Dynamic Motion State Estimation and Control via Sim-to-Real Transfer

Dynamische Bewegungszustandsschätzung und -steuerung mittels Sim-to-Real-Transfer

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg
zur
Erlangung des Doktorgrades Dr.-Ing.
vorgelegt von

Simon Bachhuber aus
Rotthalmünster

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung: TBD
Gutachter/in: Prof. Dr.-Ing. Thomas Seel
Prof. Dr. Andreas M. Kist

Dynamic Motion State Estimation and Control via Sim-to-Real Transfer

Simon Bachhuber

simon.bachhuber@fau.de
www.aibe.tf.fau.de
Chair for Intelligent Sensorimotor Systems
Department of Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg
91052 Erlangen
Germany

To my parents, who let me find my own path, never questioning my decisions, and to all open-source contributors—my work stands on your shoulders.

Abstract

Agile motion plays a vital role in biological and technological domains, contributing to animal survival, human well-being, and the practical viability of robotic systems. Motion quantification is critical for passive observation, allowing for analysis and informed conclusions, and for active intervention and control. For example, motion quantification enables tracking a patient's rehabilitation progress while enabling exoskeletons to actively support the patient's gait.

Both passive motion analysis and active motion control demand considerable engineering expertise to identify, select, and configure suitable methods for effectively utilizing the available sensors and actuators. For instance, tracking a patient's movement with wearable inertial sensors is a time-intensive task that requires careful method selection. It often involves precise system modeling and accurate calibration of both sensors and their attachment to the body. Data-driven methods can reduce the need for system modeling and calibration. However, they typically require extensive amounts of real-world data to avoid trading off generalizability and robustness. For example, the viscoelastic nature of soft robots makes each unique, limiting the transferability and robustness of data-driven solutions across different robots.

In this thesis, we develop a unified approach that is applicable to both motion analysis and motion control problems, and that addresses the limitations of current model-based and data-driven solutions by minimizing the need for 1) problem-specific prior knowledge or modeling efforts, 2) repeated manual adaptations, and 3) reliance on real-world data. The developed approach leverages RNNs (Recurrent Neural Networks) trained in simulation environments to perform well in real-world scenarios through simulation-to-reality (sim-to-real) transfer. By incorporating extensive domain randomizations during training, the trained RNN is robust to the variability and uncertainties present in real-world applications. This process enables the trained RNN to zero-shot generalize without additional tuning, offering *plug-and-play*, real-time-capable solutions while requiring far less real-world data.

We demonstrate the approach's effectiveness by applying it to the state estimation task of IMT (Inertial Motion Tracking) and to the motion control task of RT (Reference Tracking). The resulting solutions enable: 1) *observability* analysis of IMT problems that scales both simulated data and network size to enable a comprehensive assessment of the ability to estimate the motion states from the available sensor data; 2) for the first time, real-world IMT that overcomes four key challenges simultaneously, enabling *magnetometer-free*, *sparse* IMT with *sensor-to-segment calibration* capabilities and motion artifact reduction; 3) a universal solution for IMT eliminating the need to select specific methods by training a single, problem-agnostic RNN based on a novel neural network architecture that generalizes even over the number of segments in the kinematic chain; 4) a data-efficient learning control method that enables RT in unknown nonlinear dynamics without expert knowledge, and that is validated on a diverse set of simulated dynamics; 5) a soft robot arm that quickly learns to perform agile motions while effectively rejecting external disturbances.

On the one hand, these advancements make IMT less restrictive and simultaneously more accessible across established domains, unlocking new applications for users and fields previously beyond reach. On the other hand, they enable practical, user-friendly motion control of soft robots with self-calibrating capabilities. Moreover, the developed concepts may transfer to other domains that involve a reduced sensor count, e.g., in environmental monitoring tasks.The unified approach, that combines RNNs and sim-to-real transfer, can contribute to fundamental changes to how motion analysis and control problems are approached, paving

the way for innovation in fields ranging from robotics to biomechanics.

Zusammenfassung

Dies ist der deutsche Abstract...

Contents

Glossary

heading corresponds to the yaw angle, representing the horizontal orientation relative to a reference direction, often true north. 20, 21, 23, 51

inclination the orientation of an object relative to the horizontal plane; it indicates how much an object is tilted. 20

magnetometer-free without the usage of the magnetometer; in the context of inertial motion tracking also referred to as 6D (3D accelerometer and 3D gyroscope) in contrast to 9D (+ 3D magnetometer). 20, 21, 23, 24, 39, 41, 48, 49, 51, 58–60

motion artifacts refers to the concept that sensors measure different values if the system is in motion, e.g., if the sensors are nonrigidly-attached, then the skin allows for relative motion between sensor and skeleton. 21, 22, 24, 25, 41, 49, 51, 58, 61

observability the ability to infer the complete internal state of a system from its external outputs over time, see Definition B.2. 18, 19, 24, 31, 32, 35, 41, 47, 48, 57, 59, 76

plug-and-play solutions that do not require time investment or expertise by the user, e.g., in the form of calibration or tuning. v, 3, 9, 11, 13, 14, 17, 18, 23, 25, 32–34, 36, 39, 40, 42, 43, 48, 53, 55, 57, 58, 61, 67

pluripotent a pluripotent method is problem unspecific, i.e., it can be applied to a broad range of problems. x, 25, 34, 43, 48, 58

sensor-to-segment calibration identifying the joint positions and joint axes' directions in local sensor coordinates. 21–23, 25, 58, 61

sparse short for sparse sensing; with a reduced sensor count; in the context of inertial motion tracking this refers to having less than one sensor per segment. v, 21, 23, 31–33, 35, 41, 48, 49, 51, 53, 57–61

Acronyms

Articulated Rigid Body System (ARBS) a system that consists of interconnected rigid bodies joined by movable joints, allowing complex, coordinated motion across multiple segments. 39, 42, 43, 55

Artificial Neural Network (ANN) a type of artificial intelligence model that mimics the human brain to process data and create patterns for decision-making. 9–11, 13, 15, 16, 18, 24, 25, 28, 32–34, 36, 38, 41–43, 48, 49, 53, 56, 58–63, 67–69, 71, 72

Automatic Neural Ordinary Differential Equation Control (ANODEC) a learning control method proposed in Bachhuber et al. (2023b). x, 32, 34, 35, 45, 46, 51–56, 59, 63

Connectivity Graph (CG) an undirected graph where the nodes represent the bodies that constitute an articulated rigid-body system and the edges represent its joints. 43, 44

Degree of Freedom (DOF) an independent way a system can move or be positioned. 4, 18, 39, 50, 53, 60, 75

Gated Recurrent Unit (GRU) a type of recurrent neural network proposed in Cho et al. (2014). 41, 43, 44, 55, 71, 73

Inertial Measurement Unit (IMU) a sensor that typically measures 3D angular velocity (gyrscope), 3D acceleration (acceleterometer), and 3D magnetic field density (magnetometer). 12, 16–25, 31, 33, 35, 39, 41–43, 48–51, 56–62, 75

Inertial Motion Tracking (IMT) technology used to track the movement and pose of an articulated system using inertial sensors. x, 12–15, 19, 21, 23–25, 31–36, 39–43, 47–51, 53, 55–62

Inertial Orientation Estimation (IOE) a technique to estimate the orientation of a single inertial sensor. 20, 21, 24

Iterative Learning Control (ILC) a control method where a system improves its performance over repeated tasks by learning from past errors to refine future actions. 27, 28, 55

Kinematic Chain (KC) a series of rigid bodies connected by joints, enabling controlled motion in mechanisms like robotic arms or skeletal structures. 17, 19, 21, 23–25, 31, 33–36, 41, 43, 44, 47–49, 51, 57–59

Linear Quadratic Regulator (LQR) an optimal control method that minimizes a cost function for linear systems. 7, 9, 27, 28

Long Short Term Memory (LSTM) a type of recurrent neural network proposed in Hochreiter and Schmidhuber (1997). 55, 71, 73

Machine Learning (ML) a field where algorithms learn patterns from data to make predictions or decisions without explicit programming. 10, 15, 20, 21, 24, 34, 48, 52

Markov Decision Process (MDP) a mathematical framework for modeling decision-making in scenarios where outcomes are partly random and partly under the control of a decision-maker. 27, 37, 38

Mean Absolute Error (MAE) a scalar error metric. 33, 48–51, 58

Mean Squared Error (MSE) a scalar error metric. 41, 72

Model Predictive Control (MPC) a control technique that optimizes future actions by predicting system behavior over a finite horizon and updating decisions at each step based on the latest information. 8, 27, 28

Multi Layer Perceptron (MLP) a type of artificial neural network composed of multiple layers of interconnected neurons, often a building block for larger networks. 44, 46

Neural Ordinary Differential Equation (NODE) neural networks that model continuous-time processes by learning dynamics from data, without requiring physical equations. 32, 45, 46, 55, 63, 68, 71, 72

One-Dimensional (1D) a space constrained to a single axis or coordinate. 22, 23, 25, 49, 50

Optical Motion Capture (OMC) technology that tracks movement using cameras and markers. x, 19, 27, 28, 49, 61

Optimal Control (OC) a mathematical approach to determine a control policy that minimizes or maximizes a performance criterion for a dynamic system. 27, 28

Partially Observable Markov Decision Process (POMDP) an extension of Markov decision processes where the agent cannot directly observe the underlying state of the environment. 27, 37–43, 45, 67

Pneumatic Artificial Muscle (PAM) a flexible actuator that contracts when pressurized with air, mimicking the function of natural muscles. 26, 52, 53, 59

Probabilistic Inference for Learning Control (PILCO) a learning control method proposed in Deisenroth and Rasmussen (2011). 27

Proportional, Integral, Differential (PID) a type of feedback controller. 5, 6, 20, 52, 54, 59, 60

Random Chain Motion Generator (RCMG) a simulator of kinematic chains performing random motion proposed in Bachhuber et al. (2022). 39, 41, 42, 55, 57–59

Recurrent Inertial Graph-based Estimator (RING) a network of message-passing RNNs that estimates state information from sensor data in an end-to-end, decentralized manner; makes use of the structure of an articulated system. x, 33–36, 41–44, 48–51, 53, 55, 56, 58–62

Recurrent Neural Network (RNN) a class of artificial neural networks designed for sequential data processing and learning dependencies over time. v, xvii, 3, 14, 15, 17, 18, 21, 28, 31, 32, 34, 35, 38–41, 43, 47, 48, 55–57, 59, 67–73

Recurrent Neural Network-based Observer (RNNO) a recurrent neural network that estimates state information from sensor data in an end-to-end, centralized manner; does not make use of the structure of an articulated system. 31–33, 35, 41, 47–49, 53, 55–59

Reference Tracking (RT) the process of following a reference signal with minimal deviation. 5, 12–15, 25–28, 32, 34, 35, 40, 44–46, 51, 52, 54, 55, 57–59, 63

Reinforcement Learning (RL) a type of machine learning where an agent learns to make decisions by receiving rewards or penalties for its actions, aiming to maximize cumulative rewards over time.. 9, 11, 16, 18, 26–28, 38, 45

Robust IMU-based Attitude Neural Network (RIANN) an inertial orientation estimation method proposed in Weber et al. (2021). 21, 24

Root Mean Squared Error (RMSE) a scalar error metric. 32, 51–54

Six-Dimensional (6D) a space constrained to six axes or coordinates, in the context of inertial motion tracking refers to magnetometer-free measurements, consisting only of gyroscope and accelerometer. 18, 25, 42

Soft Robot (SR) a type of robot made from flexible, deformable materials, enabling safe, adaptive interactions with unpredictable environments and delicate objects. 12, 25, 26, 28, 29, 34, 35, 52, 55, 56, 59

State-of-the-Art (SOTA) the most advanced and high-performing techniques, technologies, or knowledge available in a particular field at a given time. 20, 49, 50, 58, 73

Three-Dimensional (3D) a space constrained to three axes or coordinates. 12, 18, 19, 21, 23, 25, 39, 42, 49

Two-Dimensional (2D) a space constrained to two axes or coordinates. 23, 25, 49

Versatile Quaternion-based inertial orientation estimation Filter (VQF) an inertial orientation estimation method proposed in Laidig and Seel (2023). 20, 21

Part I

Background

1

Introduction

"Research is about moving forward systematically from adequate known detail into the unknown."
– Vannevar Bush, Head of the U.S. Office of Research during World War II

This thesis develops and validates a broadly applicable RNN-based approach that uses simulation data to efficiently solve the dual problem of motion analysis and control of a system in dynamic motion, and that yields easy-to-use, plug-and-play solutions in the form of trained RNNs. The thesis begins with this introductory chapter, which motivates the research field of motion analysis and control, then introduces classical, model-based, and data-driven solutions to the problem. The chapter finishes by defining the aim of the thesis and outlining its content.

1.1 Motivation

Motion is essential. The world is filled with humans, animals, and robots that perform dynamic, agile motions (see Figure 1.1): Animals that hunt prey, humans who run and are physically active, and robots that serve our needs. Motion surrounds us everywhere. Animals rely on dynamic motion because of evolutionary adaptation. If the cheetah can't outrun the deer, it will starve. Societal demands and peer-competitive advantages favor the dynamic human. Humans work out and track their progress to stay healthy and attractive. Similarly, technological advances enable robots to become more time-efficient through fast motion. Agile motion is ubiquitous, and many technical or biological entities rely on fast, dynamic motions.

Fundamentally, there are two different tasks that involve dynamic motions: motion analysis and motion control. On the one hand, we want to analyze and track the dynamic motion itself to draw conclusions. For example, a marathon runner might want to track their training, or a doctor might want to quantify a patient's rehabilitation progress. We are passively observing the motion. On the other hand, we want to enable humans or robots to perform dynamic motions so that they can efficiently achieve real-world tasks. For example, we want to use functional electrical stimulation to stimulate muscles so that disabled humans can move again and regain their abilities, or we want to control the motor currents of a quadrotor to fly high-speed maneuvers. Here, we are actively interfering to manipulate the motion.

Figure 1.1: *Examples of animals, humans, and robotic systems in dynamic motion: A cheetah in full sprint, a woman while running, and a quadrotor flying. All three images are license-free.*

To achieve both motion analysis and motion control, some form of sensing and actuation is always required. Sensors capture the movement, typically in specific coordinates or measurable quantities, while actuators initiate and control the movement. Figure 1.2 illustrates this duality. Motion control often builds on motion analysis. It involves either a learned signal pattern triggering movement or the use of real-time sensor data to make quick adjustments to how the movement should unfold.

Let us first delve into sensing. **Various types of sensors capture movement**, typically via optical or inertial measurements, resulting in a vast amount of detail to explore. Each set of sensors has strengths and weaknesses; however, let's streamline our discussion toward a key concept: motion refers to rigid bodies changing their position and velocity relative to a reference frame. Mathematically, this type of movement can be formalized by specifying the positions of all DOFs (Degree of Freedoms), i.e., the positions of all translational and rotational joints that connect all bodies. These generalized positions, together with their velocities, form the so-called state of the system, a fundamental property that fully describes it. **However, there is a challenge: while the full motion can be described using these values, the measurable quantities (optical and inertial data) provided by the sensors are different from the latter key motion states. Thus, the motion states must be determined from the measured data. This can be achieved with the use of state estimation**, which involves algorithms that rely on sensor measurements to determine the state information of a given system. An algorithm for state estimation is called an *observer*, and the Kalman filter is one well-known example of an observer (Kalman, 1960b). Applications of observers are highly diverse. They are used for human motion analysis (Halilaj et al., 2018; López-Nava and Muñoz-Meléndez, 2016), for foundational research in biomechanics (Wong et al., 2015), for healthcare and rehabilitation purposes (Buke et al., 2015), for autonomous piloting (Macario Barros et al., 2022), and for applied robotics (Barfoot, 2017; Novak and Riener, 2015).

Similarly, there exist various forms of actuators and control strategies for creating the functional dynamic motions required for applications such as locomotion–the ability to move from one place to another–and other complex tasks. For these motions to be executed, the actuators must apply the appropriate forces, torques, voltages, or muscle stimuli at the correct moment. The computation of actuator inputs often involves two steps. First, a desirable

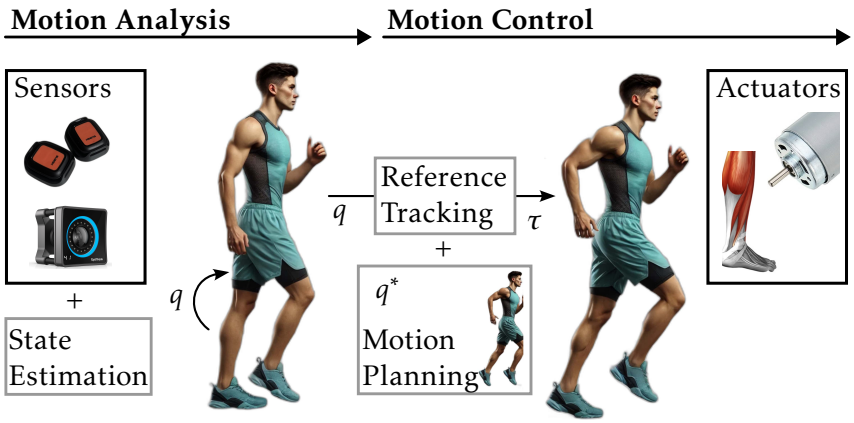


Figure 1.2: *Motion analysis uses state estimation algorithms to capture the motion, e.g., as joint angles q, from sensor data. Motion control then uses this information to control the actuators to track a planned reference motion closely.*

motion that achieves the higher level task must be identified and translated into a trajectory of reference states (e.g., joint angles of a robot arm) via motion planning. Second, the planned reference motion must be realized. **Thus, motion control, i.e., achieving a desired motion, typically involves algorithms from the fields of motion planning and RT**. RT algorithms compute the actuation (e.g., motor torques, muscle stimuli) such that the state tracks the desired reference, and the system performs the motion. The PID (Proportional, Integral, Differential) controller is one example of a, relatively simple, RT algorithm. RT algorithms are typically either feedforward or feedback controllers. Feedforward control usually involves computing the required trajectory of actuation inputs to achieve the planned reference motion based on a model of the system, without making real-time adjustments. In contrast, a feedback controller adjusts the inputs in real-time based on the deviation between reference and actual motion. The actual motion is estimated by motion analysis. Consequently, the two tasks of motion state estimation and motion control work together, influence each other, and operate in collaboration. They are dual. State estimation involves reconstructing the internal state of a system from sensor data. At the same time, control typically uses state information to influence its future behavior. Together, they form a feedback loop where accurate state estimation enables precise control. The application areas for motion control are immensely diverse. Examples are search-and-recuse drone systems (Hanover et al., 2024), lower extremity exoskeletons (Tijjani et al., 2022), and robotic surgery (Le et al., 2016).

Therefore, the task of motion analysis and motion control depends on high-performance control theory components (state estimation and RT algorithms) as well as motion planning. Simultaneously, the breadth of applications shows the high relevance of analyzing and controlling the motion of complex, real-world systems. **Unfortunately, conventional control theory methods demand strong participation from control engineers. As a result, motion analysis and control of real-world systems typically requires extensive amounts of expert labor and knowledge.**

1.2 Model-based and Data-driven Solutions

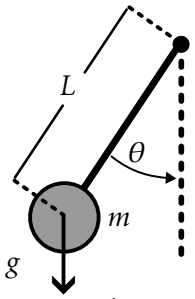
"All models are wrong, but some are useful."
– George Box, statistician and professor

Control theory studies the analysis and regulation of dynamical systems, aiming to understand

and influence their behavior over time. Conceptually, animals, humans, and robots can be modeled as dynamical systems, and the cheetah, the woman, and the quadrotor of Figure 1.1 are examples thereof. A dynamical system is a system whose state evolves according to a set of rules or equations, often influenced by both internal (the metabolic rate in animals or humans, or the battery level in robots) and external factors (the environment temperature, surface terrain conditions, or control inputs such as voltages). The state of a dynamical system refers to a set of variables that fully describe the system's current condition and allow us to predict its future behavior, given the system's dynamics, the set of rules or equations. For example, the movement of a car on a road can be modeled as a dynamical system, where the car's speed and direction (the states) evolve based on the driver's input and the road conditions (external factors). A second example of a dynamical system is introduced in more detail in Example 1.1.

Example 1.1

The pendulum consists of a point-like mass m that is attached to a massless and in-extensible string of length L and swings under the influence of gravity g without any friction or air resistance. The state of a dynamical system at time t is typically denoted by $\boldsymbol{x}_t$ and is here given by $\boldsymbol{x}_t = \begin{bmatrix} \theta_t \\ \omega_t \end{bmatrix}$ where θ_t is the angle of the pendulum from the vertical at time t. Similarly, ω_t is the angular velocity of the pendulum.



The dynamics of the system determine how the state variables change in response to internal factors, and it describes the behavior of the system as it moves from one state (from $\boldsymbol{x}_t$) to the next (to $\boldsymbol{x}_{t+1}$). Here, the dynamics of the pendulum can be easily derived in analytical form and are given in discrete time by (using explicit Euler and a step size of Δt)

$$\theta_{t+1} = \theta_t + \omega_t \Delta t \text{ and } \omega_{t+1} = \omega_t - \frac{g}{L} \sin(\theta_t)\, \Delta t. \tag{1.1}$$

The dynamics of the pendulum are nonlinear due to the sine function. Finally, note that this system has no actuation or control input. Such a system is called an autonomous system.

Loosely speaking, the control theory methods may be divided into the categories of classical, model-based, and data-driven control. Classical control refers to a traditional approach to control systems, primarily based on transfer functions and frequency domain methods. It focuses on single-input, single-output systems and relies on linear system theory, simplifying controllers' design and analysis. Control engineers often use tools like Bode plots, Nyquist plots, and root locus diagrams to analyze system stability and performance and to design controllers within this framework. A notable example is the widely used PID controller, a cornerstone of classical control, valued for its simplicity, effectiveness, and ease of implementation. However, classical control methods have several limitations that have led to the development of model-based control approaches. Classical control is primarily designed for single-input, single-output systems, assumes linear, time-invariant behavior, and lacks an inherent mechanism for achieving optimality. In contrast, real-world applications frequently involve multi-input, multi-output systems that exhibit time-variant and nonlinear behavior, where a systematic approach to achieve the best possible performance based on specific criteria becomes essential.

Modern model-based control relies on control-engineer-designed mathematical models, primarily in the form of state-space models, that approximate the dynamics of real-world systems. State-space models are first-order differential equations, and Example 1.1 can be interpreted as a model for a real-world pendulum in state-space form.

Example 1.2

Consider the woman running in Figure 1.1. Biomechanics is the field that develops models of humans. The human skeleton itself is largely a composition of connected rigid bodies (bones connected by joints), but with joints that are actuated by nonrigid muscle tendons that apply torques through contraction upon nerve stimulation (see Figure 1.3). A biomechanical model simulates the motion of the skeleton as the muscle nerve stimuli are varied. Model-based control methods, and in particular optimal control methods, can be used to reconstruct the joint torques in, e.g., gait motion from wearable inertial sensors. However, this process crucially depends on accurate biomechanical models. The most established human biomechanical model is Rajagopal et al. (2016), which models humans using 37 degrees of freedom and 80 muscle tendons. Assuming joint angles and velocities, and muscle activation and excitation as state variables, then the system's state x has $74 + 160 = 234$ dimensions. The system input vector u consists of the 80 muscle stimuli values (typically normalized). Depending on the available sensors, the system output vector y could be, e.g., the joint angles or angular velocities.

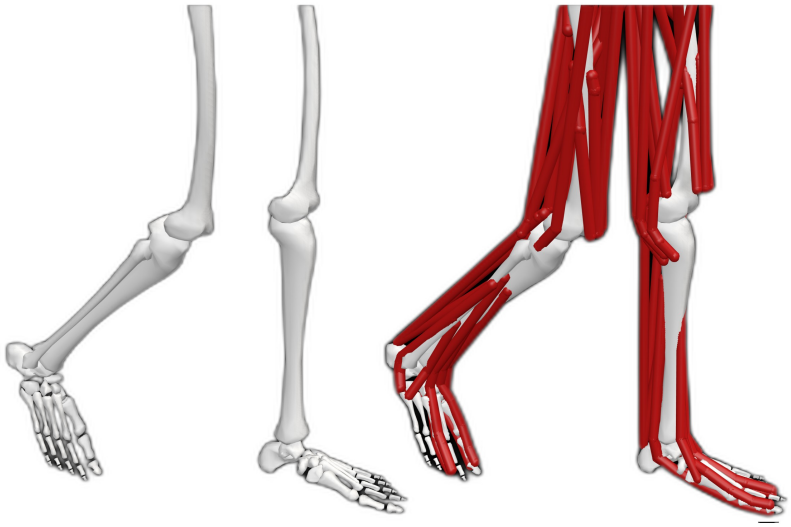


Figure 1.3: *Biomechanical model of the lower body with and without tendons. Developing biomechanical models requires large amounts of expert knowledge and time investment. Image is owned by Simon Bachhuber*[1]*.*

Within model-based control, the subfield of optimal control uses system models to derive a control law that minimizes a specific cost function, such as the deviation between the reference and actual motion. A well-known example of optimal control is the LQR (Linear Quadratic Regulator) (Kalman, 1960a). Other fields, such as robust control and H-infinity control, utilize the system model to design controllers that maintain performance and stability, even in the

[1] Figure 1.3 was created by S.B. using the OpenSim software (Delp et al., 2007) which is publicly available under the Apache License, Version 2.0.

presence of model uncertainties and external disturbances. MPC (Model Predictive Control) is another model-based control method that optimizes a cost criterion by linearizing the system model around the current observed state in real-time. This real-time optimization enables MPC to predict future system behavior and adjust control inputs accordingly, although it can be computationally demanding, particularly for complex or nonlinear systems. **Compared to classical control, where the control engineer directly tunes the controllers, model-based control shifts the focus toward accurately modeling the system.** Typically, the engineer models the system and performs system identification to identify the model's parameters as accurately as possible. **However, this task is non-trivial, time-consuming, and can be especially challenging in real-world, nonlinear, and time-varying systems.** For example, consider the biomechanical model in Figure 1.3. This model is complex due to its dependence on tendon attachment points, nonlinear muscle activations (e.g., Hill model), and the fact that it is highly individualized, making it an approximation that never fully captures the nuances of each person's unique biomechanics. In addition to the large manual efforts, the effectiveness of model-based control is heavily dependent on the accuracy of the system model. As an example of this, consider Example 1.3, as the initial pendulum angle θ_0 increases, model inaccuracies are introduced, and consequently, tracking accuracy of the model-based Kalman filter decreases.

Example 1.3

The Kalman filter is a model-based state estimation algorithm that provides optimal estimates of a system's state by combining noisy measurements with a model of the dynamics and the measurement process. Consider the pendulum from Example 1.1, and let's assume that only the Cartesian x-position of the pendulum is measured. The Kalman filter requires a linear prediction model in the form $\boldsymbol{x}_{t+1} = \mathbf{A}\boldsymbol{x}_t$ and measurement model $\boldsymbol{y}_t = \mathbf{C}\boldsymbol{x}_t$. However, the pendulum dynamics in (1.1) contain a nonlinear sine term. We can linearize the dynamics around $\theta = 0$ to obtain $\mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{L}\Delta t & 1 \end{bmatrix}$ and as measurement model $\mathbf{C} = \begin{bmatrix} L & 0 \end{bmatrix}$. Now, we can use a Kalman filter to track both state variables from the Cartesian x-position only, and we plot the convergence for three initial values for θ.

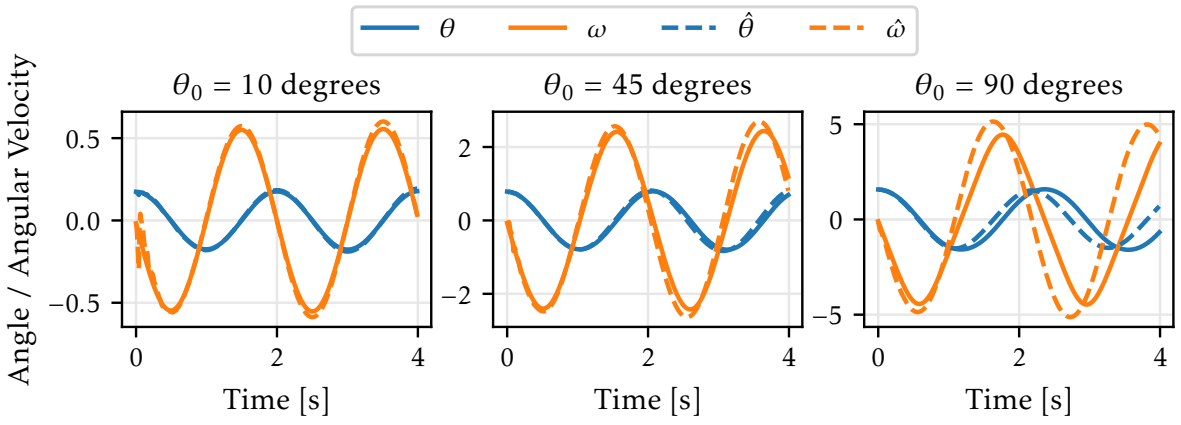


Figure 1.4: A Kalman filter is used to track the state (θ and ω, and $\hat{\theta}$ and $\hat{\omega}$ is the state estimate) of a pendulum from its Cartesian x-position. As the initial pendulum angle θ_0 increases, the linearization of the nonlinear dynamics breaks down, and the tracking accuracy of the observer decreases.

