, x_s\right) \equiv 0, \lambda_e + \lambda_{\bar{e}} = -F_s \end{cases} \tag{3.1}$$

where, D_i, H_i, and B_i denote the inertia matrix, Coriolis and gravitational term, and actuation matrix respectively for subsystem i. $u_i \in \mathbb{R}^{m_i}$ denotes the control input for subsystem i. Additionally, D_s, H_s, and B_s denote the corresponding terms for the spine. The variables $x_s \in \mathbb{R}^{n_s}$ and $\ddot{x}_s$ are the position and acceleration of the spine, respectively. The term J_e ($J_{\bar{e}}$) is the Jacobian matrix which maps the subsystem variables to the connection variables corresponding to λ_e ($\lambda_{\bar{e}}$).

The compatibility constraint, $c_{e,q}(q_1, q_2, x_s) \equiv 0$, ensures that the displacement of the connection points e on the front legs and $\bar{e}$ on the hind legs matches the position of the spine. This constraint, along with the force-balance equation $\lambda_e + \lambda_{\bar{e}} = -F_s$, enforces compatibility between the subsystems and the spine, and completes the description of the coupled mechanical system.[31] The coupling forces $\lambda_e \in \Lambda_i \in \mathbb{R}^{l_i}$ can be solved explicitly using the following equation:

$$\begin{aligned} \lambda_e = \left(J_eD_1^{-1}J_e - J_{\bar{e}}D_2^{-1}J_{\bar{e}}\right)^{-1} \cdot \left[J_eD_1^{-1}\left(H_1 - B_1u_1\right) + J_{\bar{e}}D_2^{-1}\left(H_2 - B_2u_2\right)\right] \\ - \left(J_eD_1^{-1}J_e - J_eD_2^{-1}J_{\bar{e}}\right)^{-1}\left[j_e\dot{q}_1 + j_{\bar{q}}\dot{q}_2 + F_s\right] \end{aligned} \tag{3.2}$$

where, $J_e(q_1, q_2, x_s) = \partial c_{e,q}/\partial q_i$, where $i \in \{1, 2\}$, and $e \triangleq (i, j), \bar{e} \triangleq (j, i) \in \mathcal{E}$. Further, J_e and $J_{\bar{e}}$ are the Jacobians of the coupling constraints with respect to the subsystem configurations. The matrices D_1^{-1} and D_2^{-1} refer to the inverses of the mass matrices of the subsystems. Furthermore, $\dot{J}_e$ and $\dot{J}_{\bar{e}}$ represent the time derivative of the Jacobians with re-

spect to subsystem configurations. The term F_s represents the force generated by the ideal spine to balance the forces exerted by the front and hind legs.

3.2.2 Reservoir Computing

The spine state x_s is measured using bending sensors mounted along the spine's length to model the system's dynamic behavior using Reservoir Computing [22]. The bending sensors' signals are then processed through a Reservoir Computing layer that employs Echo State Networks (ESN) to generate a time series prediction of the system's state [30]. The Reservoir Computing layer comprises three major components: the input, reservoir layers, and readout. The input layer maps the bending sensor signals into the Reservoir Computing space, while the reservoir layer is composed of a randomly connected network of neurons that transforms the input into a high-dimensional feature space. The Reservoir Computing layer then feeds the transformed input into a linear readout layer to estimate the system's response, the only trainable part is the readout, which is ridge node, can be trained by state of training and $y[t]$, where the state of training is the activation of the reservoir trigger by $x[t]$ [31], as shown in Figure 3.1.

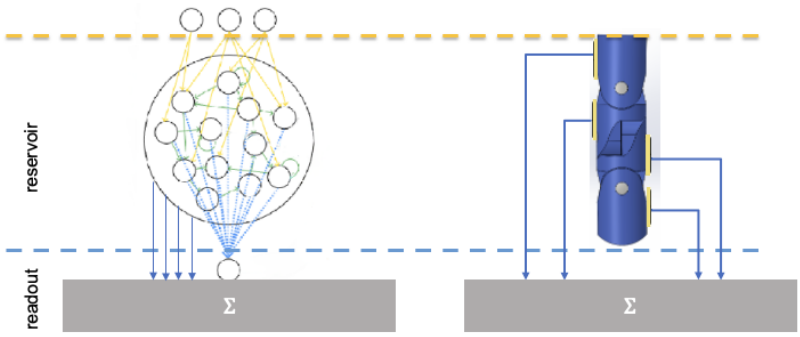

Figure 3.1: Figure shows the general picture of spine implementation with reservoir computing.