Data-driven control methods utilize data to design an observer or controller directly. Compared to classical and model-based control methods, data-driven control offers high potential for improvement in key areas:

1. Adaptability to Complex Systems: When an accurate model is unavailable, empirical data can always be recorded and analyzed. For example, in the model-based paradigm, personalizing biomechanical models for individual patients is often impractical. In contrast, data-driven approaches can directly utilize data from the specific patient with minimal additional effort, making them more adaptable to personalized applications.

2. Efficiency and Transferability: Data-driven control is typically less time-consuming and requires less expert knowledge, as the control engineer operates at a higher level, focusing on parameterizing methods that automatically learn from data. This higher-level approach makes the acquired knowledge more transferable across different tasks, enhancing the efficiency of the expert.

3. Flexibility Beyond Expert Knowledge: Data-driven methods are not limited by expert knowledge or pre-existing model assumptions. For instance, as universal function approximators, ANNs (Artificial Neural Networks) can identify and learn any statistically significant patterns within the data.

Current data-driven methods for state estimation include techniques such as Gaussian Processes or ANNs that can infer system states from sensor data. For control, RL (Reinforcement Learning) algorithms are commonly employed, allowing them to optimize control policies directly through interactions with the environment.

However, while data-driven methods offer key advantages concerning practicality and personalization, they come with significant drawbacks. They often require large amounts of training data to function effectively, and the solutions they generate can be highly specific to the task at hand, lacking generalizability and flexibility for other applications. Additionally, they frequently do not offer plug-and-play capability, making it difficult to adapt them to new contexts without extensive re-training. They also tend to be poor in robustness to variations and uncertainties in real-world conditions. Moreover, the computational demands, particularly during the training phase, can be prohibitively high, making them resource-intensive and time-consuming to implement.

In addition, these challenges are aggravated due to the requirement of providing methods that work well despite agile, dynamic motion (see Figure 1.5). Dynamic motion poses additional challenges such as:

1. Limitations of Linear Approximations: Many state estimation algorithms (such as Kalman filters, see Example 1.1) and control algorithms (such as LQR) rely on linearization of nonlinear dynamics. However, as the system state $x(t)$ changes rapidly, the nonlinear terms can no longer be ignored (see Appendix B).

2. Limitations of Sensor Sampling Rates: Dynamic motion contains higher frequency components in the state trajectory. According to the Nyquist-Shannon sampling theorem, to accurately capture frequency components of frequency h, a sampling rate of at least $2h$ is

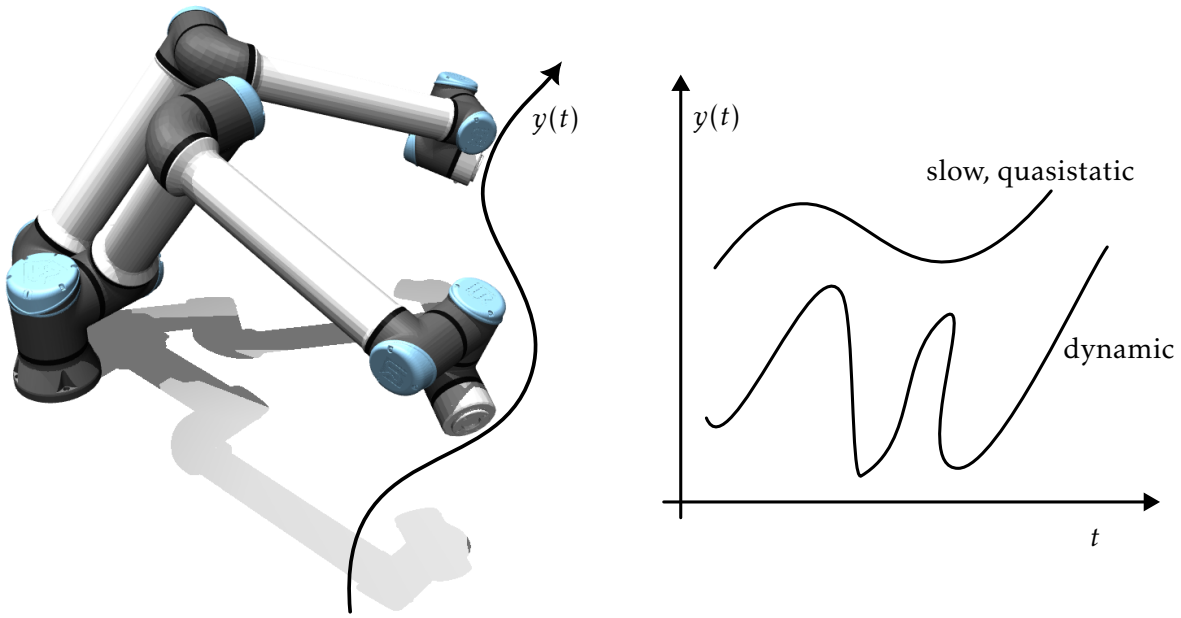


Figure 1.5: *An arm robot (here: Universal Robot 10e) can perform slow, quasistatic motions and agile, dynamic motions. Image is owned by Simon Bachhuber*[2].

required (Shannon, 1949). This implies that as motion becomes more dynamic, the ability to reconstruct the state signal is limited by the sampling rate of the available sensors. Unfortunately, high-sampling-rate sensors are often impractical for many applications due to cost and hardware limitations.

3. High-gain Observers and Controllers: Accurate-and-fast tracking-and-control requires observers and controllers with large feedback gains, which, while effective, are more susceptible to measurement and process noise. Conversely, though less sensitive to noise, low-gain observers and controllers tend to introduce higher latency, resulting in slower response times.

While data-driven control methods provide powerful algorithms for motion analysis and motion control of complex systems, they are not flawless. **Fortunately, some of these challenges can be addressed by training on vast amounts of data obtained from diverse simulation environments and leveraging the ability of ANNs to scale effectively and learn highly complex logic. After training, these solutions can be directly transferred to real-world scenarios. This process is known as sim-to-real transfer.**

1.3 Sim-to-real Transfer and Zero-data Solutions

Simulation-to-reality (sim-to-real) refers to a concept in robotics and ML (Machine Learning) that involves obtaining a solution in simulation, e.g., training a ANN on simulated data that

[2]Figure 1.5 was created by S.B. using the MuJoCo software (Todorov et al., 2012) which is publicly available under the Apache License, Version 2.0. The model of the U10e robot is part of Google's Menagerie project (Zakka et al., 2022).

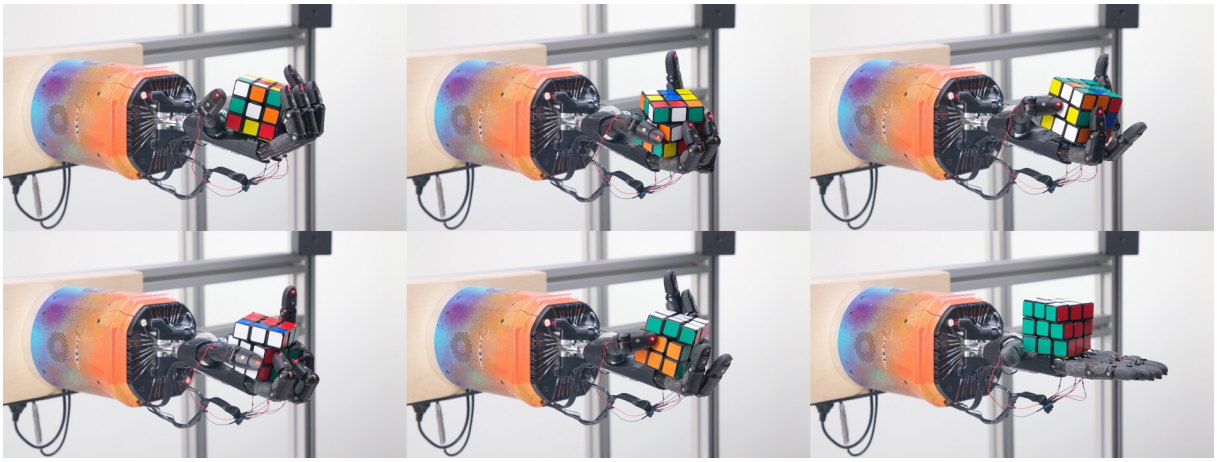


Figure 1.6: *A five-fingered humanoid hand, trained with RL and in simulation only, solving a Rubik's cube. The policy has been trained in simulation for approx. 13 thousand years. Adopted from OpenAI et al. (2019a) with consent by OpenAI.*

then gets transferred to the real world. If the transferred solution directly generalizes to real-world settings, then it is effectively a zero-(real-world)-data solution. For example, in OpenAI et al. (2019a), a ANN-based policy is trained using RL in an incredibly rich humanoid hand simulation and then later transferred to solve, for the first time, the mirrored real-world problem (see Figure 1.6).

Fueled by similar achievements, sim-to-real transfer has become a recent trend in robotics (Zhao et al., 2020). This is because **training in simulation leverages the efficiency and safety of simulations to train, validate, and refine methods**. But there is more to tell here: It is not just because simulations are cheap and safe but because **complex problems require complex solutions requiring lots (and lots) of data**. This also becomes apparent from the fact that sim-to-real is particularly well-established for learning higher-order intelligence, such as humanoid robots that learn to substitute humans in complex tasks. As an example, robot foundation models (Firoozi et al., 2023; Hu et al., 2023; Kawaharazuka et al., 2024) and most notably the Nvidia GROOT project, rely on simulated data. There is an ongoing race to build the first autonomous humanoid robot. Recently, the AI robot company Figure introduced Figure 02, while Elon Musk is working on the Optimus humanoid robot. A key component for achieving successful humanoid robots lies in their training within simulation environments, which is also crucial for building robust foundation models of embodiments Ahn et al. (2024).

In this thesis and the included publications, we utilize these recent advancements and insights by training ANNs on vast amounts of simulated data to effectively generalize from simulation to reality. The ANNs learn in simulation to solve complex motion analysis and motion control problems. By scaling both the size of the ANNs and the richness of the simulated data, we obtain trained networks that are maximally applicable, plug-and-play capable, and robust, with minimal reliance on real-world data. In the context of motion analysis and motion control for highly dynamic motions, a key insight is that the specific motion a system performs can be treated as unknown, and domain randomization can be employed to address this uncertainty. We approach generalization across these motions by simulating a broad range of dynamic motions. This strategy is applicable to both state estimation and

Figure 1.7: *Two IMUs (Inertial Measurement Units), Xsens Movella Dots, are attached to the upper and lower leg, and IMT algorithms can be used to track its motion. IMT algorithms uses sensor fusion to combine IMU data (3D (Three-Dimensional) accelerometer, 3D gyroscope, 3D magnetometer) and estimate a trajectory of orientations.*

control: for state estimation, the strategy involves generating random motions with sufficient excitation[3], while for control, it involves generating random reference motions.

1.4 Example Applications

This thesis develops a broadly applicable approach and demonstrates how it can be applied to real-world state estimation and control problems. The example applications are IMT and RT in unknown nonlinear dynamics. IMT represents a challenging motion state estimation problem from sensor data as it typically arises in real-world applications. It involves highly complex systems that can perform dynamic, agile motions (most notably the human body). RT in unknown nonlinear dynamics represents a challenging, highly relevant motion control problem involving complex systems, e.g., SRs (Soft Robots).

IMT uses body-worn inertial sensors, called IMUs, to estimate and track the motion of bodies in space. For example, in Figure 1.7, IMT is used to track the motion of a human leg using two IMUs. IMUs, which typically comprise a 3D accelerometer, a 3D gyroscope, and a 3D magnetometer, have become smaller and less expensive within the last two decades and IMT has therefore rapidly become the most promising technology for accurate, reliable, and inexpensive motion tracking, especially since camera-based systems are typically more expensive, more restrictive, and suffer from occlusion (Huang et al., 2018; von Marcard et al., 2017). Applications of IMT span across various application domains (Seel et al., 2020) ranging from aerospace engineering (Givens and Coopmans, 2019) to health applications (López-Nava and Muñoz-Meléndez, 2016).

Finite-horizon RT is the task of controlling the actuators of a system such that its output

[3]excitation refers to generating motions with enough variation or activity to fully reveal the dynamic behavior of the system

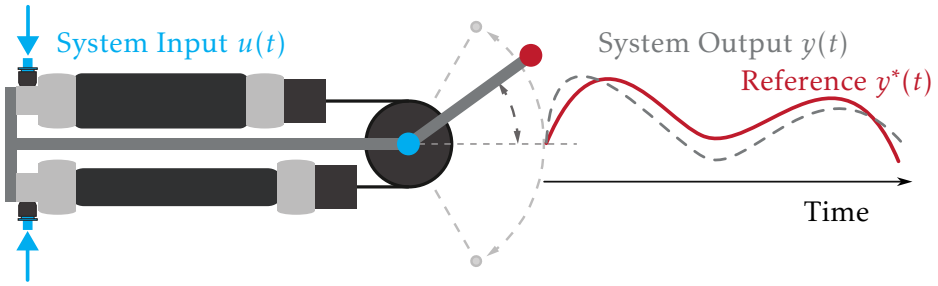


Figure 1.8: *RT is the task of controlling the output of a system such that it follows a desired reference signal for a finite duration of time.*

follows a desired reference signal for a finite duration of time. In addition, in many real-world applications, the dynamics of the system are nonlinear and unknown. For example in Figure 1.8, the angle of a robot arm is actuated by two pneumatic soft actuators that work as an antagonistic pair. Here, RT is about designing a controller that allows the robot arm to accurately track a reference signal even though the dynamics of pneumatic soft actuators are nonlinear and complex to model. Applications of RT include, e.g., autonomous piloting (Hanover et al., 2024), soft robotics (Haggerty et al., 2023), and industrial settings (Yuan et al., 2020).

1.5 Aim of the Thesis

The aim of this thesis is twofold. The first goal is to develop and validate a broadly applicable, unified approach that addresses the dual problem of state estimation and control of systems performing dynamic motions and that yields easy-to-use, plug-and-play[4] solutions in the form of trained ANNs. The developed approach uses simulation environments of complex motion analysis and motion control problems, combined with extensive domain randomizations, to train ANNs on vast amounts of simulated data. Then, by scaling both the size of the ANNs and the richness of the simulated data, the trained networks can zero-shot generalize to real-world scenarios while minimizing reliance on real-world data. One key insight is that by simulating years of random motions, the trained networks are able to generalize to any specific real-world motion.

The second goal is to demonstrate the effectiveness of the unified approach by applying it to a) the state estimation task of IMT and b) the motion control task of RT in unknown nonlinear dynamics. By applying the unified approach, we obtain solutions that address their research gaps (as defined in Section 2.2.2 and Section 2.3.2, respectively) and advance their related work (as defined in Section 2.2.1 and Section 2.3.1, respectively).

The impact of the successful development is manifold:

- Practical Solutions: The approach offers novel solutions to previously unsolved problems, as it is broadly applicable, requiring only a simulation environment and a suitable set of domain randomizations. If a simulation environment is unavailable, it can be approximated from real-world data, with domain randomizations enhancing data efficiency.

[4] solutions that do not require time investment or expertise by the user, e.g., in the form of calibration or tuning

- Automatic and Plug-and-play Applicable: The approach minimizes the required expert knowledge and time investment because a) the engineer operates at a higher level, relying on simulated data for high real-world data efficiency, and b) domain randomizations ensure robustness which reduces the need for selection, calibration, and adaption of methods.
- Centralized Development: Since specific applications are built upon the unified approach, improvements to the framework automatically benefit all downstream implementations.
- Transferability: The unified approach facilitates rapid progress across a broad range of application domains, as it is not domain-specific in its problem-solving capabilities.

To achieve this impact, the developed approach must be a solution that: 1) requires minimal problem-specific prior knowledge, 2) avoids repeated manual adaptations, 3) is plug-and-play applicable to a wide range of diverse problems, and 4) minimizes reliance on real-world data. This thesis and the included publications demonstrate that the combination of sim-to-real transfer and RNNs can provide such a unified approach.

1.6 Thesis Outline

The thesis consists of two parts. Part I contains the main body of the cumulative dissertation, and Part II contains the six included publications in non-edited, as-published format.

Part I contains six chapters. The first chapter motivates the topic of motion analysis and motion control, and introduces classical, model-based, and data-driven solutions relevant to this topic. Then, two concrete applications are introduced, namely IMT and RT, as they are used as reoccurring application examples in this thesis. The first chapter finishes by defining the aim of the thesis and outlining the content of the document (this section). The second chapter (Section 2) discusses the related work on the general topic of sim-to-real transfer and in the context of the two application examples. The required background is introduced as needed. The third chapter (Section 3) lists the publications included in this thesis and summarizes their main contributions. The fourth chapter (Section 4) provides an overview and high-level understanding through a unified lens of the specific methods developed in the included publications. It also highlights selected methods in more technical detail. The fifth chapter (Section 5) summarizes the main results of the included publications and discusses them individually and in combination. The sixth and final chapter (Section 6) summarizes the included papers' main contributions and their impact, and finishes by outlining several promising directions for future work.

2

Related Work

"The leap from theory to practice is an act of faith."
– Kenneth Stanley, Research Manager at OpenAI

This thesis uses an RNN-based approach combined with sim-to-real transfer to data-efficiently solve the dual problem of estimating and tracking state information. The approach is then applied to advance two concrete application examples: IMT and RT in unknown nonlinear dynamics. This chapter discusses the related work that uses sim-to-real transfer to solve real-world problems and also discusses related work in the context of the two application examples. The required background is introduced as needed.

2.1 Sim-to-real Transfer

Sim-to-real refers to a concept in robotics and ML that involves obtaining a solution in simulation, e.g., training an ANN on simulated data that then gets transferred to the real world. Sim-to-real is a recent trend in robotics (Zhao et al., 2020).

Training in simulation leverages the cost-efficiency and safety of virtual environments to train, develop, validate, and refine methods. This approach minimizes financial and physical risks typically associated with direct real-world experimentation. However, the benefits extend beyond cost and safety. Complex problems require complex solutions, which necessitate access to vast amounts of data. Simulations can efficiently generate this extensive data, providing the necessary resources to tackle high-complexity challenges. For example, in OpenAI et al. (2019a), a ANN-based policy is trained in simulation *for approximately 13 thousand years* to solve a real-world Rubik's Cube game with a robot hand (see Figure 1.6). A similar order of magnitude for data requirements has been observed for learning highly complex and technical computer games (OpenAI et al., 2019b) (learning Dota 2 in approximately 40 thousand years). These large data requirements are particularly evident when considering the development of higher-order intelligence. Humanoid robots that learn to perform complex tasks traditionally executed by humans significantly benefit from sim-to-real approaches. One example is robot foundation models, as seen in projects like Nvidia GR00T, an initiative to develop a general-purpose foundation model for humanoid robots that takes multi-modal instructions and past

Figure 2.1: *Various robotic arms performing different tasks in diverse environments. Comparable arm robots and environments and many other robots, e.g., quadrupeds, quadrotors, or humanoids, can be simulated with The Nvidia Isaac Lab software. Simulating a broad range of scenarios is pivotal for robot foundation model training and successful sim-to-real transfer. Image is owned by Simon Bachhuber*[1].

interactions as input and outputs robot actions. It relies on the Nvidia Isaac Lab simulation platform, a simulation environment that's optimized for robot learning and that supports all types of robot embodiments (see Figure 2.1 (Mittal et al., 2023). Robot foundation models, in general, typically rely on training in simulation before being applied in real-world settings (Hu et al., 2023). Nvidia has recently extended its ecosystem with Omniverse Replicator, a framework for developing custom synthetic data generation pipelines and services. A second example is Haarnoja et al. (2024), where the authors use Deep RL to train a humanoid robot with 20 actuated joints to play a simplified one-versus-one soccer game. The agent was trained in simulation and transferred to real robots zero-shot. This zero-shot transfer is achieved via a combination of system identification and *domain randomizations* to improve the robustness of the learned policy, e.g., they varied the location and orientation of the robot-attached IMU. However, they also detail that the sim-to-real gap was too large without system identification.

Successful sim-to-real transfer (that is, obtaining a high-performance real-world solution) relies on two components: a) accurate modeling and b) domain randomizations. To see this, consider the simulation distribution p_{sim} and real-world distribution p_{real} over the data $\mathcal{D}$. If f_θ is an ANN-based solution and $\mathcal{L}$ is some loss function, then the dilemma of sim-to-real transfer is finding a simulation distribution such that the trained solution works well on real-world data without knowing the real-world data distribution, i.e.,

$$\arg\min_{p_{\text{sim}}} \int \mathcal{L}\left(f_{\theta^*}, \mathcal{D}\right) p_{\text{real}}\left(\mathcal{D}\right) d\mathcal{D} \quad \text{where} \quad \theta^* = \arg\min_{\theta} \int \mathcal{L}\left(f_\theta, \mathcal{D}\right) p_{\text{sim}}\left(\mathcal{D}\right) d\mathcal{D} \tag{2.1}$$

If we assume that the loss function is uniformly distributed across the data, then sim-to-real transfer is about maximizing the probability that we observe real-world data that has been seen

[1] Figure 2.1 was created by S.B. using Dall-E 3. OpenAI's content policy and terms state that you own the images you create with Dall-E, including the right to reprint, sell, and merchandise.

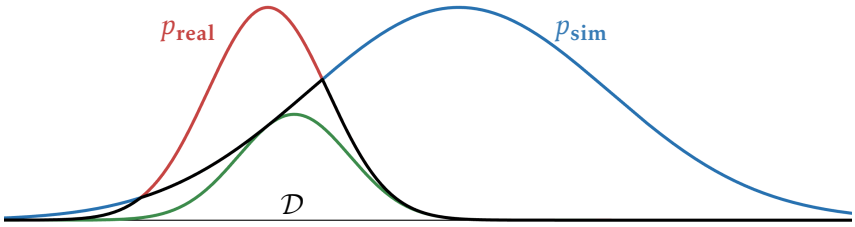


Figure 2.2: *Sim-to-real transfer requires that the simulated training data distribution p_{sim} ideally maximizes the overlap (black) or joint distribution (green) of simulated (blue) and real-world (red) data.*

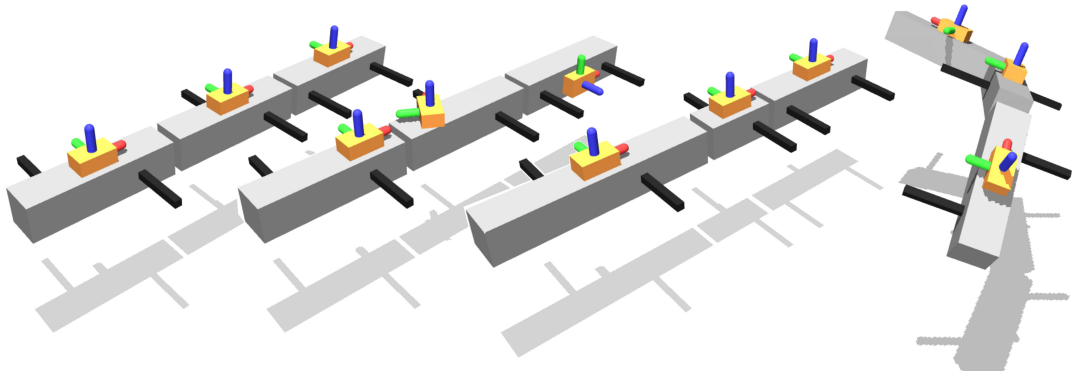


Figure 2.3: *Domain randomization techniques for a three-segment (grey boxes) KC (Kinematic Chain) with body-attached IMUs (orange boxes). The second KC is obtained by randomizing the IMUs attachment. Similarly, the third KC is obtained by randomizing the length of the segments, and the initial pose is randomized in the last KC. An RNN can be trained to estimate the rotational state of the KC from the IMU measurements by simulating random motion and the corresponding virtual IMU measurements. Suitable domain randomization techniques enable the trained RNN to generalize from simulation to experiment.*

observed in simulation, i.e.,

$$\arg\max_{p_{\text{sim}}} \int \underbrace{p_{\text{sim}}(\mathcal{D})\, p_{\text{real}}(\mathcal{D})}_{\text{joint distribution}}\, d\mathcal{D} \quad \text{or} \quad \arg\max_{p_{\text{sim}}} \int \underbrace{\min\left(p_{\text{sim}}(\mathcal{D}), p_{\text{real}}(\mathcal{D})\right)}_{\text{overlap (more conservative)}}\, d\mathcal{D}. \tag{2.2}$$

There are two scenarios: either a) uncertainty in p_{real} is low (e.g., via system identification), then a narrow p_{sim} that accurately captures reality is preferred. Or, b) uncertainty in p_{real} is high, then a broad p_{sim} that ensures that the simulation at least has support for the real-world data is well suited. Thus, domain randomization can achieve plug-and-play applicability of methods, e.g., consider Example 2.1.

Example 2.1

Suppose we want to train an ANN to estimate the attitude from 6D (Six-Dimensional) IMU measurements. Real-world gyroscopes can measure a non-zero value despite no rotation change. This phenomenon is referred to as a bias, and integrating a constant bias leads to a linear drift in time. There are two options for training an ANN such that its predictions do not drift in time: a) accurate system identification, i.e., knowledge of the precise real-world gyroscope bias value, or b) domain randomization by simulating thousands of IMUs with various gyroscope bias values such that the ANN *learns to perform bias compensation*. Note that the second option yields a more plug-and-play applicable method because it can be applied to an IMU with different bias terms without additional effort. In contrast, accurate system identification requires manual efforts and time investment.

Many applications use domain randomizations, especially in combination with Deep RL to enable successful sim-to-real transfer (Zhao et al., 2020). For example, in Matas et al. (2018), the authors use a modified DDPG algorithm, a Deep RL algorithm, to train an arm robot to manipulate clothes and other deformable objects from RGB camera observations. They use PyBullet to train in simulation and transfer the policy zero-shot to an experimental Kinova Mico 7-DOF robotic arm and a low-cost web camera as a sensor. In Rao et al. (2020), the authors propose a method titled RL-CycleGAN, which combines generative adversarial networks with Deep RL to generate realistic images of a robot arm grasping task. The solution is then transferred to an experimental Kuka IIWA robot performing several grasping tasks. In Schoettler et al. (2020), the authors use meta-RL to train control policies in simulation, which are then quickly adapted to perform real-world robotic insertion tasks with minimal real-world interaction time.

However, sim-to-real transfer comes with its challenges. At its core, we trade off less accurate modeling and broad p_{sim} domain randomizations by increasing the workload on the method or the ANN to be trained. Prediction performance decreases as the method is required to generalize over an increasingly broad spectrum of data (which requires more extensive and complex solutions). The domain randomizations force the method to (at least implicitly) learn forms of online calibration and adaption, which typically trades off filter aggressiveness with the ability to adapt. Coming back to the Rubik's Cube game example OpenAI et al. (2019a), the authors enable efficient policy training despite large domain randomizations with a process called automatic domain randomization. It automates and gradually expands the randomization ranges that parameterize a distribution over environments. Additionally, note that it is nontrivial that there continues to exist a solution as we broaden p_{sim}. As we broaden our simulation and increase the workload on the ANN to be trained, we risk entering the realm of non-observability, consider, e.g., Example 2.2 as an example of this phenomenon.

Example 2.2

Consider Example 2.1, but where we train an RNN to estimate the attitude solely from 3D gyroscope measurements. In this case, domain randomization involves simulating thousands of IMUs with various gyroscope bias values such that the RNN learns to perform bias compensation. Unfortunately, now RNN will converge to a much higher training error, regardless of the network architecture, since the state estimation can no longer be solved. The lack of a solution can easily be seen by considering the same IMU motion, however, with an additional global rotation of the IMU (superposition) with an angular velocity that corresponds exactly to the

gyroscope bias. Then, the same trajectory of IMU measurement is mapped onto *two different* attitude trajectories. I.e., the data distribution $p_{\text{sim}}(\mathcal{D})$ has gained support on two values in data space where $\boldsymbol{x}_1 = \boldsymbol{x}_2$ whilst $\boldsymbol{y}_1 \neq \boldsymbol{y}_2$. The function to be learned no longer exists. This phenomenon is an example of non-observability (see Theorem B.3).