The spine in the figure 3.1 has four segments with three actuators generate force to balance the center of mass into the dynamic surface [29]. The spine serves as the internal structure and is enveloped by muscle-like material to mimic a real dog spine, creating a continuous model. This design is more suitable for collecting input data for reservoir computing, as shown in Figure 3.2.

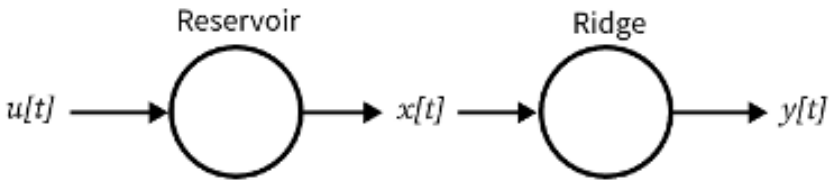

Figure 3.2: The basic logic of x[t] can trigger activation of reservoir, Reservoir and Readout corresponding Fig. 3.1 Reservoir and Ridge

3.2.3 Lyapunov Control Algorithm

The quadrupedal robot's spine is expressed as $F_s = h(x_s, u_1, u_2)$, where x_s is the spine state, and u_1 and u_2 are the external environment inputs. The desired value of spine expression noted as F_s^d, which is $F_s^d = h(x_s^d, u_1, u_2)$, in this case, if spine input u_s unknown, we can design $u_s$such that F_s goes to F_s^d, which also means x_s goes to x_s^d. To address this, we start by defining the tracking error $\epsilon = F_s - F_s^d$. Then, we can express the dynamics of the tracking error as:

$$\dot{\epsilon} = \frac{\partial h}{\partial x_s}\dot{x}_s - Lf(x_s - u_s, u_1, u_2) \tag{3.3}$$

where $\dot{x}_s$ and u_s are the spine state and input, respectively, and can design u_s as a function of the tracking error to achieve the desired control objective. The next design step use the

lipschitez property of $h(x_s - u_s, u_1, u_2)$. Let L be a Lipschitz constant for

$$|h(x_1, u_1, u_2) - h(x_2, u_1, u_2)| \leq L|x_1 - x_2| \tag{3.4}$$

For all $x_1, x_2 \in \mathbb{R}^n$ and $u_1, u_2 \in \mathbb{R}$, use Lipschitz property to define a Lyapunov function candidate as follows:

$$V(\epsilon, x_s) = \frac{1}{2}\|\epsilon\|^2 + \frac{L}{2}\|(x_s - x_s^d)\|^2 \tag{3.5}$$

where $\epsilon = F_s - F_s^d$ and $L > 0$ is the Lipschitz constant of $h(x_s - u_s, u_1, u_2)$ with respect to x_s

To show the stability of the system, we need to show that $\dot{V}(\epsilon, x_s) < 0$, where $\dot{V}$ is the total derivative of V with respect to time. Taking the total derivative of V, we get:

$$\begin{aligned} \dot{V}(\epsilon, x_s) &= \epsilon^T(\frac{\partial h}{\partial x_s})\dot{x}_s - \epsilon^T(\frac{\partial h}{\partial x_s})u_s - L\|(x_s - x_s^d)\|^2 \\ &= \epsilon^T(\frac{\partial h}{\partial x_s})u_s - \epsilon^T(\frac{\partial h}{\partial x_s})(\frac{\partial h}{\partial x_s})^{-1}Lf(x_s - u_s, u_1, u_2) - L\|(x_s - x_s^d)\|^2 \end{aligned} \tag{3.6}$$

where we have used the fact that $\dot{x}_s = (\frac{\partial h}{\partial x_s})^{-1}(F_s - F_s^d)$.

Because we require $\dot{V}(\epsilon, x_s) < 0$ for stability, we design the feedback control law for u_s to satisfy this inequality using a proportional-derivative (PD) controller of the form:

$$u_s = k_p\epsilon - k_d\dot{\epsilon} \tag{3.7}$$

where $k_p, k_d > 0$ are the proportional and derivative gains, respectively. As shown in Equation (3.6), substituting this control law into the expression for $\dot{V}(\epsilon, x_s)$, then get:

$$\dot{V}(\epsilon, x_s) = -\epsilon^T(k_pI + k_d\frac{\partial h}{\partial x_s})(\frac{\partial h}{\partial x_s})^{-1}\epsilon - L\|(x_s - x_s^d)\|^2 \tag{3.8}$$

then the algorithm can guarantee that $\dot{V}(\epsilon, x_s) < 0$ for all $\epsilon \neq 0$

3.3 Simulation Result

To improve the stability of the proposed system, we need to measure the spine state x_s more accurately by implementing a continuous dynamic model. For this purpose, we wrap up muscle-like material on the quadrupedal robot's spine, converting the discrete model into a continuous one.

Now we detect readout value to test the network's performance with a feedback loop, as shown in Figure 3.3. This is quite important for Echo State Network(ESN) learning process, as close loop feedback measured spine state, it will check if the Center of Mass is mapping on the dynamic surface.

After training and testing the data for 1000 time steps, we can determine the Reservoir network's prediction, as shown in Figure 3.4. The result fits the non-linear regression. The purpose to test after training is to test the performance and accuracy of the network's predictions.

Next, the simulation used a "warmup" to initiate the training of the network using a period of time series data before making the prediction output. This process stabilized the network's state, followed by using the trained network to make a prediction[32], as shown in Figure 3.5. The absolute deviation is quite small which the initial "warmup" is necessary for Echo State Network(ESN) learning [30].

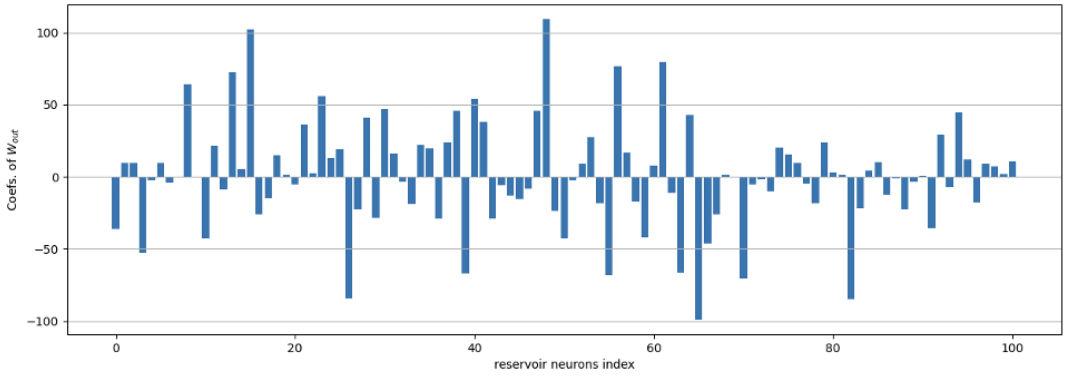

Figure 3.3: Readout index to demonstrate the reservoir computing's performance with a feedback loop

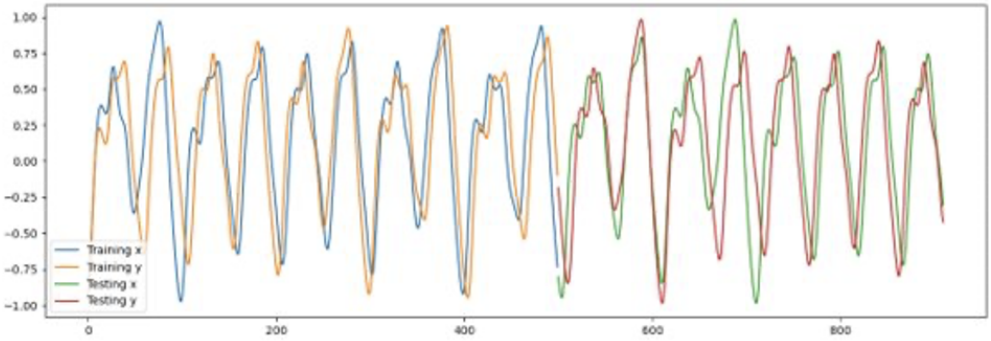

Figure 3.4: First 500 time steps are training data, the test data used to test the performance and accuracy of the network's predictions.