2.2 Application A: Inertial Motion Tracking

This thesis considers IMT as one of two specific applications of state estimation and control in systems that perform dynamic motions. In this section, we primarily discuss related work and its limitations; the latter discussion then leads us to identify the open research gap in IMT. We also introduce the challenges that arise when fusing IMU measurements to estimate orientations.

The need for reliable and accurate estimation of the orientation, attitude, or pose of articulated objects in 3D space spans across various application domains ranging from aerospace engineering (Euston et al., 2008; Givens and Coopmans, 2019) to health applications (Buke et al., 2015; López-Nava and Muñoz-Meléndez, 2016; Seel et al., 2020) and methods that address this need are known as motion capture methods. Motion capture is a broad field that uses various methodologies. Methods can be divided by the sensors used into camera-based and IMU-based methods. Camera-based methods may further be subdivided into marker-less or marker-based methods. Marker-based methods are often referred to as OMC. They combine multiple cameras with body-attached markers that act as beacons. They offer the highest precision and are commercially available. Well-known manufacturers are Vicon, OptiTrack, and Xsens. Marker-less methods use one or more cameras but without body-attached markers. Multiple methods have been proposed Chen et al. (2021); Mehta et al. (2020); Trumble et al. (2016) but these methods can a) suffer from occlusion, and b) offer poor robustness for human motion capture applications.

IMU-based methods, also known as IMT, use one or multiple body-attached IMUs to estimate the attitude, orientation, or pose of a single coordinate system or complex articulated structures, most notably the human body. IMUs, which typically comprise a 3D accelerometer, a 3D gyroscope, and a 3D magnetometer, have become smaller and less expensive within recent years and IMT has therefore rapidly become the most promising technology for accurate, reliable, and inexpensive motion tracking of rigid bodies and KCs. Example manufacturers of IMUs are Xsens (or, nowadays, Movella), Shimmer, Bosch Sensortec, Analog Devices, or STMicroelectronics. Since IMUs do not measure orientations directly, dedicated algorithms are necessary to estimate them. These algorithms will be discussed in the next section.

2.2.1 Related Work

IMUs do not measure orientations directly; in a process called sensor fusion, the accelerometer, gyroscope, and magnetometer measurements are combined to estimate orientations accurately. In a situation in which a well-calibrated IMU is kept still in an environment with no nearby ferromagnetic materials, the 3D orientation can be straightforwardly computed using the signals from the accelerometer and magnetometer, much like using a water level and a compass needle. Similarly, the vikings relied on the Polar Star to obtain the northbound direction, as illustrated in Example B.1. However, a gyroscope is essential when the sensor rotates and moves

quickly (xse, 2018). The two input sources (accelerometer & magnetometer and gyroscope) are complementary by nature. The gravitational and magnetic components give long-term stabilizing information, while the gyroscope gives high-bandwidth, responsive movement signals. Suppose the sensor fusion is applied independently for each IMU. In that case, the method is referred to as *single-IMU sensor fusion*, for the task of orientation estimation, this also known as IOE (Inertial Orientation Estimation); in contrast, *Multiple-IMU sensor fusion* fuses the measurements of all available IMUs with a central logic (and not independently for each IMU).

Single-IMU Sensor Fusion

For IOE, several model-based and ML-based methods have been proposed. Both Madgwick (2010) and Mahony et al. (2008) offer simple and computationally efficient IOE algorithms with similar albeit slightly different methodologies. Madgwick (2010) uses an iterative gradient descent algorithm and updates the orientation estimates by minimizing the error between the estimated quaternion and the measured gravity and magnetic field vectors. Mahony et al. (2008) uses a PID feedback controller to correct the drift from integrating gyroscope data. It adjusts the orientation estimate directly based on the error calculated from the accelerometer and magnetometer readings. In Zhang et al. (2012), a Kalman-filter-based IOE algorithm is proposed that uses a model where the gyroscope constitutes the input of the model's dynamics and accelerometer and magnetometer readings are used as pseudo measurements[2].

However, all three previous methods are prone to the malicious effects that inhomogeneous magnetic fields, which are often found in indoor environments, can have on the *inclination* portion of orientation estimates (Seel and Ruppin, 2017). The inclination is the orientation of an object relative to the horizontal plane (assuming the vertical direction is parallel to the earth's gravity direction), and it indicates how much an object is tilted forward or backward. *Heading* and inclination decomposition refers to breaking down an object's orientation into a first rotation around the vertical direction or gravity vector with the heading angle (or yaw angle) and a subsequent rotation of the inclination (the pitch and roll angles). Such a decomposition is used in Seel and Ruppin (2017) to ensure that orientation updates that correct the drift from integrating gyroscope data based on the magnetometer affect only the heading component of the orientation estimate. A similar ansatz[3] is used in Laidig and Seel (2023) and combined with highly-effective low-pass-filtering. The resulting VQF (Versatile Quaternion-based inertial orientation estimation Filter) filter constitutes the current SOTA (State-of-the-Art) in IOE.

However, if the magnetic field is heavily distorted, e.g., in proximity to ferromagnetic material or electric devices (de Vries et al., 2009; Weygers et al., 2023), then a magnetometer-free approach, omitting the usage of the magnetometer entirely, is required. In this magnetometer-free scenario, only the inclination and not the heading component of the whole orientation is estimated. For example, for the task of inclination estimation only (a subset of IOE), the ML-based RIANN (Robust IMU-based Attitude Neural Network) filter as proposed in Weber et al.

[2] pseudo measurements refer to artificial measurements that are not directly measured by physical sensors. Instead, they are constructed by combining sensor measurements with system knowledge, model predictions, constraints, or assumptions and are used to support the estimation process of a Kalman filter.

[3] The word "ansatz" comes from the German language and refers to a mathematical approach based on an educated guess about the form of a solution. The word became widely adopted in the early 20th century, primarily as a result of German-speaking scientists and mathematicians who contributed heavily to fields like quantum mechanics (e.g., Max Planck, Albert Einstein, Werner Heisenberg).

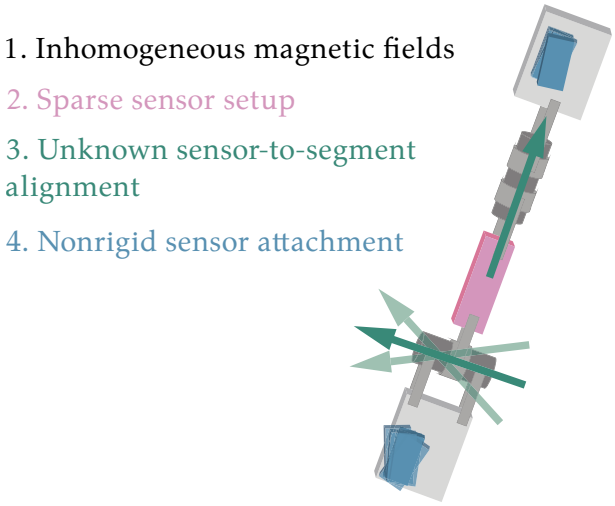


Figure 2.4: *A three-segment KC with two IMUs (blue boxes) that demonstrates four challenges of IMT. Multiple-IMU sensor fusion can be used to overcome these challenges. Reprinted from Bachhuber et al. (2024c).*

(2021) which uses an RNN to fuse the magnetometer-free IMU measurements (3D gyroscope + 3D accelerometer), rivals VQF.

Four Challenges of IMT

In IMT, there are at least four challenges. Inhomogeneous magnetic fields that distort magnetometer measurements are the first of these challenges that IMT algorithms are confronted with when estimating orientations. The lack of a heading direction in magnetometer-free IOE results in an unknown heading offset between two coordinate systems, necessitating additional algorithms for correction. Such correction algorithms often introduce a second challenge for IMT by requiring accurate modeling in sensor coordinates. This results in the need for sensor-to-segment calibration, i.e., identifying the joint position and axis orientations in local sensor coordinates. In addition, sensor-to-segment calibration is essential for tracking anatomically meaningful quantities. Additionally, while traditional IMT uses one IMU per segment, solving sparse problems in which some segments of the articulated, rigid-body system are not equipped with a sensor would significantly improve usability and reduce costs. Sparse IMT constitutes a third challenge for IMT. Finally, IMT usually relies on skin-attached IMUs , e.g., via velcro bands (see Figure 2.5). This attachment can introduce *motion artifacts* due to the relative motion of the soft tissue between segment and sensor and poses a fourth challenge for IMT as they require effective compensation for accurate segment tracking. The "four challenges of IMT" refers to these challenges, and they are illustrated in Figure 2.4. These challenges can be addressed by multiple-IMU sensor fusion algorithms.

Multiple-IMU Sensor Fusion

Multiple-IMU sensor fusion algorithms can be categorized into model- and ML-based algorithms. In general, multiple-IMU sensor fusion aims to overcome the previously mentioned challenges of IMT. Model-based algorithms rely on mechanical or biomechanical models to constrain a degenerate solution space. In contrast, ML-based algorithms typically do not require modeling. Instead, these algorithms are obtained by training a ML algorithm on input (IMU measurements) and output (orientation) data (real-world or simulated).

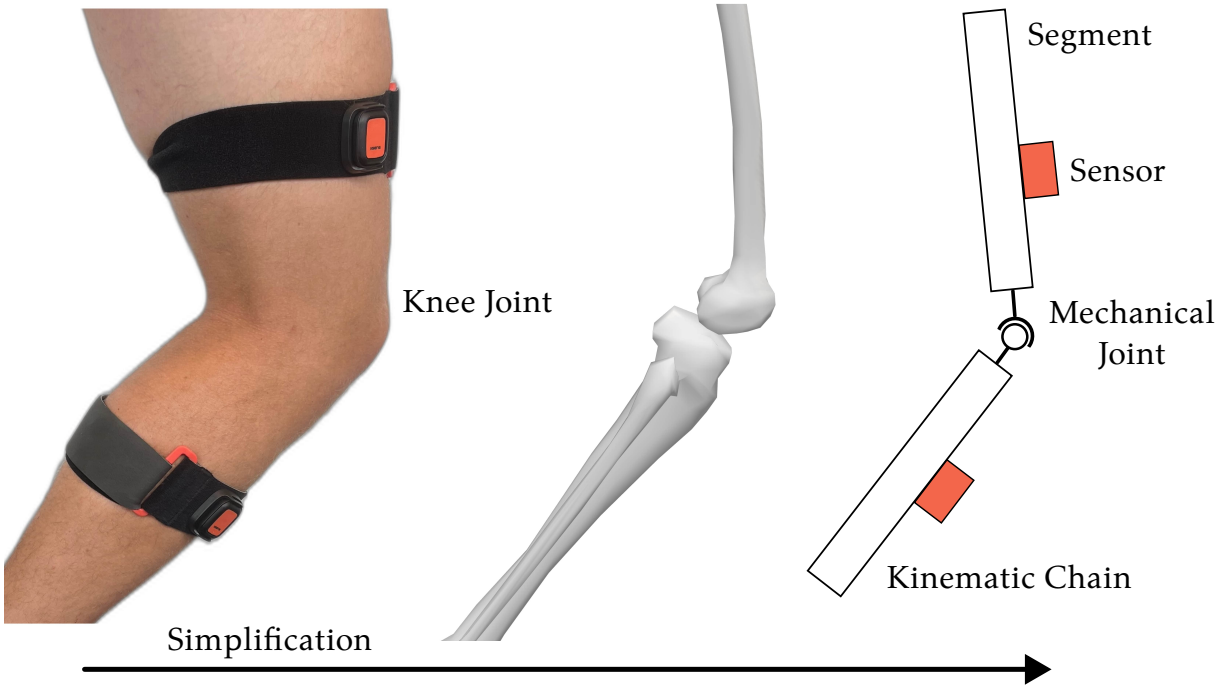


Figure 2.5: *Two IMUs are attached to a human's lower leg (left side). The knee joint's kinematics can be modeled as two segments connected by a 1D (One-Dimensional) hinge joint (right side). Multiple-IMU sensor fusion algorithms can be used to a) track the relative orientation of the knee without relying on magnetometer data, b) estimate sensor-to-segment calibration parameters, e.g., the knee joint axis direction in sensor coordinates, and c) compensate for soft tissue motion artifacts, i.e., track the bone motion from skin-attached sensors.*

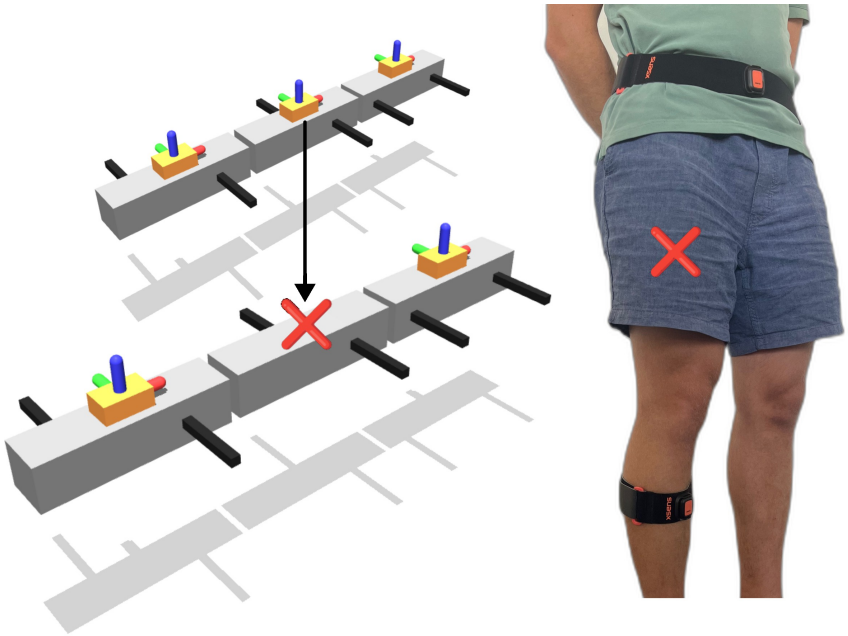


Figure 2.6: *Sparse IMT uses less than one sensor per segment. A three-segment with no IMU on the middle segment; a model of a human leg that consists of the KC of hip, thigh, and shank.*

A large body of model-based algorithms uses mechanical models of joints to derive kinematic constraints between adjacent segments. For example, Laidig et al. (2017) is a quaternion-based method that enables magnetometer-free IMT for a mechanical 1D joint model. It uses projections of the joint axes into the horizontal plane and is evaluated in the context of the knee and finger (interphalangeal) joints. Conceptually similar methods have been proposed to enable magnetometer-free IMT of 2D (Two-Dimensional) and 3D joints. For a 2D joint, Laidig et al. (2019) corrects the heading offset by exploiting the lack of a rotation axis in the relative orientation. It is experimentally validated for the metacarpophalangeal joints between the palm and the fingers. 3D joints can be tracked without the usage of magnetometers, either by exploiting range of motion constraints of an anatomically relevant Euler angle decomposition (Lehmann et al., 2020a), or by exploiting the assumption that IMUs are rigidly connected to a shared joint center point (Weygers et al., 2020). Biomechanical models can offer superior accuracy for human joints compared to mechanical models. Figure 2.5 shows the simplification process of the knee joint's kinematics as a 1D hinge joint. Consequently, similar methods exist with biomechanical models. For example, Kok et al. (2014) uses a full-body biomechanical model to enable magnetometer-free IMT. It is experimentally validated on the lower human body.

Similarly, multiple-IMU, model-based sensor fusion algorithms have been developed for sensor-to-segment calibration and sparse IMT. For sensor-to-segment calibration, the prior work by Olsson et al. (2020) proposes a plug-and-play method that exploits a 1D hinge joint model to estimate the joint axis direction in the sensor frame, and Taetz et al. (2016) uses a biomechanical model. For sparse IMT, Sy et al. (2021) uses a Kalman filter for tracking the lower body (pelvis, both thighs, both shanks) from a reduced IMU count of three instead of five. In Grapentin et al. (2020), both sparse and magnetometer-free IMT advantages have been successfully combined. They achieve hand motion tracking with a reduced sensor count of five instead of 16 magnetometer-free IMUs.

However, using a reduced number of (potentially even magnetometer-free) sensors and

calibration-free IMT naturally spirals to non-observability. Moreover, assessing observability properties is integral for providing reliable state estimates and is vital for the safe operation of downstream applications. Unfortunately, analytical observability analysis is a) limited due to the nonlinear dynamics and b) impractical in large articulated bodies. Still, isolated results have been achieved. For example, Eckhoff et al. (2020) has derived an observability condition for a model-based double-hinge-joint three-segment KC. Interestingly, it was found that the observability depends on the performed motion. A similar observation was made in von Marcard et al. (2017), where different human poses lead to drastically different orientation errors.

Compared to model-based methods, ML-based multiple-IMU algorithms do not require modeling. Instead, a solution is learned from data. ML-based solutions have been primarily proposed for human motion tracking where large ANNs are trained on input-output data of real-world experiments. Typically, they fuse six or more IMUs in order to estimate the full-body pose and are available magnetometer-aided (Huang et al., 2018; von Marcard et al., 2017; Zheng et al., 2021), or magnetometer-free (Van Wouwe et al., 2023; Yi et al., 2021, 2022). However, while addressing a challenging problem, these methods are limited to human motion capture with one specific sensor setup and assume statistical patterns of human motion (von Marcard et al., 2017), or full-body biomechanical models (Yi et al., 2022) to constrain the estimated pose. Interestingly, excluding the IOE method RIANN (Weber et al., 2021), ML-based IMT methods are largely limited to human motion capture. These limitations lead us to define the following research gaps.

2.2.2 Research Gap

This section discusses the research gaps and questions in IMT relevant for this thesis, and that are not yet addressed with prior work. *In IMT, there is a need but no solution for:*

- (Research Gap A) A method that assesses observability. Observability (see Definition B.2) is a key property for ensuring feasible, consistent, and safe state estimation, and in order to push the boundaries of IMT, it is useful first to assess the observability properties of the problem at hand, that is, ensuring that the estimation problem can be solved under perfect, in-silico conditions. As such, we aim to develop a method for observability analysis of IMT problems. Currently, there only results on the observability properties of individual, specific KC configurations. Moreover, the developed methods may have broader applicability to other domains involving ARBSs (Articulated Rigid Body Systems), suggesting opportunities for transfer.

- (Research Gap B) A method for motion artifact reduction. The entirety of IMT depends on body-attached IMUs to track the motion. However, often, while the underlying segment motion is the target variable, the IMUs are attached in a way that allows for relative motion between the IMUs and the underlying segments. This relative motion most notably occurs in the case of IMUs that are attached to human tissue (typically via velcro bands, see Figure 2.5) or that are attached to clothes. This undesired relative motion introduces motion artifacts in the estimated segment motion.

- (Research Gap C) A method that allows to address an arbitrary combination of a set of the four IMT challenges. Currently, most methods only address a single challenge in isolation;

only a few selected methods combine more challenges and none all. However, overcoming multiple challenges simultaneously is crucial to enable accurate, least restrictive, and easy-to-use IMT in real-world conditions.

- (Research Gap D) A plug-and-play pluripotent method to dramatically increase usability, currently multiple-IMU sensor fusion requires extensive expert knowledge. A pluripotent method is problem unspecific, i.e., it can be applied to a broad range of problems. A plug-and-play method can be applied without calibration or tuning. Currently, to apply IMT, the user must successfully identify the method suitable for the given problem and typically specify parameters such as joint axes' directions and sensor placement. Therefore, the user must be an expert in the field of IMT, which strongly limits the use of IMUs in many application domains. Moreover, a given problem might require a nontrivial combination of methods which may exclude each other.

By addressing the above research gaps, we also provide answers to the following research questions:

- What KCs can be accurately tracked with which IMU sensor setups under ideal, in-silico conditions? Can the IMUs be placed arbitrarily? Can the joints be 1D, 2D, or even 3D? To what extent is sensor-to-segment calibration required?
- Can ANNs be cheaply trained in simulation and afterward zero-short generalize from simulation to experiment?
- Is it possible to estimate the pose of a three- or even four-segment KC using only two 6D IMUs?
- Can ANNs learn to filter out motion artifacts due to nonrigidly attached IMUs and reconstruct the underlying segment motion?

2.3 Application B: Reference Tracking in Unknown Nonlinear Dynamics

This thesis considers RT in unknown nonlinear dynamics as the second of two specific applications of state estimation and control in systems that perform dynamic motions. Specifically, RT is an example of control that includes complex systems such as, e.g., SRs, and that involves agile, dynamic motions in the form of reference signals. As in the previous section, we discuss related work and its limitations and define the open research gap in RT.

Real-world applications such as, e.g., autonomous piloting (Hanover et al., 2024), soft robotics (Haggerty et al., 2023), and in industrial settings (Yuan et al., 2020), require controlling the output of a system with initially unknown nonlinear dynamics such that it follows a desired reference signal for a finite duration of time. This task is known as RT and can be formalized as follows: Assume there exists some unknown system dynamics Ψ that maps a time-varying, finite-time input signal $\boldsymbol{u}(t) \in \mathbb{R}^p$ to a possibly noisy output vector $\boldsymbol{y}(t) \in \mathbb{R}^q$, that is

$$\boldsymbol{y}(t) = \Psi\left[\boldsymbol{u}(t' < t)\right] \quad \forall t \in [0, T], \tag{2.3}$$

where $T \in \mathbb{R}$ is the finite trial duration. Note that Ψ includes the dependency on the unknown initial state $\boldsymbol{x}(t = 0)$. Then, RT (as considered here) aims to design a feedback controller that

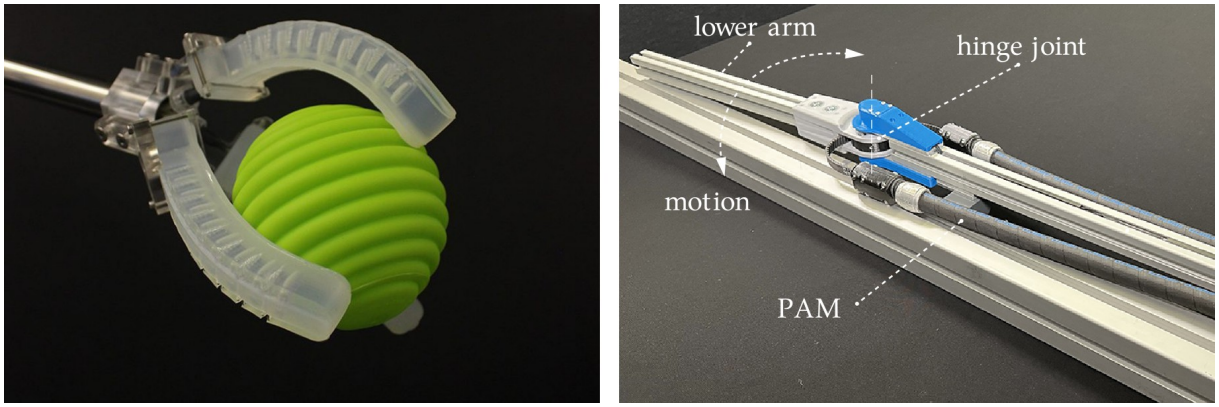


Figure 2.7: *A SR that grips a ball (left side; image reused with explicit consent[4]) and a robot arm actuated by a PAM (Pneumatic Artificial Muscle) (right side; adapted from Bachhuber et al. (2024a)). Both systems are complex and difficult to model, making data-driven control methods desirable.*

manipulates $\boldsymbol{u}(t)$ to let $\boldsymbol{y}(t)$ follow a given time-varying reference signal $\boldsymbol{y}^*(t) \in \mathbb{R}^q$. Thus, we seek to find a controller dynamics Φ that maps $\boldsymbol{y}^*(t)$ and $\boldsymbol{y}(t)$ to the input vector $\boldsymbol{u}(t)$, i.e.

$$\boldsymbol{u}(t) = \Phi\left[\boldsymbol{y}^*(t' < t), \boldsymbol{y}(t' < t), t\right] \quad \forall t \in [0, T], \tag{2.4}$$

such that it minimizes the tracking error between the output and the reference signal, i.e.

$$\Phi^* = \arg\min_{\Phi} \int_0^T \left\| \boldsymbol{y}^*(t) - \boldsymbol{y}(t) \right\|_2 dt, \tag{2.5}$$

over the finite trial duration.

This problem formulation is multifarious:

- A subset of the state of the system might be observed, i.e., $\boldsymbol{y} \subseteq \boldsymbol{x}$,
- the dimensionality of the system state might be unknown,
- the initial state $\boldsymbol{x}(t = 0)$ might vary between trials and might be unknown,
- the reference signal might vary between trials and might not be provided beforehand.

2.3.1 Related Work

The prior work that can enable accurate RT is rich. Different fields with different methodologies offer solutions, but each has its trade-offs and limitations.

RL offers a general framework encompassing almost any problem formulation. However, only a limited number of prior works exist on RL for RT. This might be due to the different scope

[4]This image is reused material of the publication Truby et al. (2018). Explicit consent was given by both the publisher, Wiley, and the first author, Ryan L. Truby. The license is available here `https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=d8b2ba1c-5435-4cf8-8f88-41a04d0b5cb4`.

of RL. RL typically assumes full state knowledge, learns unsafely by trial-and-error, and learns controllers by maximizing reward with reward functions that are often limited to a single task or motion. For example, the influential work of Deisenroth and Rasmussen (2011) proposes PILCO (Probabilistic Inference for Learning Control) and learns state-feedback controllers for setpoint tracking. It combines a Gaussian process dynamics model with gradient-based policy improvement to improve iteratively over several trials. It has been validated on several experimental systems and offers outstanding data efficiency with only 20 – 90 seconds of interaction time. However, PILCO requires full state knowledge at all times and only considers single-task learning, i.e., the feedback controllers have no reference or target state input.

This limitation can be addressed by forms of multi-task RL (Teh et al., 2017) or via multi-objective reward functions (Friedman and Fontaine, 2018). For example, Zhai et al. (2023) uses Deep RL to enable collision-free RT control of surface vehicles. However, large amounts of training data are required, and the method is only validated in simulations rather than experiments. To address the issue of full state knowledge, POMDP (Partially Observable Markov Decision Process) can be used. RL methods tailored for POMDPs have been developed (Hausknecht and Stone, 2015) and include, e.g., a recurrent policy function (Wierstra et al., 2010), stacking multiple observations, or, as a patch, a recurrent filter prior to a MDP (Markov Decision Process)-RL algorithm (Schäfer, 2008). Unfortunately, the applicability of these methods is limited by large data requirements (Deisenroth and Rasmussen, 2011; Schuitema, 2012). Finally, to overcome the issues of single-task learning and full-state knowledge, another solution might be to interpret a varying number of motions or reference signals as part of a single POMDP. Every time the environment is reset, the reference signal changes, and with it, the observations and rewards. This concept is introduced in Section 4.3.1. Overall, RL is a general framework capable of addressing many problem formulations. However, it has limitations in that it typically assumes full state knowledge, learns unsafely through trial-and-error, and is often constrained to single-task learning with reward functions that are not well-suited for multi-task scenarios.

The field of OC (Optimal Control), including most notably MPC, typically optimizes a state-dependent quadratic loss and requires a model of the system dynamics. Recent data-driven OC approaches learn models from data. For example, Torrente et al. (2021) uses Gaussian processes combined with MPC to enable precise, highly agile motion tracking with quadrotors. The MPC enables real-time, multi-task feedback control at 50 Hz, but it a) requires access to the system state at all times (the real-world quadrotor flies inside a OMC tracking range and is equipped with markers) and b) requires knowledge of a nominal model of the dynamics and only learn the model mismatch from data. As a second example for OC, in Bevanda et al. (2022), Koopman operators are used to learn high-dimensional, linear models of nonlinear dynamics. The linear model is combined with LQR controller design for state-feedback control. The solution is validated in simulation. Overall, OC is a framework that typically optimizes a state-dependent quadratic loss and relies on a system dynamics model. However, it has limitations in that it requires access to the full system state and depends on a nominal model, with only the model mismatch being learned from data in recent data-driven approaches.