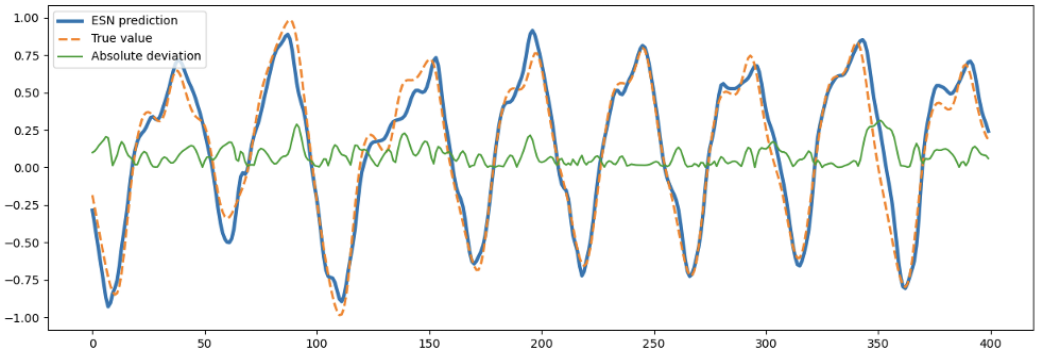

Figure 3.5: Absolute deviation is very small and fit the non-linear regression.

3.4 Conclusion

This chapter has proposed a novel methodology for designing and controlling a quadruped robot with a dynamic spine utilizing Reservoir Computing, hierarchical distributed control mechanisms, and Lyapunov Control algorithm. The developed model can navigate uneven terrain while minimizing energy costs and improving agility. The proposed methodology also overcomes the challenges related to flexible structure control, making it possible to extend the application to soft robotics as well. Additionally, it has proven the stability of the proposed system. The results have significant applications in fields such as search and rescue operations, the military sector, mining, construction, the medical industry, and robotics for autonomous exploration and monitoring, contributing to developing more effective, stable, and adaptable robots. Our future work will focus on implementing the proposed methodology on a physical quadruped robot and extend the reservoir computing to apply other dynamics. This model and methodology will provide valuable insight into the dynamic behavior of legged robots and soft robots and contribute to improving future designs.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

The concepts and methodologies introduced in this thesis are a contribution to Robot State Estimation (RSE) for legged robots. The integration of a learning-based contact estimation framework, combined with the use of multimodal proprioceptive sensory data through a Physics-Informed Neural Network (PINN) and an Unscented Kalman Filter (UKF), enhanced the accuracy and reliability of state estimation. The primary goal of effectively calibrating the Inertial Measurement Unit (IMU) and providing a detailed depiction of the robot's dynamic state was successfully met, overcoming several limitations of traditional vision-based systems.

The application of PINN-UKF mitigates IMU drift by imposing constraints on the loss function through Ordinary Differential Equations (ODEs), thus ensuring that the system remains unaffected by visual impairments and eliminates the need for dedicated contact sensors. The contact estimation component of our system proved adept at identifying contact events across diverse terrains, enhancing proprioceptive odometry and enabling the generation of precise odometric trajectories through a contact-aided invariant Kalman Filter. This led to good estimates of the robot's three-dimensional attitude, velocity, and position, enhancing operational reliability.

Despite these advances, the physical limitations of traditional robotic platforms, particularly in terms of flexibility and impact absorption, remained a significant challenge. Addressing these issues, this thesis introduces an innovative approach by implementing a dynamic spine in a quadruped robot, marking a revolutionary step toward improving

robotic mobility and stability. This modification necessitates reformulating the robot's dynamic model as the spine altered the entire system dynamics during movement. Traditional modeling techniques fell short in accommodating these changes, prompting a shift towards more adaptive and resilient modeling techniques such as Reservoir Computing. This approach effectively managed the complex dynamics that emerged from the spine's interactions during movement, providing a robust framework for real-time adaptive control.