ILC (Iterative Learning Control) is a learning control strategy used for RT in systems that perform the same task repeatedly. In contrast to the previous methods, ILC learns feedforward, not feedback controllers, making the approach inherently more prone to external disturbances. Additionally, ILC typically assumes a linear, time-invariant system model and learns to perform only a single task (Bristow et al., 2006). Data-driven ILC does not require a system model. Instead, it iteratively learns a model from input-output data. For example, Meindl et al. (2022)

iteratively applies an input trajectory to the unknown dynamics, trains a Gaussian process model based on the experimental data, and utilizes the model to update the input trajectory until the desired tracking performance is achieved. The proposed method is validated on three experimental setups. However, while impressive, the work and ILC in general is still mainly focused on single-task learning and learns feedforward control, making it more susceptible to external disturbances.

One exemplary application field that requires RT in unknown nonlinear dynamics is fields that involve SRs. SRs are gaining significant interest in diverse (bio)medical and industry application domains due to their inherent soft characteristics, which provide SRs with unique advantages over their rigid counterparts (Walker et al., 2020). Unfortunately, their unique soft materials put additional strain on control engineers since modeling these soft structures is challenging. Currently, the task of RT for SRs is addressed by either a) model-based control methods for SRs or b) data-driven methods. Model-based control methods typically require a detailed system model for accurate control and, therefore, involve extensive system modeling combined with human expertise (Della Santina et al., 2023; Johnson et al., 2021). In contrast, data-driven methods facilitate this process. For example, Johnson et al. (2021) uses a hybrid approach that combines an analytical model of the SR with a ANN-based model. The hybrid model is combined with MPC to control an experimental SR continuum joint. In Centurelli et al. (2022), Deep RL is used for non-agile, quasi-static RT of a real-world SR arm. First, an RNN-based model is trained to approximate the SR's dynamics. The SR arm is equipped with markers, and the pose and state of the SR are tracked with an OMC system. Then, a ANN-based state-feedback controller is trained by closing the loop with the RNN-based model, and the controller's parameters are optimized using RL. The trained controller is then transferred to the real-world system. In Haggerty et al. (2023), Koopman operators are combined with LQR to achieve RT with a SR arm. The method is validated with a real-world SR arm that reaches end-effector velocities of up to $1.52\,\mathrm{m\,s^{-1}}$ and requires 5 minutes of training data. Overall, control algorithms for SRs are progressing quickly in their ability to handle the unique challenges posed by soft materials. However, they still have the limitations that model-based approaches require extensive system modeling and expertise. At the same time, data-driven methods often need large amounts of training data and are typically limited in agility or task flexibility. These limitations lead us to define the following research gaps.

2.3.2 Research Gap

This section discusses the research gaps in RT that prior work has not yet addressed, and that are relevant for this thesis. *In RT, there is a need but no solution for:*

- (Research Gap E) A method
  - that enables accurate RT of agile, dynamic, non-repetitive reference motions. ILC typically requires repetitive reference motions.
  - that can be applied to systems with unknown nonlinear dynamics and that, via repeated measurements, autonomously adapts to the unknown dynamics. OC typically requires a system model.
  - that can be applied to systems where only a system output is measured and *not* the full state of the system; additionally, the state dimensionality can be unknown. RL typically requires that the system state is observed.

  - that does not require expert knowledge but instead exposes a minimal set of intuitive hyperparameters and automatically operates and accurately controls the system without further user intervention.
  - that is highly data-efficient and offers high-performance control after a minimal amount of system interaction time.
  - that generalizes (or quickly adapts) to a broad range of qualitatively different reference motions. Motions that might not have been observed before or during training.
  - that is robust to external disturbances (such as mechanical user intervention) and noise.
  - that offers comprehensive validation in simulation and experiment.

- (Research Gap F) A method that enables SRs to learn to perform agile, non-repetitive motions from only a parsimonious amount of experimental interaction time and without requiring any prior model knowledge. Prior work in SR requires either modeling, does not enable agile motions, or is not data-efficient. This makes the solution time-consuming, labor-intensive, or restricted.

By addressing the above research gaps, we also provide answers to the following research questions:

- For what systems with nonlinear dynamics can we learn high-performance output feedback control?
- What are the data requirements? How much input-output data is needed?
- How robust is data-driven learning control to noisy measurements and external disturbances?
- Is a trial-invariant initial state strictly required?

3

Publications

"What gets measured, gets recognized."
– Peter Drucker, American author

This chapter lists the publications included in this thesis and summarizes their main contributions.

3.1 Listing of Publications

Part II includes six published or accepted papers relevant to this thesis. They are listed below in chronological order.

Paper A: RNN-based Observability Analysis for Magnetometer-Free Sparse Inertial Motion Tracking

Paper A is the publication of

> Simon Bachhuber, Daniel Weber, Ive Weygers, and Thomas Seel. RNN-based Observability Analysis for Magnetometer-Free Sparse Inertial Motion Tracking. In *2022 25th International Conference on Information Fusion (FUSION)*, pages 1–8, Linköping, Sweden, July 2022. IEEE. ISBN 978-1-73774-972-1. doi: 10.23919/FUSION49751.2022.9841375.

Summary: IMUs are widely used for IMT of KCs in numerous applications, and while magnetometer-free sensor fusion enables reliably high accuracy in indoor environments and near magnetic disturbances, the use of sparse sensor setups would yield additional advantages in cost, effort, and usability. However, it is unclear which sparse sensor setups can be used to track which motions of which KCs since the observability of the underlying nonlinear dynamics is barely understood to date. This paper proposes a method, named RNNO (Recurrent Neural Network-based Observer), that utilizes RNNs and automatically generated training data to assess the observability of the relative pose of KCs in sparse IMT systems. We apply

this method to a range of double-hinge-joint systems that perform fully-exciting random motion. Results show how the degree of observability depends on the kinematic structure and that RNN-based observers can achieve small tracking errors in an extensive range of sparse and magnetometer-free setups. RNNO enables systematic assessment of observability properties in complex nonlinear dynamics and represents a crucial step toward enabling reliably accurate and non-restrictive IMT solutions.

Background and author contributions: The idea of utilizing ANNs to assess the observability of systems by training on simulated random motion has been initially proposed by DW and TS prior to the start of SB's PhD work. SB has developed the idea into a valid method with the help of fruitful discussions with DW and TS. SB has implemented all algorithms. IW and TS have supported SB in manuscript writing, structure, and formatting.

Paper E: Neural ODEs for Data-Driven Automatic Self-Design of Finite-Time Output Feedback Control for Unknown Nonlinear Dynamics

Paper E is the publication of

> Simon Bachhuber, Ive Weygers, and Thomas Seel. Neural ODEs for Data-Driven Automatic Self-Design of Finite-Time Output Feedback Control for Unknown Nonlinear Dynamics. *IEEE Control Systems Letters*, 7:3048–3053, July 2023b. ISSN 2475-1456. doi: 10.1109/LCSYS.2023.3293277.

Summary: Many application fields, e.g., robotic surgery, autonomous piloting, and wearable robotics, greatly benefit from advances in robotics and automation. A common task is to control an unknown nonlinear system such that its output tracks a desired reference signal for a finite duration of time. A learning control method that automatically and efficiently designs output feedback controllers for this task would significantly boost practicality over time-consuming and labor-intensive manual system identification and controller design methods. In this paper, we propose ANODEC, a data-efficient automatic design of output feedback controllers for finite-time RT in systems with unknown nonlinear dynamics. In-silico validation shows that ANODEC can – automatically – design competitive controllers that outperform two controller baselines and achieve an on average $\approx$ 30%/17% lower median RMSE (Root Mean Squared Error). This performance is demonstrated in four nonlinear systems using multiple, qualitatively different, and even out-of-training-distribution reference signals.

Background and author contributions: This paper's relatively straightforward idea has been proposed by SB: If both model and controller are modeled using NODEs (Neural Ordinary Differential Equations), then their closed-loop is again a differentiable NODE. This NODE should ideally behave like a unit mapping for a broad range of reference signals, and the controller's parameters may be optimized using backpropagation to achieve this. TS has offered guidance for developing and evaluating the method. SB has implemented all algorithms. IW and TS have supported SB in manuscript writing, structure, and formatting.

Paper B: Plug-and-play Sparse Inertial Motion Tracking With Sim-to-Real Transfer

Paper B is the publication of

> Simon Bachhuber, Dustin Lehmann, Eva Dorschky, Anne D. Koelewijn, Thomas Seel, and Ive Weygers. Plug-and-Play Sparse Inertial Motion Tracking With Sim-

to-Real Transfer. *IEEE Sensors Letters*, 7(10):1–4, October 2023a. ISSN 2475-1472. doi: 10.1109/LSENS.2023.3307122.

Summary: IMUs are used for IMT in a growing number of applications as sensor fusion methods are being advanced in three directions: magnetometer-free IMT methods that eliminate the effect of magnetic disturbances; sparse IMT approaches that lead to reduced setup complexity; and automatic self-calibration of sensor-to-segment positions or orientations. In this paper, we propose an approach that combines all three achievements and, for the first time, enables plug-and-play, magnetometer-free, and sparse IMT. This is accomplished by training an RNNO on just-in-time simulated motion data of KCs. We demonstrate that domain-specific training data augmentations lead to a trained RNNO, which zero-shot generalizes to previously unseen experimental data and, thus, overcomes the sim-to-real gap. The trained RNNO achieves a tracking error of < 4 degrees when estimating the relative pose of a three-segment kinematic chain with two hinge joints. The proposed method offers a novel simulation-data-driven approach for solving complex sparse sensing problems while assuring robust and plug-and-play generalizability to experimental data.

Background and author contributions: SB has proposed the idea that the in paper A trained RNNO might zero-shot generalize to real-world data. TS and IW have provided guidance for developing suitable domain randomization techniques to close the sim-to-real gap successfully. SB has implemented all algorithms. DL has planned and conducted experiments and recorded real-world validation data. ED, AK, and IW have supported SB in manuscript writing, structure, and formatting.

Paper D: Dispelling Four Challenges in Inertial Motion Tracking with One RING

Paper D is the publication of

Simon Bachhuber, Ive Weygers, and Thomas Seel. Dispelling four challenges in inertial motion tracking with one recurrent inertial graph-based estimator (RING). In *12th IFAC Symposium on Biological and Medical Systems - 12th BMS 2024*, September 2024c. doi: 10.48550/arXiv.2409.02502.

Summary: This paper extends RING, a novel neural-network-based solution for IMTs, to generalize across an extensive range of sampling rates. We demonstrate that it can overcome four real-world IMT challenges: inhomogeneous magnetic fields, sensor-to-segment misalignment, sparse sensor setups, and nonrigid sensor attachment. RING can estimate the rotational state of a three-segment KC with double hinge joints from inertial data and achieves an experimental MAE (Mean Absolute Error) of 8.10 ± 1.19 degrees if all four challenges are present simultaneously. We conduct an ablation study to analyze the impact of each of the four challenges on RING's performance, showcase its robustness to varying sampling rates, and demonstrate that RING is capable of real-time operation.

Background and author contributions: SB has proposed the idea that the in paper C trained ANN can overcome four IMT challenges simultaneously. SB has implemented all algorithms. DL and SB have planned experiments. DL, TS, and SB have conducted experiments and recorded real-world validation data. IW has supported SB in the manuscript writing, structure, and formatting.

Paper F: A Soft Robotic System Automatically Learns Precise Agile Motions Without Model Information

Paper F is the publication of

> Simon Bachhuber, Alexander Pawluchin, Arka Pal, Ivo Boblan, and Thomas Seel. A soft robotic system automatically learns precise agile motions without model information. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2024a. doi: 10.48550/arXiv.2408.03754.

Summary: Many application domains, e.g., in medicine and manufacturing, can greatly benefit from pneumatic SRs. However, the accurate control of SRs has remained a significant challenge to date, and conventional control design methods often require significant amounts of human expertise. In recent works, the data-driven method, ANODECs, has been successfully used to – fully automatically and utilizing only input-output data – design controllers for various nonlinear systems in-silico, without requiring prior model knowledge or extensive manual tuning. In this work, we successfully apply ANODECs to automatically learn to perform agile, non-repetitive RT motion tasks in a real-world SRs and within a finite time horizon. To the best of the authors' knowledge, ANODECs achieves, for the first time, performant control of a SR with hysteresis effects from only 30s of input-output data and without any prior model knowledge. Overall, this contribution not only further strengthens the validity of ANODECs but marks an important step towards more practical, easy-to-use SRs that can automatically learn to perform agile motions from minimal experimental interaction time.

Background and author contributions: TS has proposed that the method developed in paper E is well suited for SR control. AlP has designed and built the experimental soft robot. ArP has conducted the experiments under the supervision of AlP and SB. TS and IB have supported AlP and SB in manuscript writing, structure, and formatting.

Paper C: RING: A Single Pluripotent Inertial Motion Tracking Solution

Paper C is the publication of

> Simon Bachhuber, Ive Weygers, Dustin Lehmann, Mischa Dombrowski, and Thomas Seel. Recurrent Inertial Graph-Based Estimator (RING): A Single Pluripotent Inertial Motion Tracking Solution. *Transactions on Machine Learning Research*, July 2024b.

Summary: This paper introduces a novel ML-based method for IMT, named RING, that provides a pluripotent, problem-unspecific plug-and-play IMT solution that, in contrast to conventional IMT solutions, eliminates the need for expert knowledge to identify, select, and parameterize the appropriate method.

Background and author contributions: SB has proposed the idea of a decentralized network of RNNs that estimates the entire rigid-body pose recursively, motivated by modern rigid-body dynamics algorithms such as, e.g., the Recursive Newton-Euler algorithm or the Composite-Rigid-Body algorithm (Featherstone, 2008). SB has proposed that this novel architecture presents the opportunity for a single pretrained ANN to be used for a broad range of IMT problems, including KCs with a varying number of segments. SB has implemented all algorithms. DL and SB have planned experiments. DL, TS, and SB have conducted experiments

and recorded real-world validation data. TS, IW, and MD have supported SB in manuscript writing, structure, and formatting.

3.2 Main Contributions

This section lists the contributions made by the publications listed in Section 3. This section also briefly discusses how these contributions complement the related work and how they address the research gaps defined in Section 2.2.2 and 2.3.2.

- Paper A addresses the Research Gap A and its contributions are twofold: First, paper A proposes a *novel* RNN-based method, titled RNNO, for assessing the observability property of simulated IMT problems. The concept of observability by example is defined, and observability is defined as a binary and *non-binary* system property, which includes degrees of observability. Finally, RNNO is used to reveal previously unknown results about the observability property in IMU-based motion tracking of a KC with two hinge joints, with known joint axes' directions, and with a sparse IMU setup (one IMU on each outer segment). Specifically, it is shown that the degree of observability decreases (i.e., the estimation problem becomes more challenging) as the directions of the joint axes align.

- Paper E addresses a subset of the Research Gap E by proposing a data-efficient learning control method, named ANODEC, that automatically designs output feedback controllers for finite-time RT in systems with unknown nonlinear dynamics. The approach assumes no prior knowledge of the system, neither in the form of an approximate system model nor state observability and even knowledge of the state's dimensionality is not required. In paper E, ANODEC is validated extensively in-silico using various simulated systems and reference signals.

- Paper B addresses a subset of the Research Gap C, and for the first time combines three out of the four IMT challenges (see Section 2.2.1) by enabling magnetometer-free, sparse, self-calibrating IMT of a double-hinge joint KC. Paper B proposes novel domain randomization techniques such that the in-simulation-trained RNNO of paper A *zero-shoot generalizes* to experimental, real-world data by successfully overcoming the sim-to-real gap. Paper B also constitutes the first experimental proof of concept of the unified approach; that is, it showcases the validity of the combination of in-simulation-trained RNNs with extensive domain randomization to solve real-world state estimation problems.

- Paper D addresses the Research Gap B and Research Gap C. By replacing RNNO with the more powerful RING in paper B, paper D can extend the results from paper B to IMU setups that are nonrigidly attached and that suffer from motion artifacts. Despite the nonrigid IMU attachment, RING can provide accurate orientation estimates and solves for the first time a IMT problem that combines all four IMT challenges (see Section 2.2.1).

- Paper F uses the ANODEC method as developed in paper E and applies it to an experimental SR setup. As such, the paper F's contributions are twofold: On the one hand, it provides experimental validation for ANODEC, and it also addresses the Research Gap F by enabling real-world pneumatic soft actuators to – fully automatically – learn to perform agile, non-repetitive motions, from only 30 seconds of experimental interaction time.

- Paper C addresses Research Gap D. It proposes a novel, online-capable ANN architecture, named RING, that, due to its unique architecture, allows for a single ANN to be trained on a broad range of IMT problems that vary greatly in aspects such as the number of attached sensors, or the number of segments in the KC. This architecture enables RING to provide a pluripotent, problem-unspecific plug-and-play IMT solution that, in contrast to conventional IMT solutions, eliminates the need for expert knowledge to identify, select, and parameterize the appropriate method.

4

Methods

"Using a simple tool to solve a complex problem does not result in a simple solution."
– Larry Wall, author of the Perl programming language

In the previous chapter, we listed the publications included in this thesis and their contributions. In this section, we will a) provide an overview and high-level understanding of the methods that have been developed in these publications (Section 4.2.2 and Section 4.3.2), and b) look at two selected methods in more technical detail (Section 4.2.3 and Section 4.3.3).

4.1 Background

This section provides some background information required to understand the unified problem formulations and solutions of the two example applications.

4.1.1 Partially Observable Markov Decision Processes

A MDP is a mathematical framework that models decision-making in environments where outcomes are partly random and partly controlled by a decision-maker, characterized by states, actions, transition probabilities, and rewards. POMDPs generalize MDPs by modeling the relationship between an agent and its environment, where the system dynamics are determined by a MDP, but the agent cannot directly observe the underlying state.

A discrete-time, finite-time POMDP $\mathcal{P}$ is characterized by a 9-tuple $(\mathcal{S}, \mathcal{S}_0, \mathcal{O}, \mathcal{A}, F, O, R, \gamma, T)$ where $\mathcal{S}$ is the space of all states, $\mathcal{S}_0$ is the initial state distribution, $\mathcal{O}$ is the space of all observations, $\mathcal{A}$ is the space of all actions, $F(s_{t+1}|s_t, a_t)$ is the transition probability (or function if the system is deterministic), $O(o_t|s_t, a_t)$ is the observation probability (typically a probability distributions due to sensor noise), $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor, and $T \in \mathbb{N}$ is the finite time horizon.

At each moment in time $t \in [0, T]$, the environment is in some state $s_t \in \mathcal{S}$. The agent takes an action $a_t \in \mathcal{A}$ which causes the environment to transition to state s_{t+1} with probability $F(s_{t+1}|s_t, a_t)$ and to emit a) the observation $o_t \in \mathcal{O}$ with probability $O(o_t|s_t, a_t)$, and b) the reward $r_t := R(s_t, a_t)$. This loop continues until the time horizon T is reached. The goal of the agent is

to maximize the expected, discounted, accumulated reward $\mathbb{E}_{\mathcal{P}}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, which is called the *return*.

In the two considered applications A and B, the discount factor γ is set to 1 since future tracking performance is just as important as current tracking performance. At the same time, the finite time horizon guarantees that the sum is bounded despite $\gamma = 1$.

The decision-making component in an agent is referred to as a policy. Unlike the policy function in MDPs, which maps the underlying states to the actions, a POMDP's policy maps the history of observations to the actions. In other words, for a given POMDP $\mathcal{P}$, a policy π is a map $a_t = \pi(o_{1:t})$. In, e.g. Deep RL, the policy is typically an ANN with parameters θ, i.e., $\pi = \pi_\theta$. The *optimal policy* π^* of $\mathcal{P}$ is the policy that maximizes the return, i.e., $\pi^* = \arg\max_{\pi} \mathbb{E}_{\mathcal{P},\pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$.

4.1.2 Recurrent Neural Networks

RNNs are a class of ANNs designed to recognize patterns in sequences of data such as text and language, or numerical time series data. Unlike feedforward ANNs, RNNs have loops, allowing information to persist. The policy in a POMDP maps the history of observations to the action, so the policy typically processes a time series of numerical data. This makes RNNs a natural choice for parameterizing a trainable policy. RNNs are trained with backpropagation through time. The features, variants, challenges, and solutions surrounding RNNs are outlined in detail in Appendix A.

4.1.3 Sim-to-real Transfer

Recall from Section 2.1 that the dilemma of sim-to-real transfer is designing a simulation such that its data distribution and in-silico trained solution works well on real-world data. As part of the sim-to-real transfer, we trade off less accurate modeling and a broad simulation environment p_{sim} by a) increasing the workload on the method or the ANN to be trained, and b) by reducing prediction performance since the ANN is forced to learn forms of online calibration which limits filter aggressiveness. Eq. (2.1) is directly applicable to the POMDP $\mathcal{P}$, with the following substitutions. In the context of $\mathcal{P}$, the loss function $\mathcal{L}$ in eq. (2.1) is given by the return, the ANN-based solution is given by the policy π_θ, and the two data distributions p_{sim} and p_{real} are indirectly given by the simulation environment $\mathcal{P}_{\text{sim}}$ and the real-world scenario $\mathcal{P}_{\text{real}}$ of $\mathcal{P}$. With these substitutions in place, the dilemma of sim-to-real transfer is the problem that we can only optimize $\mathbb{E}_{\mathcal{P}_{\text{sim}},\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$ even though we are interested in maximizing $\mathbb{E}_{\mathcal{P}_{\text{real}},\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$.

Thus, if the policy performs much worse in the real world than in the simulation, it follows from eq. (2.2) that this performance difference is due to little overlap of the simulation and real-world data distribution. Potential differences between $\mathcal{P}_{\text{sim}}$ and $\mathcal{P}_{\text{real}}$ are: 1) Gap in transition function F, e.g., the simulation model might not capture the real world well enough; 2) Gap in measurement function O, e.g., the real-world sensors might have different noise and bias properties; 3) Gap in state distribution $\mathcal{S}_0$, e.g., simulated and real-world motions might have different statistical patterns, e.g., random motion compared to human gait results in largely different distributions of joint angles.

4.1.4 A Unified Approach

The unified approach for state estimation and control of systems that perform dynamic motions is as follows. We will use POMDPs to formalize both motion analysis and motion control problems and allow for their forward simulations. Typically, POMDPs are used for motion control. However, motion analysis is also formulated as a POMDP to address motion analysis and motion control problems within a unified framework. Then, we will train policies in-silico with thousands of forward simulations with randomly drawn reference motions. We will use RNNs to parameterize the policy, which can handle partial observations. Domain randomizations are used to obtain robust, plug-and-play optimal policies that are transferred from simulation to reality. The entire framework is visualized in Figure 4.1.

4.2 Application A: Inertial Motion Tracking

This thesis uses IMT as an example application of a challenging state estimation problem involving systems that perform dynamic motions. IMT uses body-worn inertial sensors, called IMUs, to estimate and track the motion of bodies in space. For example, in Figure 1.7, IMT is used to track the motion of a human leg using two IMUs.

4.2.1 Problem Formulation

In order to apply the unified approach for IMT, we first formulate the problem as a POMDP. The POMDP $\mathcal{P}_{\text{IMT}}$ (or to emphasis $\mathcal{P}_{\text{IMT, sim}}$) models the task of IMT of an ARBS with N bodies and (at most) one IMU per body and is given by

- $\mathcal{S}$: The state s_t of an ARBS is the set of joint positions $\boldsymbol{q}_t$ and velocities $\dot{\boldsymbol{q}}_t$ at time t. The ARBS moves because of an external force or torque $\boldsymbol{\tau}_t$ that acts on its joints (its DOFs). To encapsulate a predetermined motion $\boldsymbol{q}^*_{1:T}$ in the state, we choose $s_t = \{\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{q}^*_{1:T}\}$ and compute torques using PD control.

- $\mathcal{S}_0$: The initial state is given by $s_0 = \{\boldsymbol{q}^*_1, \boldsymbol{0}, \boldsymbol{q}^*_{1:T}\}$ and subsequently a distribution is provided for reference motion trajectories $\boldsymbol{q}^*$. The RCMG (Random Chain Motion Generator), first introduced in paper A, is a function that allows to draw random reference trajectories for any ARBS.

- $\mathcal{O}$: The observation o_t in IMT is the set of all measurements of N IMUs at a moment in time, $o_t = \{\boldsymbol{\omega}_i(t), \boldsymbol{a}_i(t), \boldsymbol{m}_i(t), j_i(t) \mid \forall i\}$ where $\boldsymbol{\omega}_i(t)/\boldsymbol{a}_i(t)/\boldsymbol{m}_t(t)$ are 3D gyroscope / accelerometer / magnetometer measurements of IMU attached to body i at time t, and $j_i(t)$ is optional and symbolic for additional known quantities such as anatomical calibration parameters. In magnetometer-free IMT, the observation o_t does not contain magnetometer measurements $\boldsymbol{m}_i(t)$.

- $\mathcal{A}$: The action a_t in IMT is the agent's predicted pose of the ARBS, i.e., $a_t = \{\hat{\mathbf{q}}_i(t) \in \mathbb{H} \mid \forall i\}$. Note that the action (the predicted pose) does not influence the system since IMT is a motion analysis problem. The pose of the ARBS can be specified in various ways; here, we use one absolute and $N-1$ relative orientations. However, conversion to, e.g., N absolute orientations, is trivial given the graph of the ARBS.

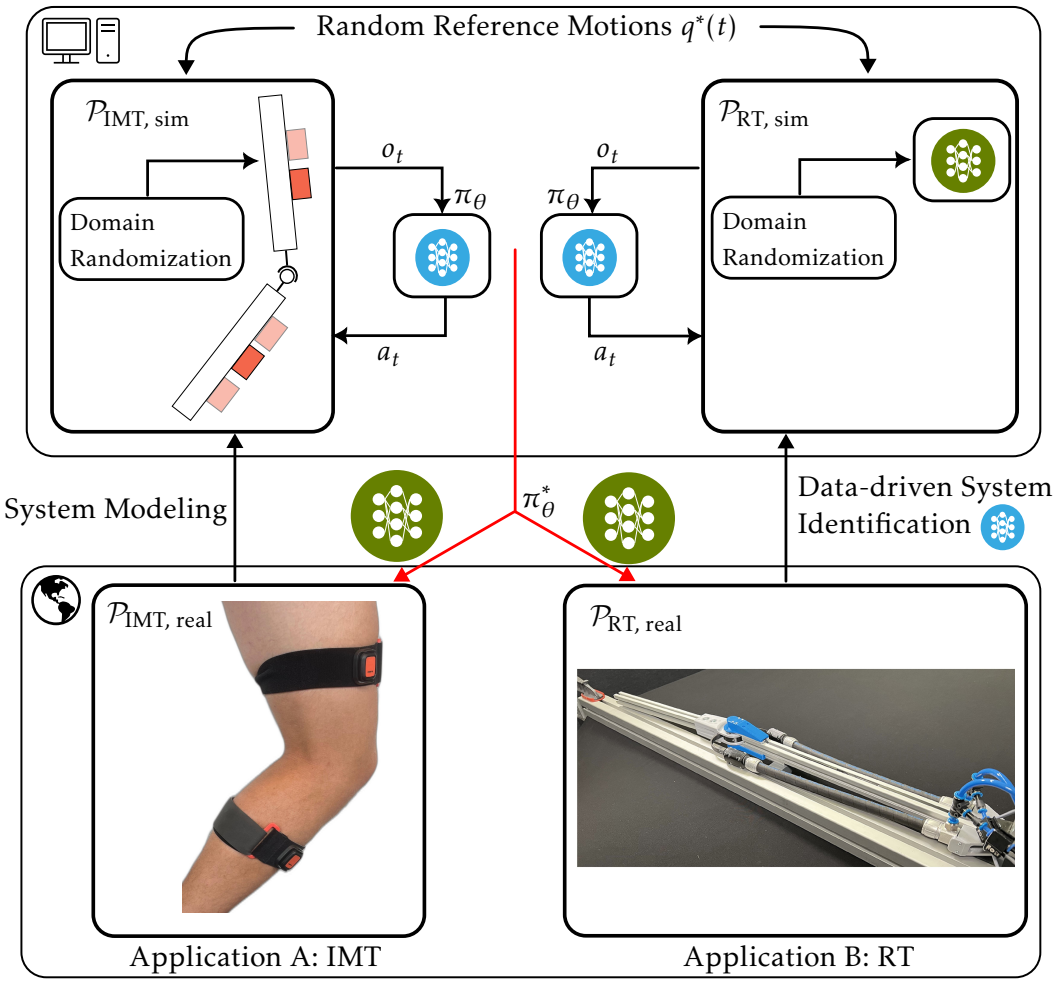


Figure 4.1: *A unified approach to state estimation and control of systems that perform dynamic motions. First, a suitable simulation environment of the real-world system in the form of a POMDP is obtained, either via system modeling, e.g., for IMT, or via data-driven system identification, e.g., for RT. Then, an RNN-based policy is trained in-silico by thousands of forward simulations of randomly drawn reference motions q^*. The simulation environment supports several domain randomizations to train a robust, plug-and-play applicable policy that afterward zero-shot generalizes and solves the real-world problem that performs any motion.*

- F: The transition function F is given by (bio)mechanical models. Typically, via forward dynamics $\ddot{\boldsymbol{q}}_t = \texttt{forDyn}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{\tau}_t)$ and subsequent integration, e.g., $\dot{\boldsymbol{q}}_{t+1} = \dot{\boldsymbol{q}}_t + \Delta t \ddot{\boldsymbol{q}}_t$.