In conclusion, developing a dynamic spine coupled with the application of Reservoir Computing addressed the inherent limitations observed in traditional robotic platforms and set a new benchmark in robotic design and functionality. These innovations lay a robust foundation for future research and development. The methodologies and technologies developed in this thesis not only enhance the operational capabilities of robotic systems but also open new avenues for their application in more dynamic and unstructured environments, promising a future where robots are more adaptable, resilient, and capable of handling the complexities of the real world. We believe that the ongoing refinement of these technologies and the exploration of new applications will continue to drive the evolution of robotics, making an indelible impact on both theoretical and practical aspects of robotics.

4.2 Future Work

Building on the results of this thesis, future research will aim to extend and refine the methodologies developed for enhanced robot state estimation and dynamic spine integration. Key initiatives include enhancing virtual sensor technologies within the Physics-Informed Neural Network (PINN) framework to improve sensor fusion accuracy and refining physical models to capture complex real-world dynamics more precisely. This effort will encompass expanding state estimation methodologies to complex, unstructured envi-

ronments such as underwater or extraterrestrial locations where proprioceptive feedback is crucial. Additionally, incorporating diverse sensory modalities such as acoustic and thermal sensors will make the state estimation process robust against environmental challenges.

For the dynamic spine implementation described in the second chapter, future work will involve the physical prototype development and rigorous field testing of the dynamic spine on quadruped robots, exploring its impact on robotic mobility and stability across various terrains. This includes expanding the application of Reservoir Computing to manage more complex dynamic models and developing sophisticated adaptive control algorithms that adjust in real-time to changes in robot posture and terrain interaction. Furthermore, applying dynamic spine concepts and Reservoir Computing to soft robotics can pioneer new functionalities in areas like minimally invasive surgery and disaster recovery. Collaborative cross-disciplinary studies in biomechanics and material science will be crucial for enhancing the design and functionality of dynamic spines, potentially leading to innovations in robotic materials that mimic biological structures.

These works will build upon the robust foundation laid by this thesis, pushing the boundaries of robotic capabilities and opening up new avenues for research and development in adaptive and resilient robotic systems for challenging environments.

BIBLIOGRAPHY

[1] D. Selmanaj, M. Corno, G. Panzani, and S.M. Savaresi. Vehicle sideslip estimation: A kinematic based approach. *Control Engineering Practice*, 67:1–12, 2017.

[2] Y.-W. Liao and F. Borrelli. An adaptive approach to real-time estimation of vehicle sideslip, road bank angles, and sensor bias. *IEEE Transactions on Vehicular Technology*, 68(8):7443–7454, 2019.

[3] Z. Liu, Y. Cai, H. Wang, L. Chen, H. Gao, Y. Jia, and Y. Li. Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions. *IEEE Transactions on Intelligent Transportation Systems*, 23:6640–6653, 2021.

[4] X. Xia, L. Xiong, Y. Lu, L. Gao, and Z. Yu. Vehicle sideslip angle estimation by fusing inertial measurement unit and global navigation satellite system with heading alignment. *Mechanical Systems and Signal Processing*, 150:107290, 2021.

[5] W. Li, Z. Xie, P. K. Wong, Y. Hu, G. Guo, and J. Zhao. Event-triggered asynchronous fuzzy filtering for vehicle sideslip angle estimation with data quantization and dropouts. *IEEE Transactions on Fuzzy Systems*, 30(8):2822–2836, 2022.

[6] X. Ding, Z. Wang, and L. Zhang. Event-triggered vehicle sideslip angle estimation based on low-cost sensors. *IEEE Transactions on Industrial Informatics*, 18(7):4466–4476, 2022.

[7] A. Wischnewski, T. Stahl, J. Betz, and B. Lohmann. Vehicle dynamics state estimation and localization for high performance race cars. *IFAC-PapersOnLine*, 52(8):154–161, 2019.

[8] V. Mazzilli, D. Ivone, S. De Pinto, L. Pascali, M. Contrino, G. Tarquinio, P. Gruber, and A. Sorniotti. On the benefit of smart tyre technology on vehicle state estimation. *Vehicle System Dynamics*, pages 1–26, 2021.

[9] A. Bertipaglia, M. Alirezaei, R. Happee, and B. Shyrokau. Model-based vs data-driven estimation of vehicle sideslip angle and benefits of tyre force measurements. In *Proceedings of the AVEC Int. Symposium on Advanced Vehicle Control*, 2022.