- O: The observation probability O is given by an IMU model that computes accelerations, angular velocities, and magnetometer readings from the trajectory of Cartesian positions and orientations of each IMU, which can be obtained using forward kinematics. Additive, constant bias terms, and sensor noise are randomly sampled.

- R: Negative MSE (Mean Squared Error), i.e., the squared angle error between estimated pose $\hat{\mathbf{q}}_i(t)$ and ground truth pose $\mathbf{q}_i(t) \in \mathbb{H}$ average over all N bodies. It is given by

$$r_t(s_t, a_t) := -\frac{1}{N} \sum_{i=1}^{N} \texttt{angle}\left(\mathbf{q}_i(t) \otimes \hat{\mathbf{q}}_i(t)^*\right)^2 \tag{4.1}$$

  where $\otimes$ denotes quaternion multiplication, * denotes the complex conjugate, and $\texttt{angle}$ extracts the rotation angle, and is given by $\texttt{angle}(\mathbf{q}) := 2 \arctan\left(\frac{\sqrt{q_x^2+q_y^2+q_z^2}}{q_w}\right)$. The ground truth pose $\mathbf{q}_i(t)$ can be computed from the joint positions using forward kinematics $\{\mathbf{q}_i(t) \mid \forall i\} = \texttt{forKin}(\boldsymbol{q}_t)$.

4.2.2 Methods of Papers A, B, D, and C

The four papers A, B, D, and C all consider the POMDP $\mathcal{P}_{\text{IMT}}$. They estimate the expected return $\mathbb{E}\left[\sum_t r_t\right] \approx \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} r_t$ using $B \in \mathbb{N}$ sequences that are randomly drawn using the RCMG. The papers A and B use T set to one minute at 100 Hz, while the papers D and C use various time horizons and sampling rates. All four papers use an RNN-based policy π_θ and optimize the parameters by maximizing the return using gradient-based, first-order optimization. RNNs with two different architectures are used. In paper A and B, a deep GRU (Gated Recurrent Unit) network, titled RNNO, is used, and in paper D and C a greatly improved architecture, titled RING, is used.

However, the four papers are distinct. The differences are as follows. In paper A, $\mathcal{P}_{\text{IMT, sim}}$ is altered while increasing the complexity of π_θ in order to draw conclusions about the observability of the underlying POMDPs under ideal, in-silico conditions. A non-binary system-specific quantity, called degree of observability, is introduced, and three arguments for showing observability and the degree of observability are defined. In paper A, sim-to-real transfer is not considered.

In contrast, in the papers B, D, and C, the goal is to obtain a high-performance state estimator that can be applied to the *real-world*. I.e., achieving a high expected return for $\mathcal{P}_{\text{IMT, real}}$ is integral. To this end, several domain randomization techniques are introduced that ensure that an optimization of the expected return w.r.t. the simulated $\mathcal{P}_{\text{IMT}}$ yields a policy that generalizes to the real world. They are, for example, randomized superposition of motions (a form of randomizing the system's underlying graph to influence the system's forward kinematics computations) or random IMU placements. Figure 2.3 shows a subset of the used domain randomizations.

The paper D extends paper B by adding IMU motion artifacts and anatomical calibration into the state estimation problem of paper B that already is both magnetometer-free and sparse.

In paper C, a single policy for IMT of a KC with one, two, three, and four segments is trained. This pluripotency is enabled by the development of the novel, graph-aware ANN architecture RING.

Example 4.1

Consider IMT of an ARBS with two bodies that are interconnected by a hinge joint, and both bodies are attached with a 6D IMU that measure angular velocity $\boldsymbol{\omega}_{1/2}(t)$ and acceleration $\boldsymbol{a}_{1/2}(t)$. Then, the POMDP is given by

1. $s_t = \{\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{q}^*_{1:T}\}$ and $\mathcal{S} = \mathcal{S}_q \times \mathcal{S}_{\dot{q}} \times \mathcal{S}_q^T$ where the joint position vector $\boldsymbol{q}_t \in \mathcal{S}_q$ and joint velocity vector $\dot{\boldsymbol{q}}_t \in \mathcal{S}_{\dot{q}}$, and where $\mathcal{S}_q := \mathbb{H} \times \mathbb{R}^3 \times [-\pi, \pi)$ and $\mathcal{S}_{\dot{q}} := \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}$,
2. $o_t = \{\boldsymbol{\omega}_1(t), \boldsymbol{\omega}_2(t), \boldsymbol{a}_1(t), \boldsymbol{a}_2(t)\}$ and $\mathcal{O} \in \mathbb{R}^{12}$,
3. $a_t = \{\hat{\mathbf{q}}_1(t), \hat{\mathbf{q}}_2(t)\}$ where $\mathcal{A} \in \mathbb{H}^2$.

For example, $\boldsymbol{q}_t$ is given by a 3D orientation from base to body one ${}^0_1\mathbf{q}(t)$, a 3D vector from base to body one $\boldsymbol{r}_1(t)$, and an angle $\varphi(t)$ of the hinge joint that represents the orientation ${}^1_2\mathbf{q}(t)$.

The RCMG generates random motion by drawing joint position trajectories $\boldsymbol{q}^*_{1:T}$ while constraining the motion to certain upper and lower bounds on quantities such as, e.g., angular velocities or linear velocities. As a second step, it uses the transition function F in the form of a physical simulator and PD control to aggressively track the reference motion $\boldsymbol{q}^*$ and uses the observation function O in the form of an IMU model to record the IMU measurements. In this scenario, the randomized superposition of motion would result in drawing $\boldsymbol{q}^*_{1:T}$ such that segment two connects to the base, i.e., using ${}^0_2\mathbf{q}(t)$, $\boldsymbol{r}_2(t)$, and ${}^2_1\mathbf{q}(t)$. The random IMU placement domain randomization would place the IMUs with a randomly drawn offset vector on their respective segment, which enables the trained policy to be applicable without knowledge of the IMU placement. Such domain randomizations enable plug-and-play capabilities.

4.2.3 Selected Method: RING

RING is introduced in paper C, and it is a pluripotent plug-and-play IMT solution that eliminates the need for expert knowledge to identify, select, and tune the appropriate method. I.e., RING is a single policy π_θ that can be applied to a broad range of different POMDPs $\mathcal{P}_{\text{IMT}}$ that vary significantly in aspects such as the number of attached sensors, or the number of segments in the KC.

This pluripotency is enabled by its unique ANN architecture (see Figure 4.3) that uses a decentralized network of message-passing RNNs. Each segment (or body) in the ARBS is a node in a CG (Connectivity Graph). A CG is an undirected graph where the nodes represent the bodies that constitute the ARBS, and the edges represent its joints. The CG is often specified via a parent array $\boldsymbol{\lambda} \in \mathbb{N}^N$ where $\boldsymbol{\lambda}[i]$ is the body number of the parent of body i of an ARBS with $N \in \mathbb{N}$ bodies (Featherstone, 2008). At each node in the graph, the network observes the local IMU measurements and nearest-neighbor messages, and estimates the relative-to-parent orientation. For all bodies, the nearest-neighbor messages are the messages from the parent

[1] problem-unspecific

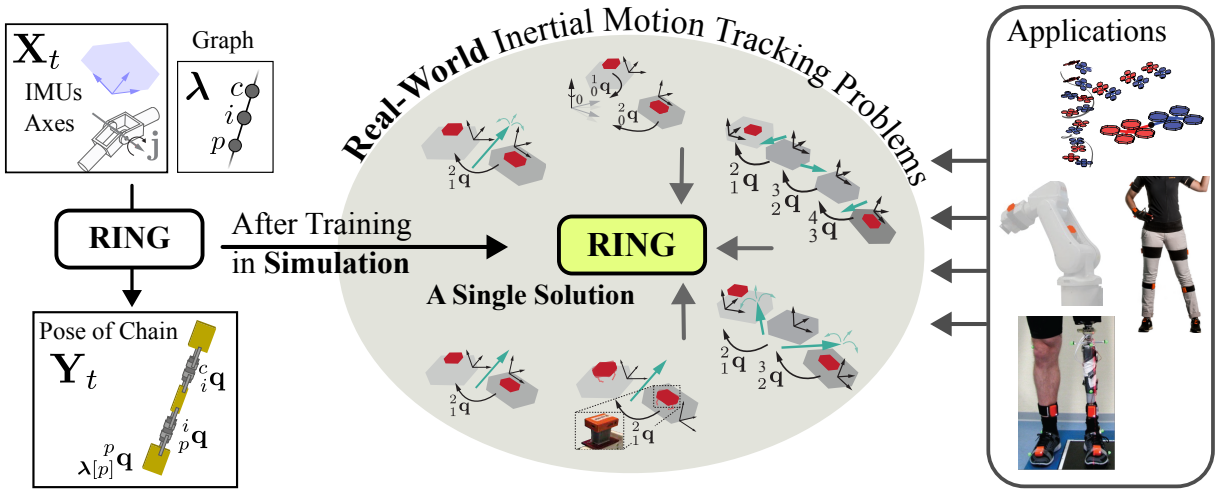


Figure 4.2: *RING is both an ANN architecture and, after training on simulated data generated by the RCMG, it provides a versatile, pluripotent[1] IMT solution applicable across a broad range of challenging problems, designed for use without the need for expert knowledge. Remarkably, RING is trained solely on simulated data, yet zero-shot generalizes to real-world experiments. Reprinted from Bachhuber et al. (2024b).*

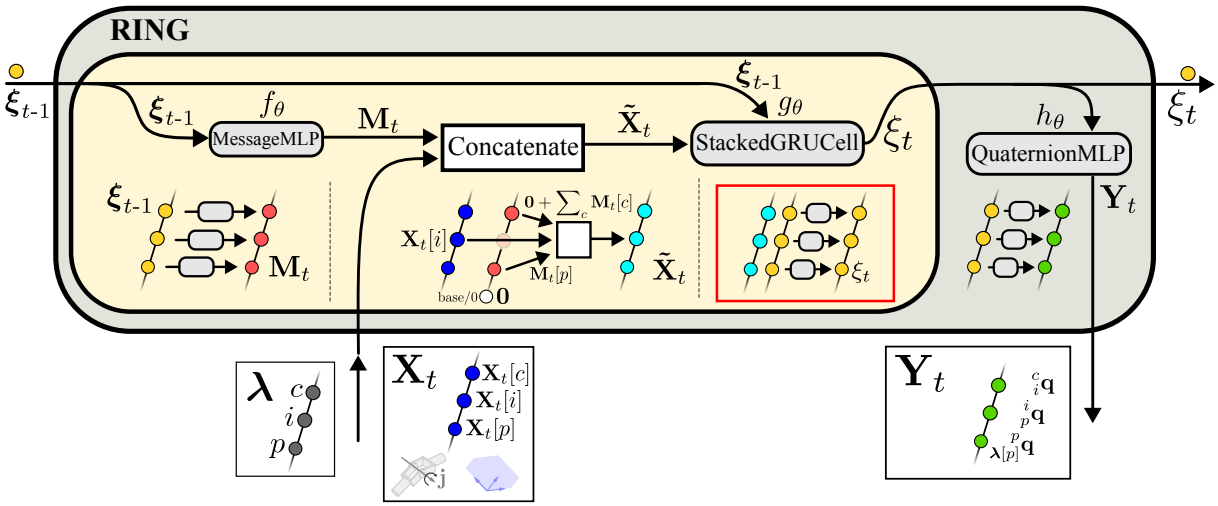


Figure 4.3: *The architecture of RING consists of a decentralized network (see, e.g., red box) of message-passing RNNs. The previous hidden state ξ_{t-1} and current IMU measurements $\mathbf{X}_t$ are processed by RING and the next hidden state ξ_t and current, predicted pose $\hat{\mathbf{Y}}_t$ of the ARBS is computed. Internally, a single set of parameters is applied independently for each body to send and receive messages to adjacent bodies, update the per-body hidden state, and compute and return the child-to-parent orientation. This design enables RING to be used with a single set of parameters for IMT of ARBSs with a varying number of bodies. Adapted from Bachhuber et al. (2024b).*

body and all child bodies. If the body has no child bodies or the parent body is the base[2], then these messages are set to zeros. Let $\boldsymbol{\lambda}$ be the CG of an ARBS with $N \in \mathbb{N}$ bodies, let $F \in \mathbb{N}$ be the number of input features per node, let $H \in \mathbb{N}$ be the half-hidden state dimensionality[3], and let $M \in \mathbb{N}$ be the dimensionality of the latent messages passed inside the cell based on the edges in the CG. Then, let $\boldsymbol{\xi}_{t\text{-}1} \in \mathbb{R}^{N\times 2H}$ be the hidden state of the RING cell from the previous timestep $t-1$, and let $\mathbf{X}_t \in \mathbb{R}^{N\times F}$ be the F inputs for all N bodies at time t. Then, the next hidden state $\boldsymbol{\xi}_t$ and prediction $\hat{\mathbf{Y}}_t \in \mathbb{H}^N$ is obtained by

$$\boldsymbol{\xi}_t, \hat{\mathbf{Y}}_t = \texttt{ring}(\boldsymbol{\xi}_{t\text{-}1}, \mathbf{X}_t, \boldsymbol{\lambda}) \quad \forall t \tag{4.2}$$

with $\boldsymbol{\xi}_0 = \mathbf{0}$.

RING has the parameters of a Message-MLP (Multi Layer Perceptron)-network $f_\theta : \mathbb{R}^H \to \mathbb{R}^M$, a stacked-GRU-network $g_\theta : \mathbb{R}^{2H} \times \mathbb{R}^{2M+F} \to \mathbb{R}^{2H}$, and a Quaternion-MLP $h_\theta : \mathbb{R}^H \to \mathbb{R}^4$. The step function `ring` has five consecutive steps, $\forall i = 1 \dots N$:

1. Messages $\mathbf{M}_t \in \mathbb{R}^{N\times M}$ are computed, i.e., $\mathbf{M}_t[i] = f_\theta\left(\boldsymbol{\xi}_{t\text{-}1}[i, H{:}]\right)$

2. Messages are passed and latent input $\tilde{\mathbf{X}} \in \mathbb{R}^{N\times(2M+F)}$ computed.

$$\tilde{\mathbf{X}}[i] = \texttt{concat}\left(\mathbf{M}_t\Big[\boldsymbol{\lambda}[i]\Big], \mathbf{0} + \sum_{c\in\mu_\lambda(i)} \mathbf{M}_t[c], \mathbf{X}_t[i]\right)$$

   where $\mathbf{M}_t\Big[\boldsymbol{\lambda}[i]\Big]$ is $\mathbf{0}$ if $\boldsymbol{\lambda}[i] = 0$ and $\mu_\lambda(i)$ returns the set of children of body i

3. Hidden state is updated, i.e., $\boldsymbol{\xi}_t[i] = g_\theta\left(\boldsymbol{\xi}_{t\text{-}1}[i], \tilde{\mathbf{X}}[i]\right)$

4. Unnormalized output $\tilde{\mathbf{Y}} \in \mathbb{R}^{N\times 4}$ is computed.

$$\tilde{\mathbf{Y}}[i] = h_\theta\Big(\texttt{layernorm}(\boldsymbol{\xi}_t[i, H{:}])\Big)$$

5. Normalize to allow interpretation as unit quaternions. One unit quaternion per node.

$$\hat{\mathbf{Y}}_t[i] = \frac{\tilde{\mathbf{Y}}[i]}{\sqrt{\sum_{j=1}^4 \tilde{\mathbf{Y}}[i,j]^2}}$$

The parameters θ are shared across all bodies. Hence, the set of parameters of RING is not influenced by the number of bodies. For example, a single RING network can be used for predicting the orientations of both two-segment or three-segment KCs even though the number of bodies and, consequently, the dimensionality of input and output arrays is different, e.g., $\mathbf{X} \in \mathbb{R}^{T\times 2\times F}$ compared to $\mathbf{X} \in \mathbb{R}^{T\times 3\times F}$.

[2]The base is assigned the number 0 and serves as the root node in the CG

[3]The state of two GRU cells each with hidden state dimensionality H is combined into one hidden state of dimensionality $2H$

4.3 Application B: Reference Tracking in Unknown Nonlinear Dynamics

This thesis uses RT in unknown nonlinear dynamics as the second example application for motion analysis and motion control of systems performing dynamic motions. Finite-time horizon RT is the task of controlling the output of a system such that it follows a desired reference signal for a finite duration of time.

4.3.1 Problem Formulation

In order to apply the unified approach for RT, we first formulate the problem as a POMDP. The POMDP $\mathcal{P}_{\text{RT}}$ (or to emphasize $\mathcal{P}_{\text{RT, real}}$) models the task of manipulating $\boldsymbol{u}$ such that the system output $\boldsymbol{y}$ tracks a reference motion $\boldsymbol{y}^*$ in a dynamical system given by Ψ (with unknown dynamics, see Section 2.3). It is given by:

- $\mathcal{S}$: $s_t = \{\boldsymbol{\xi}_t, \boldsymbol{y}^*_{1:T}\}$ where $\boldsymbol{\xi}_t$ is the (unknown) state of the dynamical system, and $\boldsymbol{y}^*$ is a trial-varying and unknown reference motion, also recall that the state is not observed.
- $\mathcal{S}_0$: We assume the existence of an unknown, trial-invariant initial state $\boldsymbol{\xi}_0$. The predetermined reference motions can be, e.g., randomly drawn step functions or smooth curves.
- $\mathcal{O}$: $o_t = \{\boldsymbol{y}_t, \boldsymbol{y}^*_t\}$.
- $\mathcal{A}$: $a_t = \boldsymbol{u}_t$.
- F and O are the unknown dynamics and measurement functions; they are provided by the dynamical system Ψ.
- R: Negative squared-(tracking)-error-norm, i.e., $r_t(s_t) := \left|\boldsymbol{y}_t - \boldsymbol{y}^*_t\right|_2^2$.

4.3.2 Methods of Paper E and F

In principle, we can solve $\mathcal{P}_{\text{RT, real}}$ using two approaches. The first approach solves it directly in a model-free approach. This would involve applying the policy π_θ and a (small set of) reference motions $\boldsymbol{y}^*_{1:T}$ to the real-world system and estimating the expected reward. Then, using, e.g., Deep RL, we can optimize the policy's parameters (gradient-free RL since $\frac{dr_t}{d\theta}$ is not available). Unfortunately, such a model-free approach has the disadvantage that it is unsafe and typically data-inefficient.

Instead, paper E proposes a model-based approach titled ANODEC. First, ANODEC builds a simulation environment $\mathcal{P}_{\text{RT, sim}}$ by probing the system for input-output data and then training a differentiable, approximate system model. ANODEC uses NODEs (see Appendix A.3.1) for the system model. Then, ANODEC uses first-order optimization (because $\frac{dr_t}{d\theta}$ is available since neural ODEs are differentiable) to train a policy (or controller) that solves $\mathcal{P}_{\text{RT, sim}}$ by using a (large set of) reference motions to estimate the expected reward. ANODEC uses neural ODEs for the policy as well. The trained policy is then directly transferred to the real world. The disadvantage of the model-based approach is that it relies on a successful sim-to-real transfer. In paper E, ANODEC is validated extensively in simulation. In paper F, ANODEC is used to enable a soft robotic system to automatically and from only 30 s of input-output data learn to perform precise, agile motions.

4.3.3 Selected Method: ANODEC

ANODEC is a model-based learning control approach for output RT in systems with unknown nonlinear dynamics. It is introduced in paper E. First, ANODEC builds a simulation environment $\mathcal{P}_{\text{RT, sim}}$. In this step, it approximates the real-world dynamics Ψ by a NODE that is learned from $N \in \mathbb{N}$ experimental input-output data pairs $\{(\boldsymbol{u}_i(t), \boldsymbol{y}_i(t)) \mid \forall i\}$. The state of the dynamical system Ψ is not observed. The neural ODE that approximates Ψ is given by

$$\begin{aligned} \frac{d\boldsymbol{\xi}^{(m)}(t)}{dt} &= f_\theta^{(m)}\left(\boldsymbol{\xi}^{(m)}(t), \boldsymbol{u}(t)\right), \\ \hat{\boldsymbol{y}}(t) &= g_\theta^{(m)}\left(\boldsymbol{\xi}^{(m)}(t)\right), \end{aligned}$$

where $f_\theta^{(m)}$ and $g_\theta^{(m)}$ are MLPs, $u(t) \in \mathbb{R}^p$ the network input, $\hat{\boldsymbol{y}}(t) \in \mathbb{R}^q$ the network output, and $\boldsymbol{\xi}^{(m)}$ is the latent state vector in which the NODE model evolves (dimensionality is a hyperparameter). We denote the vector of all parameters of $f^{(m)}, g^{(m)}$ by $\theta^{(m)}$ and use supervised learning to optimize these parameters

$$\theta^{(m)^*} = \arg\min_{\theta^{(m)}} \mathop{\mathbb{E}}_{(u,y)} \left[\int_0^T \left\| \boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t) \right\|_2 dt \right].$$

As the second step, ANODEC learns the policy $\pi_{\theta^{(c)}}$. It uses the trained model with frozen parameters $\theta^{(m)^*}$ to cheaply simulate the closed-loop of the trained model and policy. During forward simulation, domain randomization techniques are used to robustify the policy, e.g., by simulating various output noise levels or adding dropout layers into the model architecture, thus incorporating stochasticity into the simulation. The policy $\boldsymbol{u}_t = \pi_{\theta^{(c)}}(o_{1:t})$ is given by the NODE

$$\begin{aligned} \frac{d\boldsymbol{\xi}^{(c)}(t)}{dt} &= f_\theta^{(c)}\left(\boldsymbol{\xi}^{(c)}(t), \boldsymbol{y}^*(t), \boldsymbol{y}(t)\right), \\ \boldsymbol{u}(t) &= g_\theta^{(c)}\left(\boldsymbol{\xi}^{(c)}(t)\right), \end{aligned}$$

where $f_\theta^{(c)}$ and $g_\theta^{(c)}$ are MLPs and $\boldsymbol{\xi}^{(c)}$ is the latent state vector (dimensionality is a hyperparameter; typically slightly smaller than the dimensionality of the NODE model). Then, ANODEC closes the loop between $\mathcal{P}_{\text{RT, sim}}$ and policy and optimizes the expected return with an additional regularization term proportional to $\lambda^{(c)}$, i.e.,

$$\theta^{(c)^*} = \arg\min_{\theta^{(c)}} \left(\lambda^{(c)} \left\| \theta^{(c)} \right\|_2 + \mathop{\mathbb{E}}_{\mathcal{P}_{\text{RT, sim}}, \pi_{\theta^{(c)}}} \left[\sum_t r_t(s_t) \right] \right).$$

As a final step, the trained policy $\pi_{\theta^{(c)^*}}$ is applied in the real-world system $\mathcal{P}_{\text{RT, real}}$.

5

Results and Discussion

"What can be asserted without evidence can be dismissed without evidence."
– Christopher Hitchens, American writer

This chapter summarizes the major results of the papers included in this thesis (see Section 3). Individual results are discussed in Section 5.1 and Section 5.2. Section 5.3 jointly discusses sets of results of all papers.

5.1 Application A: Inertial Motion Tracking

Four of the six papers contribute to the IMT field. Their results are listed below.

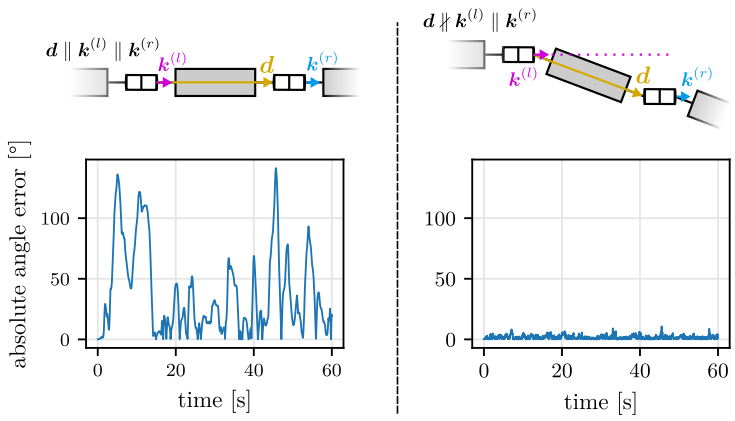


Figure 5.1: *Absolute angle error of the relative orientation estimate between two segments in a double-hinge-joint, three-segment KC. (Left side) The error spikes to large values due to non-observability, whereas (right side) the trained RNN-based observer (RNNO) accurately tracks the motion at all times for the observable case. Reprinted from Bachhuber et al. (2022).*

5.1.1 Paper A: Observability Analysis

Observability (and controllability) are essential prerequisites for successful state estimation and control. However, the strict definitions of control theory (Bechhoefer, 2005) are often not applicable and thus of limited use. Instead, definitions by example can be more useful (Brunton and Kutz, 2022). Observability by example states that a dynamical system is considered observable if, for any input trajectory and initial condition, an observer can reliably estimate the target state variable from past measurements, see Definition B.2.

Paper A first proposes a ML-based method for assessing observability by example, titled RNNO. It combines ANNs and their universal function approximation guarantees with useful conjectures[1] to draw conclusions about observability under ideal, in-silico conditions.

Paper A then applies the proposed method to assess observability of IMT. It is shown that for magnetometer-free and sparse IMT of a double-hinge-joint three-segment KC, the observability decreases as the two hinge joint axes' directions align. This observation follows from the three proposed conjectures (see paper A) combined with the evidence that the training of the RNNO saturates to a high error value regardless of network complexity for the case of parallel joint axes' directions due to sudden error spikes as observed in Figure 5.1. Thus, the realization of the system with parallel joint axes' directions is non-observable. Moreover, the realization with non-parallel joint axes' directions is observable since the error decreases as network complexity is increased. Albeit not to the same extent as a system realization with orthogonal joint axes' directions. Thus, the latter realization has a higher degree of observability, as observability is defined by the conjectures from paper A. Finally, it is shown that the three-segment KC with a biaxial joint and hinge joint is observable if sensor-to-segment parameters are known.

5.1.2 Paper B: Real-world Sparse and Magnetometer-free IMT

Paper B focuses on sim-to-real-transfer. In paper A, we have shown that IMT of a double-hinge-joint, three-segment KC with a sparse set of magnetometer-free IMUs has a high degree of observability as long as the two joint axes are not well aligned. In paper B, we show that the in-simulation trained RNNO can zero-short generalize to real-world data and enable accurate tracking of the three-segment KC from only two magnetometer-free IMUs with a MAE of < 4 degrees. It is also shown that the proposed domain randomization techniques are crucially important for successful sim-to-real transfer (see Table 5.1). The MAE of RNNO is low in all simulations $\mathcal{P}_{\text{IMT, sim}}$, but only with the domain randomization is it also low in $\mathcal{P}_{\text{IMT, real}}$.

Paper B also shows that RNNO consistently and quickly converges within two seconds while enabling accurate motion tracking post-convergence. Moreover, the paper shows that RNNO is long-term stable. Both latter properties are essential for real-world applicability of RNNO.