[10] O. Galluppi, M. Corno, and S.M. Savaresi. Mixed-kinematic body sideslip angle estimator for high performance cars. In *European Control Conference (ECC)*, pages 941–946, IEEE, 2018.

[11] E. Villano, B. Lenzo, and A. Sakhnevych. Cross-combined ukf for vehicle sideslip angle estimation with a modified dugoff tire model: design and experimental results. *Meccanica*, 56(11):2653–2668, 2021.

[12] E. Hashemi, M. Pirani, A. Khajepour, A. Kasaiezadeh, S.-K. Chen, and B. Litkouhi. Corner-based estimation of tire forces and vehicle velocities robust to road conditions. *Control Engineering Practice*, 61:28–40, 2017.

[13] J. Guerrero-Ibáñez, S. Zeadally, and J. Contreras-Castillo. Sensor technologies for intelligent transportation systems. *Sensors*, 18:1212, 2018.

[14] M. Kissai. *Optimal Coordination of Chassis Systems for Vehicle Motion Control*. PhD thesis, Université Paris-Saclay (ComUE), Orsay, France, 2019.

[15] W. Liu, X. Xia, L. Xiong, Y. Lu, L. Gao, and Z. Yu. Automated vehicle sideslip angle estimation considering signal measurement characteristic. *IEEE Sensors Journal*, 21(19):21675–21687, 2021.

[16] L. Yuan, Y.Q. Ni, X.Y. Deng, and S. Hao. A-pinn: Auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations. *Journal of Computational Physics*, 462:111260, 2022.

[17] A. Barrau and S. Bonnabel. Intrinsic filtering on lie groups with applications to attitude estimation. *IEEE Transactions on Automatic Control*, 60:436–449, 2014.

[18] M. Brossard, A. Barrau, and S. Bonnabel. A code for unscented kalman filtering on manifolds (ukf-m). In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5701–5708, Paris, France, 2020.

[19] T.D. Barfoot and P.T. Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30:679–693, 2014.

[20] M. Brossard, S. Bonnabel, and J.P. Condomines. Unscented kalman filtering on lie groups. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2485–2491, Vancouver, BC, Canada, 2017.

[21] S.J. Julier and J.K. Uhlmann. New extension of the kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition VI, SPIE*, 3068:182–193, 1997.

[22] Dong-Hyun Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *Cornell University - arXiv*, Sep 2019.

[23] Grant Gibson, Oluwami Dosunmu-Ogunbi, Yukai Gong, and Jessy Grizzle. Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[24] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems XVII*, 2021.

[25] Paolo Arena, Fabio Di Pietro, Alessia Li Noce, Salvatore Taffara, and Luca Patane. Assessment of navigation capabilities of mini cheetah robot for monitoring of landslide terrains. *2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI)*, 2021.

[26] MLIK SALWA. Next generation reservoir computing. 2021.

[27] Qian Zhao, Kohei Nakajima, Hidenobu Sumioka, Helmut Hauser, and Rolf Pfeifer. Spine dynamics as a computational resource in spine-driven quadruped locomotion. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[28] Satomi Hanasaki, Yuichi Tazaki, Hikaru Nagano, and Yasuyoshi Yokokohji. Running trajectory generation including gait transition between walking based on the time-varying linear inverted pendulum mode. *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022.

[29] Amir Iqbal, Sushant Veer, and Yan Gu. Drs-lip: Linear inverted pendulum model for legged locomotion on dynamic rigid surfaces. 2022.

[30] Mario Calandra, Luca Patanè, Tao Sun, Paolo Arena, and Poramate Manoonpong. Echo state networks for estimating exteroceptive conditions from proprioceptive states in quadruped robots. *Frontiers in Neurorobotics*, 15, 2021.

[31] Wen-Loong Ma, Noel Csomay-Shanklin, Shishir Kolathaya, Kaveh Akbari Hamed, and Aaron D. Ames. Coupled control lyapunov functions for interconnected systems, with application to quadrupedal locomotion. *IEEE Robotics and Automation Letters*, 6(2):3761–3768, 2021.

[32] Andrew P. Sabelhaus, Huajing Zhao, Edward L. Zhu, Adrian K. Agogino, and Alice M. Agogino. Model-predictive control with inverse statics optimization for tensegrity spine robots. *IEEE Transactions on Control Systems Technology*, 29(1):263–277, 2021.