5.1.3 Paper C: A Single, Pluripotent IMT Solution

Paper C focuses on developing a new and improved ANN architecture for IMT applications, titled RING. This new architecture aims to be broadly applicable (pluripotent), offering an easy-to-use, plug-and-play solution for a wide range of IMT problems. The proposed solution

[1] For example, conjecture non-observability: *If $\mathcal{P}_{sim}$ is non-observable, then the proposed RNNO cannot converge to low residual error, even if the amount of training data is increased, even if the parameter count of the RNN is increased, and even if the noise and bias levels are reduced.*

Table 5.1: *Comparison of RNNO's orientation estimation accuracy when trained on simulated data with or without various domain randomizations. While the MAE in simulation is low in all scenarios, domain randomization techniques are crucially important for the trained RNNO to overcome the sim-to-real gap and achieve a low, real-world MAE. Result from paper B. Training in $\mathcal{P}_{IMT,\,sim}$, MAE evaluated in $\mathcal{P}_{IMT,\,real}$. Adapted from Bachhuber et al. (2023a).*

DR 1 (RoM)	DR 2(RFK)	DR 3 (RG)	MAE [deg]
✗	✗	✗	73.6 ± 27.0
✓	✓	✓	3.25 ± 0.29

Column acronyms: Range of Motion (RoM), Randomized Forward Kinematics (RFK), Randomized Geometry (RG)

RING is validated experimentally on real-world motion of a 3D-printed five-segment KC. In total, ten IMUs are attached to the KC, and ground truth data is recorded via OMC. To validate RING's ability to compensate motion artifacts, each segment of the five-segment KC is equipped with two IMUs, one rigidly- and one foam-attached. The foam attachment introduces motion artifacts. The segments of the KC are interconnected by 1D, 2D, and 3D joints, offering a broad spectrum of IMT problems. The experimental validation shows that RING achieves similar performance (and sometimes outperforms) nine problem-specific SOTA methods across a variety of IMT problems (see Table 5.2). Apart from unifying a large body of previous work with a single, large ANN, RING also enables several novel, for-the-first-time applications:

- RING can track a three-segment, double-hinge-joint KC with unknown hinge joint axes' directions from a sparse set of two magnetometer-free IMUs with an experimental MAE of 5.37 ± 0.71 degrees.
- RING can track a four-segment, triple-hinge-joint KC with known joint axes' directions from a sparse set of two magnetometer-free IMUs with an experimental MAE of 6.78 ± 1.41.
- RING can successfully mitigate IMU motion artifacts due to nonrigidly-attached IMUs. For example, RING can track the segment-to-segment orientation of a hinge joint with known axis direction from two foam-attached IMUs with an experimental MAE of 5.56 ± 2.33 degrees.

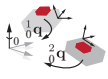
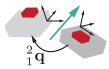
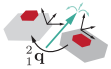
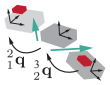
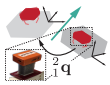
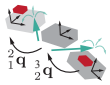
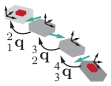
In addition to remarkable estimation performance, robustness to differing noise and bias levels of IMUs is shown in simulation. Moreover, an experimental validation on datasets involving various IMU manufacturers shows consistent estimation performance and further demonstrates RING's robustness to IMUs with different noise and bias properties.

Paper C also shows that the decentralized approach of RING provides an advantageous, structural prior that aids network training compared to more typical, centralized approaches such as, e.g., RNNO. This is the case even if only the solution of a single, specific IMT problem is required.

All code[2] and data[3] is made openly available.

[2] `https://github.com/SimiPixel/ring`
[3] `https://github.com/SimiPixel/diodem`

Table 5.2: *Motion tracking accuracy (in degrees) of RING compared to various SOTA methods across a variety of IMT problems. While previous methods are problem-specific and Not Applicable (NA) to many IMT problems, RING is the only method that accurately solves all problems. The table contains results for eight IMT problems and each problem is characterized by an image that shows the segments (gray boxes), attached IMUs (orange boxes), whether or not the IMU is foam-attached (image of foam), DOF of joints (green arrows, here all joints are 1D), whether or not the joints' axes directions are known (small green arrows), and the orientation estimation target as quaternions. Adapted from Bachhuber et al. (2024b).*

Problems	1)	2)	3)	4)	5)	6)	7)	8)
Method								
(1)	2.06 ± 1.03	NA	NA	NA	≥(5)	NA	NA	NA
(2)	2.25 ± 0.81	≥(5)	≥(5)	NA	≥(5)	NA	NA	NA
(3)	2.09 ± 0.87	≥(5)	≥(5)	NA	≥(5)	NA	NA	NA
(4)	2.56 ± 0.93	≥(5)	≥(5)	NA	≥(5)	NA	NA	NA
(5)	1.61 ± 1.04	→	19.3 ± 8.02	NA	9.20 ± 2.31	24.9 ± 17.6	NA	NA
(5)+(6)	↑	3.32 ± 2.12	NA	NA	↑	7.00 ± 1.57	NA	NA
(5)+(7)	↑	4.15 ± 2.05	NA	NA	↑	8.00 ± 2.78	NA	NA
(5)+(6)+(8)	↑	→	3.18 ± 2.05	NA	↑	8.50 ± 2.60	NA	NA
(5)+(7)+(8)	↑	→	4.06 ± 2.23	NA	↑	7.90 ± 2.48	NA	NA
(9)	NA	NA	NA	5.60 ± 2.35	NA	NA	NA	NA
RING	2.13 ± 0.91	3.52 ± 1.00	3.92 ± 1.40	4.14 ± 0.53	7.59 ± 2.85	5.56 ± 2.33	5.37 ± 0.71	6.78 ± 1.41

Methods: Weber et al. (2021)(1), Madgwick (2010)(2), Mahony et al. (2008)(3), Seel and Ruppin (2017)(4), Laidig and Seel (2023)(5), Laidig et al. (2017)(6), Lehmann et al. (2020b)(7),Olsson et al. (2020)(8), Bachhuber et al. (2023a)(9)
≥ (*i*) refers to the MAE of being expected to be larger or equal than for method (*i*)
↑ or → indicate that the MAE is equal to the MAE of the cell above or to the cell to the right

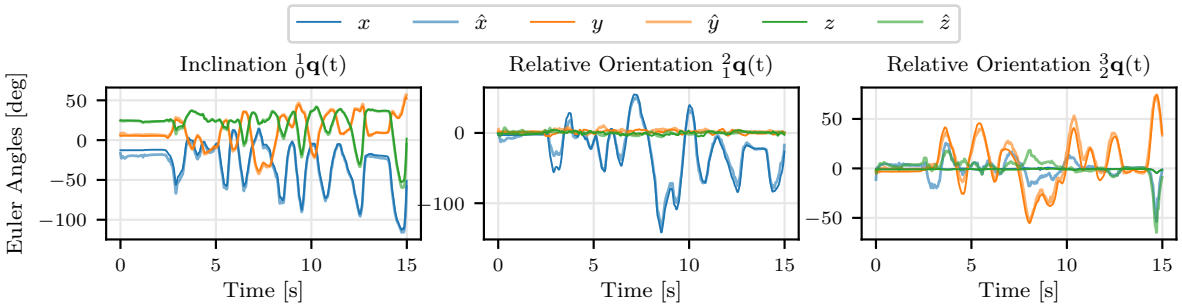


Figure 5.2: *Visualization of ground truth orientations (thin, solid lines) and RING's predicted orientation (thick, opaque lines) for one experimental example sequence. The example sequence contains the first* 15 s *of a IMT problem involving a double hinge joint KC with joint axes' directions in x− and y−direction. RING estimates the orientations from sparse, magnetometer-free, nonrigidly attached IMUs and without joint axes direction. Reprinted from Bachhuber et al. (2024c).*

5.1.4 Paper D: Dispelling Four IMT Challenges

Paper D shows that RING can overcome the four main challenges of IMT simultaneously. The four challenges (see Section 2.2.1) are: 1) dealing with inhomogeneous magnetic fields distorting magnetometer readings, addressing heading offset due to magnetometer-free orientation estimation, managing sparse sensor setups with fewer sensors than segments, and compensating for motion artifacts caused by skin-attached IMUs.

Paper D demonstrates that RING can solve an IMT problem that combines all four challenges. The paper considers tracking the motion of a three-segment double-hinge-joint KC from two magnetometer-free IMUs attached via foam to the segments. The hinge joint axes' directions are not known. Instead, we rely on RING's auto-calibration features. Despite this complexity, RING achieves an experimental MAE of ≈ 8 degrees. An example sequence is shown in Figure 5.2.

Paper D also shows that RING is real-time capable of up to ≈ 1000 Hz on desktop-grade hardware and that it provides consistent estimations across a broad range of sampling rates (60 Hz to 200 Hz). Estimation accuracy degrades only notably at sampling rates below 60 Hz.

5.2 Application B: Reference Tracking in Unknown Nonlinear Dynamics

Two of the six papers contribute to the RT field. Their results are listed below.

5.2.1 Paper E: Development and Extensive In-Silico Validation of ANODEC

Paper E introduces ANODEC, a novel model-based learning control method for output RT in systems with unknown nonlinear dynamics. The method was evaluated across four simulated systems and 500 reference motions sourced from five distinct distributions. In nearly all cases, ANODEC consistently outperforms two classical control methods, achieving an average of approximately $\approx 30\%/17\%$ lower median tracking RMSE (see Figure 5.3).

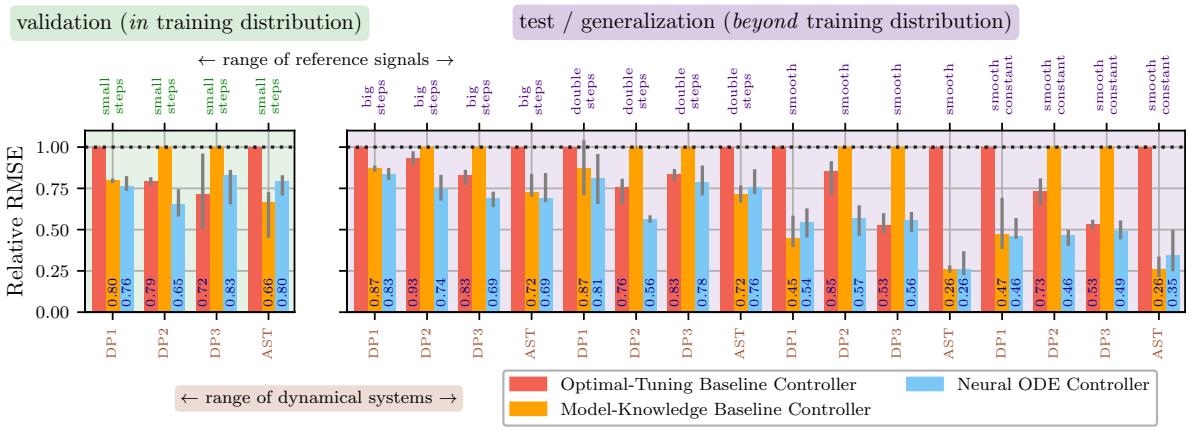


Figure 5.3: *ANODEC's RT performance relative to the optimal-tuning controller baseline and model-knowledge controller baseline on all combinations of four systems and five reference signal distributions. For each combination of system and reference source, the 25%/50%/75%–percentiles are obtained from 100 randomly drawn references from the distribution. ANODEC achieves an on average ≈ 30 % / 17 % / 7 % / 40 % lower median RMSE across all validation and test combinations compared to the optimal-tuning controller baseline / model-knowledge controller baseline / the best out of the two baselines / the worst out of the two baselines. Reprinted from Bachhuber et al. (2023b).*

ANODEC requires only 150 s seconds of input-output data for double-pendulum systems and 270 s seconds for the other cases, without needing any state information. To address concerns regarding the generalizability of ML-based solutions, we evaluated the model using 400 reference motions outside the training distribution, including double steps and smooth curves.

The four simulated systems include three spring-damper systems, double-pendulum dynamics, and a system with Ackermann steering. The two baseline control methods consist of 1) an optimally tuned PID controller and 2) a model-knowledge-based controller that assumes perfect knowledge of the system's local linearization, utilizing pole placement and grid search to design a fifth-order transfer function controller.

Notably, ANODEC reduces tracking errors, effectively dampens oscillations, and achieves a more controlled and faster response to reference steps compared to the baseline methods. A video demonstration of ANODEC's performance is available online[4], and the code is openly accessible[5].

5.2.2 Paper F: Experimental Validation using a Soft Robotic System

Paper F uses ANODEC and applies it to an experimental SR system that consists of an arm with two PAMs. The experimental system is considered in two different setups (see Figure 5.4), and for both ANODEC successfully learns to perform agile, non-repetitive motions. It achieves this while requiring only a parsimonious amount of input-output data. As shown in Table 5.2, ANODEC consistently outperforms a manually tuned PID controller baseline across three reference motion distributions. For example, for the experimental Setup 1, ANODEC achieves an

[4] https://youtube.com/watch?v=tttkFFD81Qw
[5] https://github.com/SimiPixel/chain_control

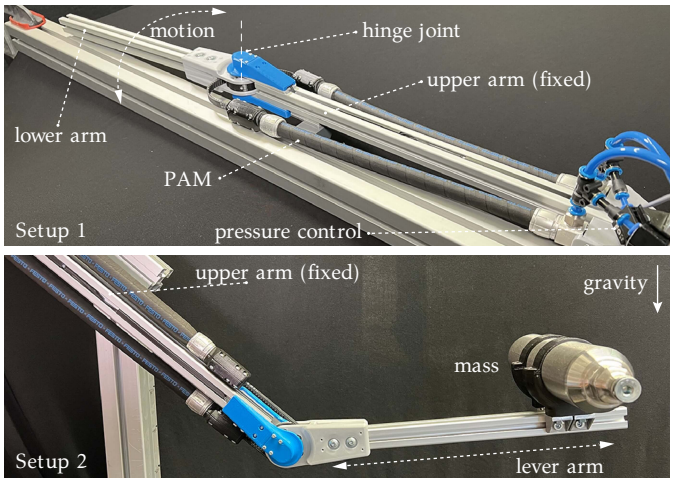


Figure 5.4: *Two experimental setups of a pneumatic arm with a single DOF. Two PAMs (black tubes) are used as an antagonistic pair to control the arm's forces and position in both setups. The upper setup shows the simplest configuration without the influence of gravity and external load. The lower setup is loaded with an external weight of* 0.6 kg*, with a lever arm of* 0.25 m *oriented against gravity. Both arms are fixed to the ground to prevent undesired movements. Reprinted from Bachhuber et al. (2024a).*

approximately 45.9% lower RMSE tracking error on average, requiring only 30 seconds of experimental input-output data. Moreover, ANODEC designs robust and stable controllers, even under heavily disturbed trials, as illustrated in Figure 5.5.

5.3 Joint Discussion of Sets of Publications

"The whole is greater than the sum of its parts."
– Aristotle, Greek Philosopher

This section presents claims and evidence that become only apparent when considering multiple papers simultaneously.

On the cost of Plug-and-play Applicability

The broader applicability of RING over RNNO for challenging IMT problems comes at the cost of increased training workload and a performance hit, particularly in terms of slower initial convergence and occasional instability in predictions.

RNNO enables real-world sparse IMT but RING is a superior solution because it outperforms RNNO on the same task (see Table 5.2) and because it is applicable to even more challenging IMT problems (see Table 5.2 and paper D). According to Section 4.1.3, the broader applicability of RING compared to RNNO incurs the cost of a) an increased workload on the ANN to be trained, and b) a performance hit due to the requirement of implicitly learning forms of online calibration that enable plug-and-play capabilities in the first place. a) is acceptable and mainly increases the technical requirements for successful network training. This is evident from the fact that, e.g., in paper B, the training of RNNO is finished after $\approx$ 750 epochs, whereas in paper C the training of RING is finished after $\approx$ 5000 epochs. b) is unfortunate, and it manifests in the following two behaviors. First, the time for initial convergence of RING is higher than for RNNO. RNNO consistently converges to low errors within the first two seconds, whereas RING occasionally requires up to ten seconds before its predictions are stable and accurate. Second,

Table 5.3: *RT RMSE (in degrees) of PID and ANODEC for three reference signal distributions and two systems: Steps, double steps, and cubic splines. For each distribution, N distinct reference signals are drawn and used to estimate the RMSE and one standard deviation. Setup 1 uses the pneumatic arm as shown in Figure 2.7 (right side). In Setup 2, the pneumatic arm is loaded with an external weight on the lower arm and rotated by 90 degrees to be oriented against gravity. Reprinted from Bachhuber et al. (2024a).*

References	PID	ANODEC
Setup 1		
Steps ($N = 2$)	12.89 ± 6.14	10.03 ± 4.93
Double Steps ($N = 2$)	12.86 ± 2.66	11.08 ± 1.37
Cubic Splines ($N = 12$)	10.01 ± 1.44	4.48 ± 1.11
Setup 2		
Steps ($N = 2$)	4.00 ± 0.45	5.54 ± 0.53
Double Steps ($N = 2$)	9.50 ± 4.03	6.81 ± 0.75
Cubic Splines ($N = 4$)	4.56 ± 0.81	5.01 ± 0.79

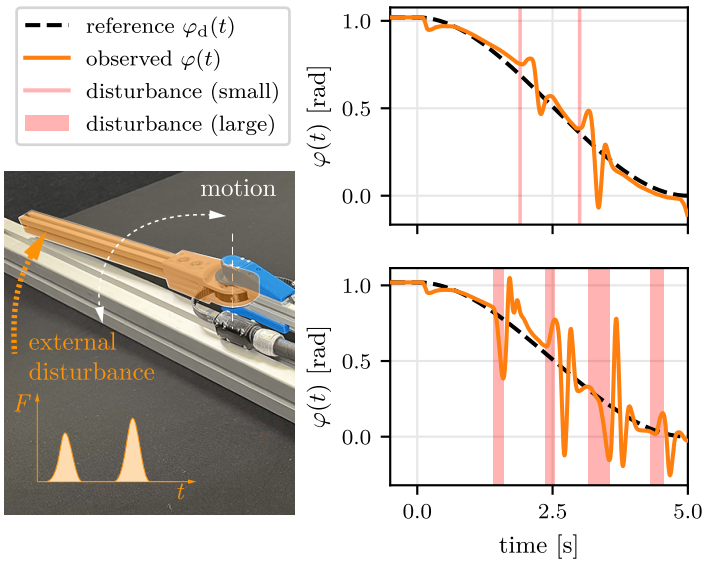


Figure 5.5: *Tracking performance of ANODEC in two disturbed trials. In the first trial (top), there are two small disturbances (poking the end effector with a stick), whereas in the second trial (bottom), there are four large disturbances (grabbing and briefly holding the end effector). ANODEC designs feedback controllers that remain stable and reject external disturbances. Reprinted from Bachhuber et al. (2024a).*

the prediction of RNNO is more stable and less nervous. RING 's predictions can occasionally drift to wrong estimates (see, e.g., Figure 5.2 right subplot). The author hypothesizes that the combination of short-term, smooth GRU cells with a data stream that can become temporarily non-observable, i.e., due to the broad spectrum of training scenarios, there exist two scenarios with a locally (w.r.t. the specific motion) identical measurement data stream yet two different and competing prediction data streams. The usage of LSTM (Long Short Term Memory) cells or an alternative Transformer architecture inside of the RING Cells (see paper C) might alleviate the latter issue. A similar issue has also been observed in paper F where the even more short-termed, alternative RNN architecture of NODEs can struggle with keeping the hysteresis effects of the SR in its internal memory.

Random Reference Motions and Proactive Versus Reactive Behavior

For both IMT and RT, the trained policy maximizes the expected reward based on randomized reference motions. IMT requires multiple reference motion distributions for optimal transfer, and proactive behavior is undesirable for state estimation due to its non-influence on the system. In contrast, RT can generalize from only step functions, where proactive behavior benefits control tasks.

For both IMT and RT, the trained policy (RNNO, RING, or ANODEC) maximizes the expected reward, where the expectation includes the reference motions that are given via trajectories $\boldsymbol{q}^*_{1:T}$ or $\boldsymbol{y}^*_{1:T}$ for IMT or RT, respectively. Therefore, in order to obtain a trained policy that is real-world applicable, the reference motion must be either known a-priori (e.g., ILC assumes a known reference signal) or, according to Section 4.1.3, the reference motion must be randomized. Interestingly, for state estimation and IMT, the RCMG generates a broad range of reference motions to include qualitatively identical to the unknown, real-world motions. This is also evident by the fact that for the training of RING in total four different reference motion *distributions* are needed (and a single distribution does not suffice) for optimal transfer (see paper C). On the other hand, for control and RT, a random set of non-physical step functions are sufficient for obtaining a trained policy that generalizes to, e.g., smooth reference motions, as can be observed in Figure 5.3. In fact, for control, the reference motions provided as step functions offer a clear pattern that allows for, in addition to reactive feedback control, learning of proactive behavior in the form of a dynamic feedforward signal. This can be seen in, e.g., ANODEC's behavior for double steps reference in the double-pendulum system (see Section 5.2.1).

In contrast, proactive behavior is undesirable for IMT since the action does not influence the system. Hence, proactive behavior can only be enabled by statistical patterns such as, e.g., typical human motion patterns. To some extent, this may also be attributed to the cost of plug-and-play applicability where we desire a solution that requires no prior knowledge of the types of motions to be performed, i.e., that is not restricted to, e.g., human gait motion.

Smooth Problems and Smooth Solutions

Unsurprisingly, smooth problems can be solved with smooth solutions.

In the absence of discontinuous events, e.g., collisions, both IMT and RT are smooth problems. By smooth, it is meant that the dynamics are continuous and Lipschitz-bounded. IMT involves ARBS, which have smooth dynamics, and all system dynamics under consideration for RT are smooth. Similarly, this thesis's proposed policies (or solutions) are smooth. RNNO and RING are RNN-based, and ANODEC is Neural-ODE-based. Both have smooth dynamics (see

Appendix A). However, while this smoothness is an advantage for ANN training, it is not an advantage for the trained network and its expressivity. First, problems with non-smooth dynamics (or hybrid dynamics) can only be sub-optimally solved by strictly smooth solutions. They are inert and cannot react quickly enough to changes. Second, even if the problem is smooth, a non-smooth solution can have advantages for online calibration and learning of hard constraints, e.g., in IMT, two rigid-body segments must always be connected. The latter can only be modeled as soft constraints if the solution offers only smooth dynamics. It is the author's second conjectured explanation for the sub-optimal behavior of RING discussed in the previous paragraph. Because at every moment in time, there exists a larger than zero decay of all internally learned soft constraints (see Example A.4).

Causal, Online Solutions

RNNs are practical, real-time-capable solutions.

All derived solutions in this work are RNN-based (and do not use bidirectional RNNs, see Appendix A.2), therefore they are online applicable and causal. For example, RNNO is applied online in paper B, and RING is applied online in paper C and paper D. Additionally in paper D, RING is shown to be real-time capable of up to $\approx 1000\,\text{Hz}$ on a desktop-grade CPU (Intel Xeon, single-core, 2.2 GHz), and to be applicable on a broad range of IMU sampling rates without subsequent training. ANODEC designs feedback controllers, and in paper F, it is used for SR control in real-time at 100 Hz using desktop-grade hardware. Higher sampling rates are easily possible because, after ANODEC's controller design, only the RNN that constitutes the designed feedback controller (with typically a parameter count of magnitude $\approx 100 - 1000$) must be applied online.

6

Conclusions and Future Work

"If we knew what it was we were doing, it would not be called research, would it?"
– Albert Einstein, theoretical physicist

In the present chapter, we revisit the contributions of this thesis, summarize them, and discuss their impact. We will also revisit the aim of the thesis and finish by discussing possible directions for future work.

6.1 Summary and Conclusions

This thesis presents a unified approach that is applicable to both motion analysis and motion control problems of systems performing dynamic motions. The unified approach uses simulation environments of the problem-to-be-solved to train RNNs on large amounts of data. The trained networks achieve zero-shot generalization from simulation to reality by combining extensive domain randomization with diverse simulated motions, offering easy-to-use, plug-and-play real-world solutions.

This unified approach is validated for two representative application examples (IMT and RT) for state estimation and control, and it advances their respective research fields in a variety of aspects. These achievements have resulted in six publications that are included in this thesis.

To illustrate the breadth of these contributions, we first explore the work presented in paper A, where we address the foundational question of system observability. Paper A addresses Research Gap A by proposing a method for observability analysis of IMT problems, titled RNNO. Observability is an essential property for ensuring feasible, consistent, and safe state estimation. In the paper, RNNO is used to derive previously unknown results about the observability properties of a three-segment KC with sparse IMU setup. Hence, RNNO provides answers to the research question, which KCs are observable under which conditions. Specifically, it is shown that the degree of observability decreases as the joint axes' directions of the two hinge joints align. RNNO and the simulation software, titled RCMG, are openly available such that future development of IMT algorithms is facilitated by first answering the observability question under ideal, in-silico conditions prior to a prolonged time investment of the researcher. Additionally, the concepts developed in paper A are potentially transferable to other domains with a

similar need for observability assessment. Thus, RNNO's applicability might reach beyond IMT since precise simulation environments are readily available in many domains, e.g., for battery development or autonomous piloting.

Building upon the observability framework established in paper A, paper B extends the RNNO to real-world scenarios by proposing novel domain randomization techniques to enable sim-to-real transfer. The successful transfer demonstrates the practical applicability of the unified approach in real-world settings, particularly in challenging environments where the lack of magnetometer measurements and a sparse sensor setup complicates state estimation. Paper B partly addresses Research Gap C and demonstrates real-world magnetometer-free and sparse IMT of a three-segment KC. The paper proposes several novel domain randomization techniques that extend the RCMG such that the in-simulation-trained RNNO of paper A zero-shoot generalizes from simulation to experiment. It also shows that the domain randomizations are critical for optimal sim-to-real transfer. After rapid initial convergence, the trained RNNO can achieve long-term stable tracking errors of < 4 degrees. These results constitute the first experimental proof of concept of the unified approach and demonstrate that it can be used for plug-and-play IMT in magnetically disturbed environments with minimal efforts and thus achieve unprecedented high applicability and usability at unprecedented low cost.

While paper B demonstrated the effectiveness of the unified approach in practical applications, paper C takes this a step further by proposing a plug-and-play pluripotent IMT solution, titled RING, that fundamentally changes how IMT is solved. Conventional IMT solutions typically rely on expert knowledge to identify, select, and finetune a problem-specific method, whereas RING offers an easy-to-use solution to a broad range of IMT problems. The paper addresses Research Gap D and eliminates the need for expert knowledge, allowing for a broader and more accessible use of the technology across various domains. Specifically, it is shown that RING – a single ANN – outperforms and thus unifies nine SOTA methods. Moreover, it enables several for-the-first-time applications; for example, it successfully tracks a four-segment KC from a sparse set of two magnetometer-free IMUs with an experimental MAE of < 7 degrees. Beyond the scope of IMT, it is shown that the decentralized approach of RING provides an advantageous, structural prior that aids network training compared to a more typical, centralized approach; a vital insight with possible implications for all of robotics. Overall, the introduction of RING not only makes IMT more accurate and less restrictive in established domains but also facilitates the accessibility of IMT technology by non-expert users and broadens its applicability to previously untapped domains.

To further demonstrate the capabilities of this pluripotent solution, paper D addresses the Research Gap B and Research Gap C and tackles – for the first time – one of the most complex IMT problems that combines: 1) magnetometer-free sensing, 2) sparse sensing[1], 3) sensor-to-segment calibration[2], and 4) motion artifacts[3]. By substituting RNNO with the more powerful RING, paper D can track a three-segment KC from two, nonrigidly-attached IMUs with an experimental MAE of ≈ 8 degrees. Additionally, it is shown that RING provides consistent estimates across a broad range of sampling rates and that it is real-time capable at > 500 Hz even on low-end hardware. Overall, paper D demonstrates that RING is a versatile and widely applicable solution capable of addressing even the most deeply rooted state estimation challenges.

[1] fewer sensors than segments

[2] identifying the joint positions and joint axes' directions in local sensor coordinates

[3] refers to the concept that sensors measure different values if the system is in motion, e.g., if the sensors are nonrigidly-attached, then the skin allows for relative motion between sensor and skeleton

Having demonstrated the unified approach's validity for state estimation, paper E expands its scope to the control of systems and accurately tracking dynamic reference motions. RT is one of the three pillars (next to state estimation and motion planning) for achieving dynamic motions in real-world systems. In paper E, the unified approach is used to address the Research Gap E by proposing a data-efficient learning control method titled ANODEC. ANODEC designs controllers in nonlinear systems without requiring prior model knowledge or extensive manual tuning. It efficiently learns from minimal input-output data and can control complex systems with nonlinearities like hysteresis. The method generalizes well to different tasks, achieving higher performance than classical control approaches, such as manually tuned PID controllers or transfer-function-based control. ANODEC enables practical real-time control in various domains with its automatic modeling-free approach, reducing human expertise and time for controller design.

Paper F utilizes ANODEC to address Research Gap F and enable a real-world robot arm actuated by an antagonistic pair of PAMs to self-calibrate and automatically learn to perform dynamic motions. ANODEC demonstrates its exceptional data-efficiency requiring only 30 seconds of experimental interaction time to outperform a manually-tuned PID controller. ANODEC's automatic and modeling-free nature is especially beneficial for SRs due to their complex nonlinear dynamics, hysteresis, and viscoelastic properties, which pose challenges for classical and model-based control approaches. To further strengthen the validity of ANODEC and the practicality of the self-calibrating SR, these results are confirmed for a modification of the SR (attaching a heavy mass and re-orienting the arm to move in the vertical plane) that significantly alters the dynamics. Overall, the unified approach, specifically ANODEC, can revolutionize SR control by making it more accessible, practical, and less dependent on expert knowledge.

The variety of contributions and their corresponding publications, in particular, demonstrate the validity of the unified approach for both state estimation and control of systems that perform dynamic, agile motions. The contributions made in the field of IMT have shown that in-simulation-trained RNNs can zero-shot generalize to reality and effectively fuse IMUs data in real-time, enabling accurate motion tracking of fast-moving KCs. Similarly, the contributions made in the field of RT have shown that a conceptually identical approach can be used to train RNNs that effectively actuate the system in real-time and enable accurate tracking of dynamic reference motions.

The code of (RCMG , RNNO, and RING)[4] and ANODEC [5], and the data[6] are made openly available. The author aims to support this software; if there are any problems, please open an issue on the respective GitHub repository.

These contributions also confirm the predicted impact of the unified approach (see Section 1.5). The approach has led to novel solutions for assessing observability, enabling sparse, magnetometer-free IMT, and enabling self-calibration in a SR arm. It minimizes the engineer's workload by a) using automatically generated simulated data, reducing the need for real-world data collection, and b) employing data-driven, ANN-based solutions instead of model-based approaches, lowering modeling efforts. Robustness is ensured through automatic domain randomizations rather than manual system identification. Furthermore, the approach reduces the requirement for expert knowledge, as single ANNs can address diverse problems, and domain randomizations enable self-calibration. RING exemplifies this, offering a broadly applicable,

[4] https://github.com/Simipixel/ring

[5] https://github.com/Simipixel/chain_control

[6] https://doi.org/10.7910/DVN/SGJLZA

self-calibrating IMT solution.

Unfortunately, no solution is perfect. Despite the significant advancements outlined in this thesis, specific challenges remain, particularly in handling more diverse sensor configurations, including other modalities such as optical measurements, and enhancing applicability to real-world human motion capture. These challenges form the basis of future work, discussed in the next section.

6.2 Future Work Directions

"Nothing is perfect, and life is a messy compromise, but there is beauty in its imperfections."
– Unknown

6.2.1 Random Motion and Domain Randomization of Biomechanical Models

The IMT methods developed in paper A, B, D, and C rely on training data generated by simulating random mechanical joint motions. To further enhance IMT performance for real-world human motion capture, training ANNs in simulations that use biomechanical models could be beneficial, as less accurate modeling and broad domain randomizations often compromise prediction performance (see Section 2.1).

However, the applicability of the unified approach also requires the ability to generate random motions. For biomechanical models, this is slightly more complex than mechanical models, as the joints in biomechanical models are actuated via tendons that contract upon nerve stimuli. For random motion generation, treating tendons (and secondary joint degrees of freedom) as passive force generators may prove beneficial, actuating only the primary DOFs. For instance, the human knee is usually modeled as a hinge joint with a coupled translational degree, actuated by an antagonistic tendon pair. Instead of simulating muscle stimuli, PID control could be applied to the hinge's rotational degree, allowing the translational degree and tendons to follow naturally. This framework should allow for the generation of natural random motion in biomechanical models.

6.2.2 GaitTracker: A Specialized Flavor of RING for Lower Extremities

While RING is a general-purpose method, tailored solutions may yield better performance for specific tasks. Building on the ideas from Section 6.2.1, a specialized variant of RING, potentially titled GaitTracker, which focuses on tracking lower extremity movements (hip, femur, tibia, and foot) using a sparse number of magnetometer-free IMUs, would be beneficial.

The RING architecture can be used from paper A and combined with a feature-rich simulation environment of lower extremities. MyoSuite offers a diverse range of biomechanical models, including models of lower extremities, upper body, arms, and hands (Caggiano et al., 2022). MuJoCo[7] efficiently simulates these models, making them highly suitable for training of GaitTracker.

[7] MuJoCo stands for Multi-Joint dynamics with Contact, a general purpose physics engine (Todorov et al., 2012)

6.2.3 Multi-modal IMT

For the task of motion analysis, combining data from multiple sources–such as one or several cameras positioned at various perspectives and numerous IMUs–provides a more comprehensive understanding of the motion. Each sensor type offers complementary information; camera data captures a visual understanding of the attachment of IMUs, while the IMUs provide detailed angular velocity and acceleration data. The challenge lies in fusing these diverse data streams to achieve the highest possible motion-tracking accuracy.

The unified approach and its derivatives, such as RING, can be extended to fuse, yielding a hybrid solution that fuses visual and inertial information. The required training data is readily available since modern simulation environments, e.g., MuJoCo, natively support rendering the physical scene, enabling detailed simulation of camera data. Domain randomizations of camera angles, perspectives, and dynamic ranges are straightforward. This hybrid solution could be highly beneficial for enabling RING to perform advanced sensor-to-segment calibration of IMUs, as accurate calibration is essential for reducing misalignment errors between the sensor readings and the actual physical movements. Moreover, the IMUs could be physically enhanced by printing orientation markers on their housing, enabling the RING network to identify segment alignment and sensor-to-segment calibration in real-world scenarios. This setup offers additional potential for trained networks' plug-and-play capabilities while enhancing tracking performance in sparse sensor setups.

6.2.4 OMC Motion Artifact Reduction

For motion analysis, we typically estimate the motion of an articulated, rigid-body system from sensor data. For human motion analysis and using IMUs, this corresponds to estimating the skeleton motion from wearable inertial sensors.

However, in real-world conditions, the attachment of these sensors violates the rigid body assumption[8]. The sensors are nonrigidly-attached to the skin, introducing motion artifacts in the motion estimates, due to relative motion between sensor and skeleton during dynamic motions. In paper C, these motion artifacts are modeled, corresponding IMU data simulated, and, after training, RING has learned to compensate real-world motion artifacts effectively.

However, not only IMUs are nonrigidly attached to the human skin. OMC markers are similarly attached to the skin, and thus, motion artifacts are entering the motion estimates obtained via gold-standard OMC systems. The unified approach could be used to learn ANNs that filter the OMC position trajectories and reduce motion artifacts by simulating large amounts of motion-artifacts-prone data. The obtained ANN would result in a broadly applicable solution, improving the motion tracking performance of existing OMC systems.

6.2.5 An IMT Foundation Model and RING Finetuning

Foundation models are large-scale pretrained models that serve as the basis for a wide range of downstream tasks. These models are trained on vast amounts of data. The key idea behind foundation models is that they can learn general-purpose representations of data, which can then be adapted to specific tasks with minimal additional training. Examples include GPT

[8]The rigid body assumption refers to the simplification that an object does not deform under force, meaning that the distances between all points on the object remain constant, allowing only for translation and rotation as modeled by the joints, without any internal structural changes.

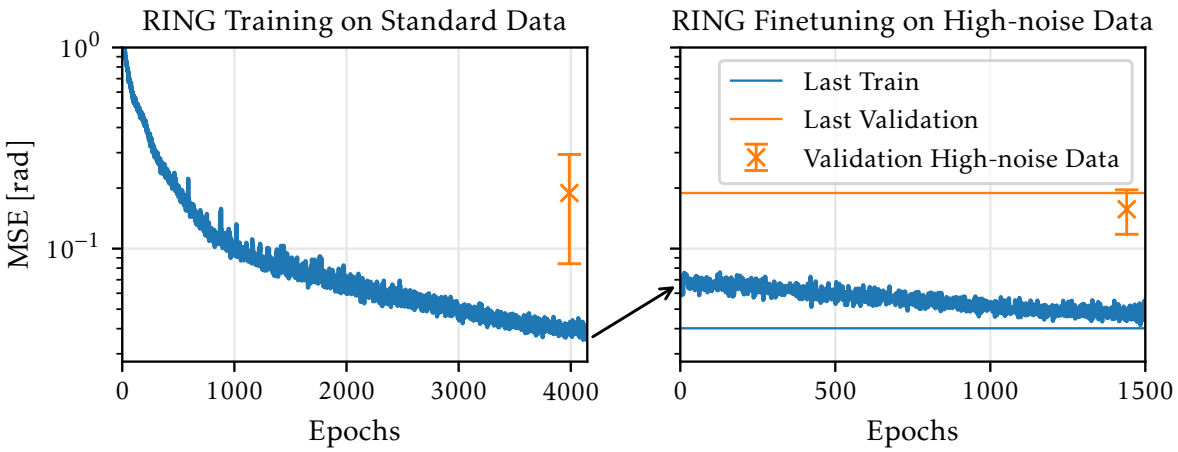


Figure 6.1: *Finetuning can compute-efficiently adapt a pretrained version of* RING *to a new data distribution. (Left subplot) The training of* RING *as published in paper C and the experimental validation performance of the trained* RING *on high-noise* IMU *data. (Right subplot) Additional training on high-noise simulated* IMU *data increases* RING *'s performance on the experimental validation data with* 1500 *training epochs compared to the* > 4000 *epochs required for* RING *'s training from scratch.*

(used in ChatGPT), BERT, RoBERTa for natural language processing, and models like CLIP and Dall-E for vision-language tasks.

Finetuning is the process of taking a pretrained foundation model and adapting it to a specific task or domain by training it further on a smaller, task-specific dataset. Finetuning allows the model to adjust its general-purpose knowledge to suit better the nuances of the specific task to which it is applied.

RING is a foundation model for IMT in the sense that it is trained on vast amounts of (simulated) data and that it can be applied to a broad range of IMT problems. In Figure 6.1, RING is finetuned to achieve a lower validation error on especially high-noise IMU data. However, RING is not a foundation model because it is trained to predict orientations and not practical, general-purpose representations for downstream tasks. For example, in its current state, RING can not be easily used for explicitly estimating the IMU calibration parameters, which requires the prediction of values in units of meters. The latter problem would require complete retraining of RING and can not easily be achieved by finetuning.

Two distinct future work directions are worth investigating. First, in its current state, RING can already be finetuned (as seen in Figure 6.1) using simulated data. Using experimental data for finetuning to achieve an even better sim-to-real transfer is intriguing. Second, future work should investigate the possibility of a general-purpose RING foundation model for IMT in combination with task-specific network heads (or decoders) and finetuning (if required). The idea is as follows: There is a single, (very) large ANN that processes all IMU data and task-specific prior knowledge to update and return a high-dimensional, highly informative latent state. This latent state is then processed by task-specific, feedforward ANNs (network heads) that decode the latent state to the task-specific output spaces. This ensemble of foundation model and task-specific heads is then trained end-to-end on as many different tasks as possible.

This decomposition has two key advantages: 1) if in the future a new downstream task emerges, then either a pretrained or new network head is finetuned in combination with the frozen foundation model, and 2) if in the future the data distribution changes (e.g., a new oddly-behaving IMU becomes available) then the foundation model can be finetuned with frozen network heads. The latter is because the latent state contains already processed higher-level information that stays identical; thus, the network heads do not require updating. Overall, both scenarios significantly reduce the computational demands required for network training.

6.2.6 Learning to Learn: ANODEC's Evolution to a Zero-data Solution for Control

Recall from Section 4.3.3 that ANODEC works in a two-step approach. First, it approximates the real-world dynamics Ψ by a NODE that is learned from $N \in \mathbb{N}$ experimental input-output data pairs $\{(\boldsymbol{u}_i(t), \boldsymbol{y}_i(t)) \mid \forall i\}$. The selection of input trajectories is critical for optimal system identification and requires expert knowledge. In addition, recording the data pairs can be time-consuming.

In contrast, the proposed learning-to-learn approach entirely and automatically handles this system identification step, which previously involved selecting input trajectories and manually recording training data. To this end, instead of two ANNs, one for system identification and one for control, the learning-to-learn approach uses two ANNs that together form the controller; a trainable ANN f_θ that predicts the required reference motion that should be applied to the closed-loop system for optimal system identification by the second, trainable ANN π_θ that controls the system. This controller combination is trained in simulation to perform online system identification for an arbitrary, unknown dynamics.

Thus, the learning-to-learn approach trains a first network that predicts the needed reference motion. This network aids the second network in identifying the system, thus enabling good control performance later. The entire approach is illustrated in Figure 6.2, and it will provide a fully autonomous learning control approach that can learn to adapt to unknown dynamics online and without user intervention.

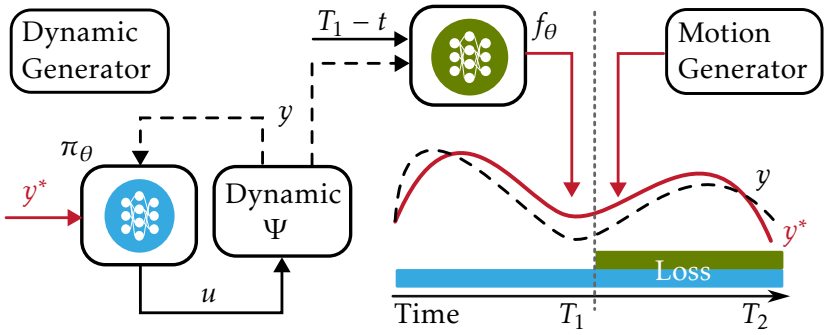


Figure 6.2: *Learning to learn. An ANN-based policy π_θ is trained in simulation by closing the loop with a randomly generated system dynamics Ψ. Initially, a second ANN generates a reference motion y^* online until a non-trainable motion generator takes over. Both ANNs are trained end-to-end by minimizing the RT loss, but f_θ optimizes the tracking error after the motion generator takes over. This architecture should enable f_θ to learn to generate motions useful to the policy π_θ for system identification. By training a single set of parameters on a broad range of dynamics and motions, the two trained ANNs hopefully zero-shot generalize to real-world scenarios.*

Appendix

A

Recurrent Neural Networks

"Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less."
– Marie Curie, the first woman in France to obtain a doctorate in physics

This thesis focuses on developing a unified approach for motion analysis and control of dynamic systems, using RNNs to address data efficiency, robustness, and real-world applicability challenges. RNNs are central to the thesis because their ability to process sequential data makes them ideal for parameterizing policies in POMDPs, as encountered in both motion control and analysis tasks. This chapter delves deeply into the structure, training methods, and challenges of RNNs, highlighting their relevance in achieving the robust, plug-and-play solutions proposed in this thesis.

A.1 What are RNNs?

RNNs are a class of ANNs designed to recognize patterns in sequences of data such as text and language or numerical time series data. Unlike feedforward ANNs, RNNs have loops, allowing information to persist. Mathematically, an RNN is typically characterized by a discrete-time step function f_θ which processes $F \geq 1$ features stacked as the input vector $\boldsymbol{x}_t \in \mathbb{R}^F$ at time t and the previous memory state (or hidden state) of the RNN with M units $\boldsymbol{\xi}_t \in \mathbb{R}^M$, and computes the next hidden state $\boldsymbol{\xi}_{t+1}$, i.e.

$$\boldsymbol{\xi}_{t+1} = f_\theta\left(\boldsymbol{\xi}_t, \boldsymbol{x}_t\right). \tag{A.1}$$

Here, θ denotes the set of parameters of the RNN, and these parameters are shared across time, i.e., the same step function with the same parameters is used to process the input data at all times.

Example A.1

A vanilla RNN with hidden state size $M \in \mathbb{N}$ that processes inputs of size $F \in \mathbb{N}$ has the parameters given by $\mathbf{W}_\xi \in \mathbb{R}^{M\times M}$, $\mathbf{W_i} \in \mathbb{R}^{M\times F}$, and $\boldsymbol{b} \in \mathbb{R}^M$, a nonlinear activation function σ (typically tanh or ReLU), and is defined by the step function given by

$$\boldsymbol{\xi}_{t+1} = \sigma\left(\mathbf{W}_\xi \boldsymbol{\xi}_t + \mathbf{W}_i \boldsymbol{x}_t + \boldsymbol{b}\right). \tag{A.2}$$

Example A.4 shows the equivalent NODE or continuous time expression.

Often, the RNN is combined with a second feedforward ANN, which creates the required output of a suitable dimensionality. The output function g_θ processes the hidden state (and sometimes the input vector) and returns the output vector $\boldsymbol{y}_t \in \mathbb{R}^G$ at time t, i.e.,

$$\boldsymbol{y}_t = g_\theta\left(\boldsymbol{\xi}_t, \boldsymbol{x}_t\right) \tag{A.3}$$

Note that in eq. (A.1), the step function f_θ is itself a feedforward network, and it is used to model the *discrete-time* dynamics, this is in contrast to NODEs (see Section A.3.1) where a feedforward network is used to model the *continuous-time* dynamics. Due to their ability to store information in memory, RNNs are also called stateful functions. Mathematically, this can be better seen by combining equations eq. (A.1) and eq. (A.3) and omitting the explicit input-output handling of the hidden state $\boldsymbol{\xi}$. This then gives:

$$\boldsymbol{y}_t = \overleftarrow{h}_\theta\left(\boldsymbol{x}_t\right) \tag{A.4}$$

where the backward-pointing arrow $\overleftarrow{(..)}$ indicates that the ANN h_θ has an internal state, a loop back to itself. The internal state is why RNNs are referred to as stateful functions. Given an initial value for the hidden state $\boldsymbol{\xi}_0$ at time zero (typically zeros), eq. (A.4) can be *unrolled* in time, which creates a neural network that maps a time series (or sequence) of input data $\mathbf{X} \in \mathbb{R}^{T\times F}$ consisting of T timesteps to a time series of output data $\mathbf{Y} \in \mathbb{R}^{T\times G}$, i.e.,

$$\mathbf{Y} = h_\theta\left(\mathbf{X}, \boldsymbol{\xi}_0\right). \tag{A.5}$$

Both eq. (A.4) and eq. (A.5) are referred to as an RNN.

A.2 Why RNNs?

Dynamic problems require dynamic solutions, and RNNs are designed for time-series data and excel in this area. They are well-suited because they allow for 1) online applicability (causality), 2) maintaining an internal state (memory), and 3) sharing parameters across time, enabling them to capture temporal dependencies effectively. Let us dive into each aspect in more detail.

- Online applicability/Causality: RNNs process every input after the other in a causal way. Consider a conventional feedforward ANN f/f_θ and an RNN step function g/g_θ. The RNN is causal because, in lifted form, the upper-diagonal terms (highlighted in blue) are zero.

$$\overbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} f_{11} & \color{blue}{f_{12}} & \dots & \color{blue}{f_{1T}} \\ f_{21} & f_{22} & \dots & \color{blue}{f_{2T}} \\ \vdots & \vdots & \ddots & \color{blue}{\vdots} \\ f_{T1} & f_{T2} & \dots & f_{TT} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_T \end{bmatrix}}^{\text{Feedforward NN}} \qquad \overbrace{\begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_T \end{bmatrix} = \begin{bmatrix} g & \color{blue}{0} & \dots & \color{blue}{0} \\ 0 & g & \dots & \color{blue}{0} \\ \vdots & \vdots & \ddots & \color{blue}{\vdots} \\ 0 & 0 & \dots & g \end{bmatrix} \begin{bmatrix} u_1, \xi_0 \\ u_2, \xi_1(u_1) \\ \vdots \\ u_T, \xi_{T-1}(u_{T-1}, \dots) \end{bmatrix}}^{\text{Recurrent NN}}$$

A notable exception are bidirectional RNNs. These networks typically first unroll their step function forwards in time and then backwards in time while maintaining the hidden state between the two passes. This process is similar to a forward-backward filter, and such an operation mode is neither causal nor online applicable.

- Memory: RNNs can store information that persists in time in their hidden state, which gives them a form of memory. In other words, the output from an RNN unit depends on the current input and previous inputs. The RNN has memory because it computes its current output from the current input and hidden state (highlighted in blue). In contrast, the feedforward ANN is a static function of the inputs $\boldsymbol{u}$.

$$\overbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1T} \\ f_{21} & f_{22} & \dots & f_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ f_{T1} & f_{T2} & \dots & f_{TT} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_T \end{bmatrix}}^{\text{Feedforward NN}} \qquad \overbrace{\begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_T \end{bmatrix} = \begin{bmatrix} g & 0 & \dots & 0 \\ 0 & g & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & g \end{bmatrix} \begin{bmatrix} u_1, \xi_0 \\ u_2, \xi_1(u_1) \\ \vdots \\ u_T, \xi_{T-1}(u_{T-1}, \dots) \end{bmatrix}}^{\text{Recurrent NN}}$$

- Parameter Sharing Across Time: RNNs share the same parameters (weights and biases) across all steps. Parameter sharing reduces the total number of parameters and complexity of the model. The RNN has shared parameters across time because it uses the same weights and biases g at every timestep (highlighted in blue). In contrast, the feedforward ANN has independent parameters for each timestep, and its number of parameters thus scales quadratically with the input length.

$$\overbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & \dots & f_{1T} \\ f_{21} & f_{22} & \dots & f_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ f_{T1} & f_{T2} & \dots & f_{TT} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_T \end{bmatrix}}^{\text{Feedforward NN}} \qquad \overbrace{\begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_T \end{bmatrix} = \begin{bmatrix} g & 0 & \dots & 0 \\ 0 & g & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & g \end{bmatrix} \begin{bmatrix} u_1, \xi_0 \\ u_2, \xi_1(u_1) \\ \vdots \\ u_T, \xi_{T-1}(u_{T-1}, \dots) \end{bmatrix}}^{\text{Recurrent NN}}$$

Example A.2

Let us consider the double integrator system $m\ddot{x} + c\dot{x} + kx = u$ where $x(t) \in \mathbb{R}$ is the displacement of the mass from its equilibrium position, $m \in \mathbb{R}$ is the mass, $c \in \mathbb{R}$ is the damping coefficient, $k \in \mathbb{R}$ is the spring constant, and $u \in \mathbb{R}$ is the external force applied to the system. This system can be written in so-called state-space representation characterized by $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ matrices by defining $x_1 := x$, $x_2 := \dot{x}$, and $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ such that system dynamics takes on the following form

$$\begin{aligned} \dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u} \qquad &\leftrightarrow \qquad \dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \boldsymbol{u}, \\ \boldsymbol{y} = \mathbf{C}\boldsymbol{x} + \mathbf{D}\boldsymbol{u} \qquad &\leftrightarrow \qquad \boldsymbol{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \end{bmatrix} \boldsymbol{u}. \end{aligned}$$

Note that the second equation is already of the form of equation eq. (A.3) and the dynamics function $\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u}$ can be easily converted to discrete-time form with a time interval $\Delta t \in \mathbb{R}$,

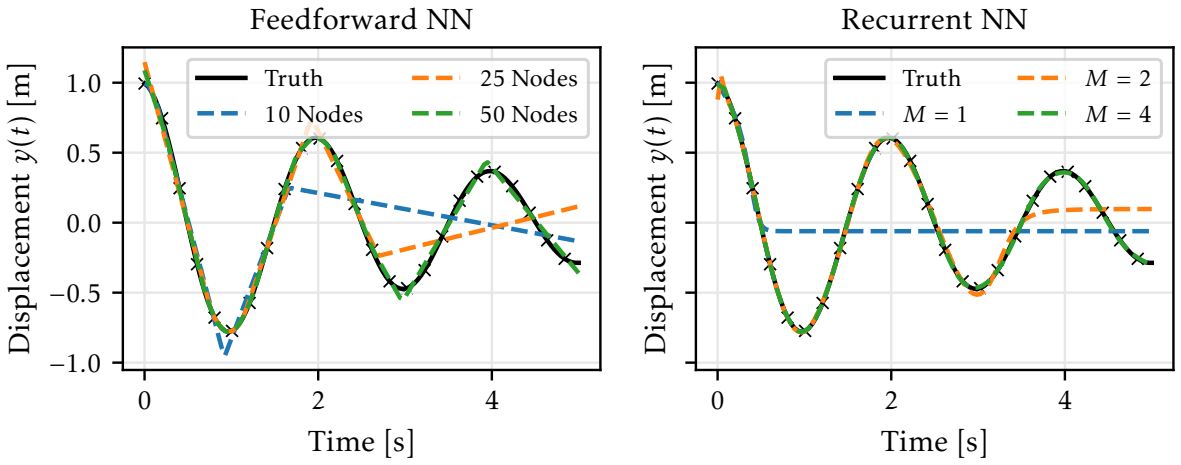


Figure A.1: *FNN and RNN are used to fit input-output data of the double integrator system of Example A.2. Observations: 1) RNN is naturally smooth, 2) RNN requires (much) less parameters to be accurate. Perhaps surprisingly, the RNN must have $M > 2$ hidden state dimensionality for accurate prediction.*

i.e. $\boldsymbol{x}_{t+1} = \exp^{\mathbf{A}\Delta t}\boldsymbol{x} + \mathbf{A}^{-1}\left(\exp^{\mathbf{A}\Delta t} - \mathbf{1}\right)\mathbf{B}\boldsymbol{u}$. This has been achieved by transforming a second-order single-state ODE into a first-order ODE with two coupled state variables (because the off-diagonal terms of the A matrix are not zero). The state of the system, that is, the displacement and velocity, structurally corresponds to the internal state of the RNN.

Suppose we have collected the system's input-output data at T equidistant time intervals and would like to learn the system input-output map for, e.g., subsequent control-related purposes. Then, we can fit either a FNN or a RNN to the data and obtain Figure A.1. Perhaps surprisingly, the RNN must have $M > 2$ hidden state dimensionality for accurate prediction. Note how RNN with $M = 1$ can not (and does not) oscillate; it only decays to a fixed value.

Example A.3

The dynamics of natural language. RNNs have also been proposed for NLP tasks. The reason is that words and language depend on the context, which is due to the dynamic aspect of language; consider the following example:

- Sentence 1: "He saw that her eyes were full of tears."
- Sentence 2: "He saw that her project was full of errors."

In both sentences, the word "saw" appears, but its meaning shifts depending on the context provided by the rest of the sentence: In Sentence 1, "saw" is used in the context of perceiving visually; he notices her emotional state as indicated by her tears. In Sentence 2, "saw" pertains to noticing or realizing something intellectually; in this case, the many mistakes in her project. This shift in meaning is a clear example of how language is dynamic and context-dependent. The word remains the same, but its interpretation changes dramatically based on the surrounding words. Also note that in contrast to physical systems, language is not causal. The word

"saw" 's intended meaning only becomes apparent after observing the entire sentence.

A.3 Variants of RNNs

Several variants of RNNs have been developed to address different challenges in sequence modeling. These include:

- Vanilla RNN (defined in Example A.1),
- LSTM proposed in Hochreiter and Schmidhuber (1997) to address gradient-related problems, or
- GRU, a simplified LSTM, proposed in (Cho et al., 2014).
- Ongoing research also explores newer architectures such as the Linear Recurrent Unit (LRU) (Orvieto et al., 2023).

A.3.1 Neural Ordinary Differential Equations

While RNNs are typically characterized by learning a discrete-time step function f_θ (see eq. (A.1)), NODEs are obtained by modeling the right-hand side of a continuous-time differential equation using a feedforward ANN

$$\frac{d\boldsymbol{\xi}(t)}{dt} = f_\theta\left(\boldsymbol{\xi}(t), \boldsymbol{x}(t)\right). \tag{A.6}$$

More precisely: the expression $\frac{d\boldsymbol{\xi}(t)}{dt} = f_\theta\left(\boldsymbol{\xi}(t)\right)$ is referred to as a NODE, whereas the expression eq. (A.6) is referred to as a controlled NODE (Kidger, 2022), however we do not make this distinction here. NODEs are well established for modeling and forecasting of time series data (Fitzgerald et al., 2023; Hasani et al., 2022; Kidger, 2022; Kidger et al., 2020). Also note that they should not be confused with physics-informed ANNs which use ANNs to learn solutions to known differential equations, i.e., $\frac{d\boldsymbol{y}_\theta(t)}{dt} = f\left(\boldsymbol{y}_\theta(t)\right)$ where f is known (Cuomo et al., 2022; Hao et al., 2023).

Example A.4

We can recover the equation of the vanilla RNN as defined in eq. (A.2) by explicit Euler discretization with a time interval $\Delta t \in \mathbb{R}$ of

$$\frac{d\boldsymbol{\xi}(t)}{dt} = \frac{1}{\Delta t}\left(\sigma\left(\mathbf{W}_\xi \boldsymbol{\xi}(t) + \mathbf{W}_i \boldsymbol{x}(t) + \boldsymbol{b}\right) - \boldsymbol{\xi}(t)\right)$$

Note the term $-\boldsymbol{\xi}$, which shows that the hidden state exponential decays, and this, in parts, explains why RNNs struggle to capture long-range dependencies (Kidger, 2022). Example A.1 shows the equivalent RNN step function and discrete time expression.

A.4 Backpropagation Through Time

Both RNNs and NODEs are typically trained using standard backpropagation However, the hidden state $\boldsymbol{\xi}_{t+1}$ at time $t+1$ depends on the current input $\boldsymbol{x}_t$ *and the previous hidden state* $\boldsymbol{\xi}_t$ (which itself depends on the previous input $\boldsymbol{x}_{t-1}$ and so on). Hence, it depends on *all previous inputs* More formally, suppose we want to minimize the MSE between prediction $\hat{y}_t$ and truth y_t. To this end, we compute the gradient using the chain rule and equations (A.1), (A.3):

$$\begin{aligned}
\frac{d\left((\hat{y}_t - y_t)^2\right)}{d\theta} &= 2\,(\hat{y}_t - y_t)\,\frac{d\hat{y}_t}{d\theta} \\
&= 2\,(\hat{y}_t - y_t)\,\frac{dg_\theta(\boldsymbol{\xi}_t, \boldsymbol{x}_t)}{d\theta} \\
&= 2\,(\hat{y}_t - y_t)\left(\frac{dg_\theta}{d\theta}\Big|_{\xi_t,\boldsymbol{x}_t} + \frac{dg_\theta}{d\boldsymbol{\xi}}\Big|_{\xi_t,\boldsymbol{x}_t}\,\frac{d\boldsymbol{\xi}_t}{d\theta}\right) \\
&= 2\,(\hat{y}_t - y_t)\left(\frac{dg_\theta}{d\theta}\Big|_{\xi_t,\boldsymbol{x}_t} + \frac{dg_\theta}{d\boldsymbol{\xi}}\Big|_{\xi_t,\boldsymbol{x}_t}\,\frac{df_\theta(\boldsymbol{\xi}_{t-1}, \boldsymbol{x}_t)}{d\theta}\right) \\
&= 2\,(\hat{y}_t - y_t)\left(\frac{dg_\theta}{d\theta}\Big|_{\xi_t,\boldsymbol{x}_t} + \frac{dg_\theta}{d\boldsymbol{\xi}}\Big|_{\xi_t,\boldsymbol{x}_t}\left(\frac{df_\theta}{d\theta}\Big|_{\xi_{t-1},\boldsymbol{x}_t} + \frac{df_\theta}{d\boldsymbol{\xi}}\Big|_{\xi_{t-1},\boldsymbol{x}_t}\,\frac{d\boldsymbol{\xi}_{t-1}}{d\theta}\right)\right)
\end{aligned} \tag{A.7}$$

where there will be a recursive expansion (blue highlighted terms) until the first timestep of the sequence.

Example A.5

Backpropagation learns inverse in time. Note how the error for both feedforward ANN and RNN predictions in Figure A.1 are more prominent towards the end of the sequence. One may interpret this as due to the larger absolute amplitude of the target variable $y(t)$ in this time interval. However, this only tells half the story in the case of the RNN. This missing half is due to backpropagation through time, which emphasizes earlier predictions because everything afterward depends on them (and not vice versa). To see this, let us reconsider the example but with a constant signal amplitude and look at the prediction of the perfect fit RNN ($M = 4$) after a different number of learning steps in Figure A.2.

A.5 Challenges, Solutions and Future Directions for RNNs

RNNs are well-suited for sequential data, but several challenges limit their applicability. First, due to their recurrence relation, RNNs require sequential computation. While this allows them to store information in their internal memory, allowing information to persist, it also restricts scalability. Using just-in-time compilers such as JAX Bradbury et al. (2018), one can (to some extent) trade off memory footprint for speed. Additionally, there are two main challenges when training RNNs with backpropagation through time: They are referred to as vanishing and exploding gradients. To motivate these two challenges, consider a learning problem that can be solved by identifying a pattern in the input data $\boldsymbol{x}_t$ at time t that manifests in the input that is $T \in \mathbb{N}$ timesteps apart (or, in the past), then from eq. (A.7) it follows that the required

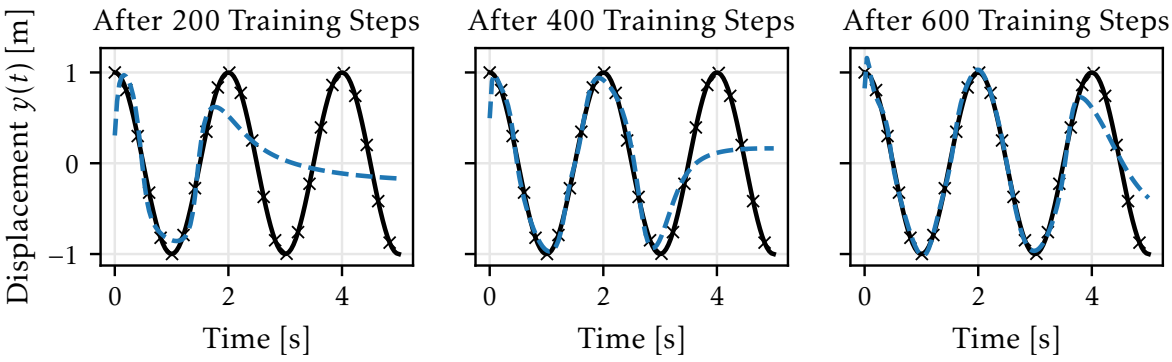


Figure A.2: *RNN fits the cosine function and its prediction after a different number of training steps. Due to backpropagation through time earlier values are learned first.*

gradient for learning this pattern is proportional to

$$\nabla_\theta \mathcal{L} \propto \prod_{i}^{T-1} \frac{df_\theta}{d\boldsymbol{\xi}}\bigg|_{\boldsymbol{\xi}_{t-i}, \boldsymbol{x}_{i-i+1}} . \tag{A.8}$$

Consider Example A.6 for an example of such a learning problem. In this learning problem, vanishing gradients occur when $\left|\frac{df_\theta}{d\boldsymbol{\xi}}\right| < 1$ and T is large, then the recursive expansion in eq. (A.7) leads to a gradient that exponentially decays to zero. On the other hand, exploding gradients occur when $\left|\frac{df_\theta}{d\boldsymbol{\xi}}\right| > 1$ and T is large since then the gradient exponentially grows.

The above variants of RNNs have been proposed to address these challenges. For example, the careful design of network architectures as found in the LSTM and GRU architecture ensures that $\left|\frac{df_\theta}{d\boldsymbol{\xi}}\right| = 1 \ \forall \boldsymbol{\xi}, \boldsymbol{x}$ which avoids both vanishing and exploding gradients. A second complementary solution is to use truncated backpropagation through time. It involves truncating the recursive expansion early, i.e., assuming in eq. (A.7) that after $K \in \mathbb{N}$ steps the hidden state no longer depends on the parameters and is just a constant, that is $\frac{d\boldsymbol{\xi}_{t-K}}{d\theta} \stackrel{!}{=} 0$. Finally, a last solution is simply not using RNNs. Initially proposed in Vaswani et al. (2023), Transformers have proven highly effective at capturing long-range dependencies.

Example A.6

Revisit the Example A.3. In order to properly distinguish the two intended meanings of the word "saw" in the two contexts, an RNN would require efficient backpropagation through the entire context. If we assume a single character dictionary (this is not the SOTA choice, SOTA large language models typically use tokenization with 8-32k vocabulary sizes using the, e.g., byte-pair-encoding (Sennrich et al., 2016) or unigram language model (Kudo, 2018)), then this would imply backpropagation through a sequence of length 40 and 43, respectively.

B

Remarks, Definitions, Theorems, and Else

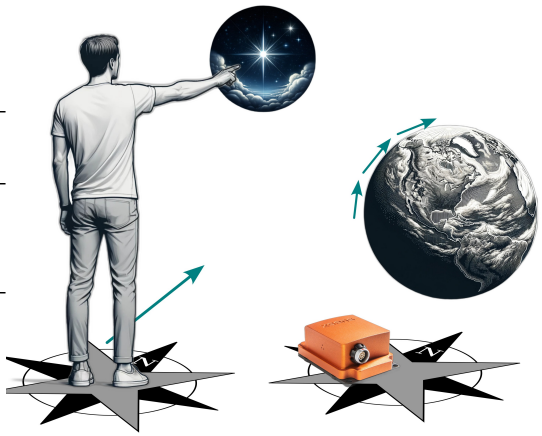
Limitations of Linearizations Without loss of generality, consider the nonlinear dynamics $\dot{x} = f(x)$. Let Δt be the discrete timestep size and Δf be the difference in linearization between timesteps which equals the maximum error due to the linearization assumption *in between* timesteps:

$$\Delta f(x,t) = \frac{1}{2!} \underbrace{\left.\frac{d^2 f}{dx^2}\right|_x}_{\text{nonlinearity}} \left(\underbrace{\left.\frac{dx}{dt}\right|_t}_{\text{variation}} \Delta t + O\left(\Delta t^2\right) \right)^2 + O(\Delta t^3) \tag{B.1}$$

Thus, the variability of the state $\frac{dx}{dt}$ increases the linearization error quadratically.

Example B.1

Analogous navigation principles. The Polar Star's unique position, nearly aligned with Earth's rotational axis, makes it appear stationary in the night sky directly above the North Pole. Vikings exploited this natural phenomenon for open-sea navigation when no landmarks were visible, establishing one of humanity's earliest reliable orientation systems.



Our interaction with the Polar Star can be modeled as a directional sensor that parallels modern IMUs using Earth's magnetic field for northbound orientation: when pointing at the Polar Star, our arm's orientation relative to our body encodes our rotation. As we turn while maintaining this pointing gesture, the arm-to-body angle continuously reflects our orientation relative to this fixed celestial reference. However, this measurement alone leaves one DOF undetermined - the rotation around the arm's axis.

To establish unique orientation, we need a second reference direction, either from another star or from pointing downward (analogous to an IMU's accelerometer measuring gravity's

direction). The plane formed by these two pointing directions remains invariant under rotation, while its projection into our body's coordinate system provides sufficient information to uniquely determine our complete orientation.

Definition B.2. Observability by example. If for all input trajectories $\boldsymbol{u}(t)$ and initial conditions $\boldsymbol{x}(t = 0)$ (i.e., for all motions), there exists an observer that accurately tracks the target state variable $\hat{\boldsymbol{x}} \subseteq \boldsymbol{x}$ from the measurements $\boldsymbol{y}(t' < t)$ in the dynamical system governed by $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$ and $\boldsymbol{y}(t) = g(\boldsymbol{x}(t), \boldsymbol{u}(t))$, then the tuple of $f, g, \hat{\boldsymbol{x}}, \boldsymbol{y}$ is said to be observable. Note that in practice we assess the observability not for all state trajectories but rather for a (large) number thereof.

Theorem B.3. ***Non-observability by example.*** *From Definition B.2 it follows directly that, if for the tuple of $f, g, \hat{\boldsymbol{x}}, \boldsymbol{y}$, there exists at least two input trajectories $\boldsymbol{u}_{1/2}(t)$ and/or initial conditions $\boldsymbol{x}_{1/2}(t = 0)$ (i.e., two or more motions) with corresponding measurement $\boldsymbol{y}_{1/2}(t)$ and state trajectories $\hat{\boldsymbol{x}}_{1/2}(t)$, such that there exists a moment in time $t' \in \mathbb{R}$ with $\hat{\boldsymbol{x}}_1(t') \neq \hat{\boldsymbol{x}}_2(t')$ and $\boldsymbol{y}_1(t) = \boldsymbol{y}_2(t)\ \forall t < t'$, then the tuple is said to be non-observable.*

Definition B.4. Controllability by example. If for all initial state $\boldsymbol{x}_0$ and final states $\boldsymbol{x}_f$, there exists an input trajectory $u(t)$ such that the systems evolves from the initial to the final state.

Bibliography

MTw Awinda User Manual, May 2018.

Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Edward Lee, Sergey Levine, Yao Lu, Isabel Leal, Sharath Maddineni, Kanishka Rao, Dorsa Sadigh, Pannag Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Steve Xu, and Zhuo Xu. AutoRT: Embodied Foundation Models for Large Scale Orchestration of Robotic Agents, July 2024.

Simon Bachhuber, Daniel Weber, Ive Weygers, and Thomas Seel. RNN-based Observability Analysis for Magnetometer-Free Sparse Inertial Motion Tracking. In *2022 25th International Conference on Information Fusion (FUSION)*, pages 1–8, Linköping, Sweden, July 2022. IEEE. ISBN 978-1-73774-972-1. doi: 10.23919/FUSION49751.2022.9841375.

Simon Bachhuber, Dustin Lehmann, Eva Dorschky, Anne D. Koelewijn, Thomas Seel, and Ive Weygers. Plug-and-Play Sparse Inertial Motion Tracking With Sim-to-Real Transfer. *IEEE Sensors Letters*, 7(10):1–4, October 2023a. ISSN 2475-1472. doi: 10.1109/LSENS.2023.3307122.

Simon Bachhuber, Ive Weygers, and Thomas Seel. Neural ODEs for Data-Driven Automatic Self-Design of Finite-Time Output Feedback Control for Unknown Nonlinear Dynamics. *IEEE Control Systems Letters*, 7:3048–3053, July 2023b. ISSN 2475-1456. doi: 10.1109/LCSYS.2023.3293277.

Simon Bachhuber, Alexander Pawluchin, Arka Pal, Ivo Boblan, and Thomas Seel. A soft robotic system automatically learns precise agile motions without model information. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2024a. doi: 10.48550/arXiv.2408.03754.

Simon Bachhuber, Ive Weygers, Dustin Lehmann, Mischa Dombrowski, and Thomas Seel. Recurrent Inertial Graph-Based Estimator (RING): A Single Pluripotent Inertial Motion Tracking Solution. *Transactions on Machine Learning Research*, July 2024b.

Simon Bachhuber, Ive Weygers, and Thomas Seel. Dispelling four challenges in inertial motion tracking with one recurrent inertial graph-based estimator (RING). In *12th IFAC Symposium on Biological and Medical Systems - 12th BMS 2024*, September 2024c. doi: 10.48550/arXiv.2409.02502.

Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, Cambridge, 2017. doi: 10.1017/9781316671528.

John Bechhoefer. Feedback for physicists: A tutorial essay on control. *Reviews of Modern Physics*, 77(3):783–836, August 2005. doi: 10.1103/RevModPhys.77.783.

Petar Bevanda, Max Beier, Shahab Heshmati-Alamdari, Stefan Sosnowski, and Sandra Hirche. Towards Data-driven LQR with Koopmanizing Flows. *IFAC-PapersOnLine*, 55(15):13–18, January 2022. ISSN 2405-8963. doi: 10.1016/j.ifacol.2022.07.601.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

D.A. Bristow, M. Tharayil, and A.G. Alleyne. A survey of iterative learning control. *IEEE Control Systems Magazine*, 26(3):96–114, June 2006. ISSN 1941-000X. doi: 10.1109/MCS.2006.1636313.

Steven L. Brunton and J. Nathan Kutz. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control. https://www.cambridge.org/highereducation/books/data-driven-science-and-engineering/6F9A730B7A9A9F43F68CF21A24BEC339, May 2022.

Ao Buke, Fang Gaoli, Wang Yongcai, Song Lei, and Yang Zhiqi. Healthcare algorithms by wearable inertial sensors: A survey. *China Communications*, 12(4):1–12, April 2015. ISSN 1673-5447. doi: 10.1109/CC.2015.7114054.

Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. MyoSuite – A contact-rich simulation suite for musculoskeletal motor control, May 2022.

Andrea Centurelli, Luca Arleo, Alessandro Rizzo, Silvia Tolu, Cecilia Laschi, and Egidio Falotico. Closed-Loop Dynamic Control of a Soft Manipulator Using Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, 7(2):4741–4748, April 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3146903.

Long Chen, Haizhou Ai, Rui Chen, Zijie Zhuang, and Shuang Liu. Cross-View Tracking for Multi-Human 3D Pose Estimation at over 100 FPS, July 2021.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, September 2014.

Salvatore Cuomo, Vincenzo Schiano di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next, June 2022.

W. H. K. de Vries, H. E. J. Veeger, C. T. M. Baten, and F. C. T. van der Helm. Magnetic distortion in motion labs, implications for validating inertial magnetic sensors. *Gait & Posture*, 29(4):535–541, June 2009. ISSN 0966-6362. doi: 10.1016/j.gaitpost.2008.12.004.

Marc Deisenroth and Carl Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Icml 2011*, pages 465–472, January 2011.

Cosimo Della Santina, Christian Duriez, and Daniela Rus. Model-based control of soft robots: A survey of the state of the art and open challenges. *IEEE Control Systems Magazine*, 43(3): 30–65, 2023. doi: 10.1109/MCS.2023.3253419.

Scott L. Delp, Frank C. Anderson, Allison S. Arnold, Peter Loan, Ayman Habib, Chand T. John, Eran Guendelman, and Darryl G. Thelen. OpenSim: Open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on bio-medical engineering*, 54(11):1940–1950, November 2007. ISSN 0018-9294. doi: 10.1109/TBME.2007.901024.

Karsten Eckhoff, Manon Kok, Sergio Lucia, and Thomas Seel. Sparse Magnetometer-free Inertial Motion Tracking – A Condition for Observability in Double Hinge Joint Systems. *IFAC-PapersOnLine*, 53:16023–16030, January 2020. doi: 10.1016/j.ifacol.2020.12.403.

Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim, and T. Hamel. A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2008. doi: 10.1109/IROS.2008.4650766.

Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, New York, 2008. ISBN 978-0-387-74314-1 978-0-387-74315-8.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, Brian Ichter, Danny Driess, Jiajun Wu, Cewu Lu, and Mac Schwager. Foundation Models in Robotics: Applications, Challenges, and the Future, December 2023.

Oisin Fitzgerald, Oscar Perez-Concha, Blanca Gallego-Luxan, Alejandro Metke-Jimenez, Lachlan Rudd, and Louisa Jorm. Continuous time recurrent neural networks: Overview and application to forecasting blood glucose in the intensive care unit, April 2023.

Eli Friedman and Fred Fontaine. Generalizing Across Multi-Objective Reward Functions in Deep Reinforcement Learning, September 2018.

Matthew W. Givens and Calvin Coopmans. A Survey of Inertial Sensor Fusion: Applications in sUAS Navigation and Data Collection. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1054–1060, June 2019. doi: 10.1109/ICUAS.2019.8798225.

Aaron Grapentin, Dustin Lehmann, Ardjola Zhupa, and Thomas Seel. Sparse Magnetometer-Free Real-Time Inertial Hand Motion Tracking. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 94–100, September 2020. doi: 10.1109/MFI49285.2020.9235262.

Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Jan Humplik, Markus Wulfmeier, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, April 2024. doi: 10.1126/scirobotics.adi8022.

David A. Haggerty, Michael J. Banks, Ervin Kamenar, Alan B. Cao, Patrick C. Curtis, Igor Mezić, and Elliot W. Hawkes. Control of soft robots with inertial dynamics. *Science Robotics*, 8(81): eadd6864, 2023. doi: 10.1126/scirobotics.add6864.

Eni Halilaj, Apoorva Rajagopal, Madalina Fiterau, Jennifer L. Hicks, Trevor J. Hastie, and Scott L. Delp. Machine learning in human movement biomechanics: Best practices, common pitfalls, and new opportunities. *Journal of Biomechanics*, 81:1–11, November 2018. ISSN 1873-2380. doi: 10.1016/j.jbiomech.2018.09.009.

Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous Drone Racing: A Survey. *IEEE Transactions on Robotics*, 40:3044–3067, 2024. ISSN 1941-0468. doi: 10.1109/TRO.2024.3400838.

Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications, March 2023.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form Continuous-time Neural Models. *Nature Machine Intelligence*, 4(11):992–1003, November 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00556-7.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, 2015. doi: 10.48550/arXiv.1507.06527.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Shibo Zhao, Yu Quan Chong, Chen Wang, Katia Sycara, Matthew Johnson-Roberson, Dhruv Batra, Xiaolong Wang, Sebastian Scherer, Zsolt Kira, Fei Xia, and Yonatan Bisk. Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis, December 2023.

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics*, 37(6):185:1–185:15, December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275108.

Curtis Johnson, Tyler Quackenbush, Taylor Sorensen, David Wingate, and Marc Killpack. Using first principles for deep learning and model-based control of soft robots. *Frontiers in Robotics and AI*, 8, May 2021. doi: 10.3389/frobt.2021.654398.

Rudolph Emil Kalman. Contributions to the Theory of Optimal Control. *Boletin de la Sociedad Matematica Mexicana*, 5:102–119, 1960a.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960b.

Kento Kawaharazuka, Tatsuya Matsushima, Andrew Gambardella, Jiaxian Guo, Chris Paxton, and Andy Zeng. Real-World Robot Applications of Foundation Models: A Review, February 2024.

Patrick Kidger. On Neural Differential Equations, February 2022.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural Controlled Differential Equations for Irregular Time Series. In *Advances in Neural Information Processing Systems*, volume 33, pages 6696–6707. Curran Associates, Inc., 2020.

Manon Kok, Jeroen D. Hol, and Thomas B. Schön. An optimization-based approach to human body motion capture using inertial sensors. *IFAC Proceedings Volumes*, 47(3):79–85, January 2014. ISSN 1474-6670. doi: 10.3182/20140824-6-ZA-1003.02252.

Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, April 2018.

Daniel Laidig and Thomas Seel. VQF: Highly accurate IMU orientation estimation with bias estimation and magnetic disturbance rejection. *Information Fusion*, 91:187–204, March 2023. ISSN 1566-2535. doi: 10.1016/j.inffus.2022.10.014.

Daniel Laidig, Thomas Schauer, and Thomas Seel. Exploiting kinematic constraints to compensate magnetic disturbances when calculating joint angles of approximate hinge joints from orientation estimates of inertial sensors. In *2017 International Conference on Rehabilitation Robotics (ICORR)*, pages 971–976, July 2017. doi: 10.1109/ICORR.2017.8009375.

Daniel Laidig, Dustin Lehmann, Marc-André Bégin, and Thomas Seel. Magnetometer-free Realtime Inertial Motion Tracking by Exploitation of Kinematic Constraints in 2-DoF Joints. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1233–1238, July 2019. doi: 10.1109/EMBC.2019.8857535.

Huu Minh Le, Thanh Nho Do, and Soo Jay Phee. A survey on actuators-driven surgical robots. *Sensors and Actuators A: Physical*, 247:323–354, August 2016. ISSN 0924-4247. doi: 10.1016/j.sna.2016.06.010.

Dustin Lehmann, Daniel Laidig, Raphael Deimel, and Thomas Seel. Magnetometer-free inertial motion tracking of arbitrary joints with range of motion constraints. *IFAC-PapersOnLine*, 53(2):16016–16022, January 2020a. ISSN 2405-8963. doi: 10.1016/j.ifacol.2020.12.401.

Dustin Lehmann, Daniel Laidig, and Thomas Seel. Magnetometer-free motion tracking of one-dimensional joints by exploiting kinematic constraints. *Proceedings on Automation in Medical Engineering*, 1(1):027–027, February 2020b.

Irvin Hussein López-Nava and Angélica Muñoz-Meléndez. Wearable Inertial Sensors for Human Motion Analysis: A Review. *IEEE Sensors Journal*, 16(22):7821–7834, November 2016. ISSN 1558-1748. doi: 10.1109/JSEN.2016.2609392.

Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A Comprehensive Survey of Visual SLAM Algorithms. *Robotics*, 11(1):24, February 2022. ISSN 2218-6581. doi: 10.3390/robotics11010024.

Sebastian O H Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. 2010.

Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear Complementary Filters on the Special Orthogonal Group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, June 2008. ISSN 1558-2523. doi: 10.1109/TAC.2008.923738.

Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-Real Reinforcement Learning for Deformable Object Manipulation, October 2018.

Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. XNect: Real-time Multi-Person 3D Motion Capture with a Single RGB Camera. In *ACM Transactions on Graphics (SIGGRAPH 2020)*, 2020.

Michael Meindl, Dustin Lehmann, and Thomas Seel. Bridging reinforcement learning and iterative learning control. *Frontiers in Robotics and AI*, 9, 2022. ISSN 2296-9144. doi: 10.3389/frobt.2022.793512.

Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, June 2023. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2023.3270034.

Domen Novak and Robert Riener. A survey of sensor fusion methods in wearable robotics. *Robotics and Autonomous Systems*, 73:155–170, November 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2014.08.012.

Fredrik Olsson, Manon Kok, Thomas Seel, and Kjartan Halvorsen. Robust Plug-and-Play Joint Axis Estimation Using Inertial Sensors. *Sensors*, 20(12):3534, January 2020. ISSN 1424-8220. doi: 10.3390/s20123534.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand, October 2019a.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, December 2019b.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting Recurrent Neural Networks for Long Sequences, March 2023.

Apoorva Rajagopal, Christopher L. Dembia, Matthew S. DeMers, Denny D. Delp, Jennifer L. Hicks, and Scott L. Delp. Full-Body Musculoskeletal Model for Muscle-Driven Simulation of Human Gait. *IEEE transactions on bio-medical engineering*, 63(10):2068–2079, October 2016. ISSN 1558-2531. doi: 10.1109/TBME.2016.2586891.

Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. RL-CycleGAN: Reinforcement Learning Aware Simulation-To-Real, June 2020.

Anton Maximilian Schäfer. *Reinforcement Learning with Recurrent Neural Networks*. PhD thesis, 2008.

Gerrit Schoettler, Ashvin Nair, Juan Aparicio Ojea, Sergey Levine, and Eugen Solowjow. Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks, May 2020.

E. Schuitema. *Reinforcement Learning on Autonomous Humanoid Robots*. PhD thesis, 2012.

Thomas Seel and Stefan Ruppin. Eliminating the Effect of Magnetic Disturbances on the Inclination Estimates of Inertial Sensors**This work was conducted within the research project BeMobil, which is supported by the German Federal Ministry of Research and Education (FKZ 16SV7069K). *IFAC-PapersOnLine*, 50(1):8798–8803, July 2017. ISSN 2405-8963. doi: 10.1016/j.ifacol.2017.08.1534.

Thomas Seel, Manon Kok, and Ryan S. McGinnis. Inertial Sensors—Applications and Challenges in a Nutshell. *Sensors*, 20(21):6221, January 2020. ISSN 1424-8220. doi: 10.3390/s20216221.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162.

C.E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, January 1949. ISSN 2162-6634. doi: 10.1109/JRPROC.1949.232969.

Luke Sy, Michael Raitor, Michael Del Rosario, Heba Khamis, Lauren Kark, Nigel H. Lovell, and Stephen J. Redmond. Estimating Lower Limb Kinematics Using a Reduced Wearable Sensor Count. *IEEE transactions on bio-medical engineering*, 68(4):1293–1304, April 2021. ISSN 1558-2531. doi: 10.1109/TBME.2020.3026464.

Bertram Taetz, Gabriele Bleser, and Markus Miezal. Towards Self-Calibrating Inertial Body Motion Capture, June 2016.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust Multitask Reinforcement Learning, July 2017.

Ibrahim Tijjani, Shivesh Kumar, and Melya Boukheddimi. A Survey on Design and Control of Lower Extremity Exoskeletons for Bipedal Walking. *Applied Sciences*, 12(5):2395, January 2022. ISSN 2076-3417. doi: 10.3390/app12052395.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 Iros*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Guillem Torrente, Elia Kaufmann, Philipp Foehn, and Davide Scaramuzza. Data-Driven MPC for Quadrotors. *CoRR*, 2021. doi: 10.48550/arXiv.2102.05773.

Ryan L. Truby, Michael Wehner, Abigail K. Grosskopf, Daniel M. Vogt, Sebastien G. M. Uzel, Robert J. Wood, and Jennifer A. Lewis. Soft Somatosensitive Actuators via Embedded 3D Printing. *Advanced Materials*, 30(15):1706383, 2018. ISSN 1521-4095. doi: 10.1002/adma.201706383.

Matthew Trumble, Andrew Gilbert, Adrian Hilton, and John Collomosse. Deep Convolutional Networks for Marker-less Human Pose Estimation from Multiple Views. In *Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016)*, CVMP '16, pages 1–9, New York, NY, USA, December 2016. Association for Computing Machinery. ISBN 978-1-4503-4744-0. doi: 10.1145/2998559.2998565.

Tom Van Wouwe, Seunghwan Lee, Antoine Falisse, Scott Delp, and C. Karen Liu. Diffusion Inertial Poser: Human Motion Reconstruction from Arbitrary Sparse IMU Configurations, August 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023.

Timo von Marcard, Bodo Rosenhahn, Michael J. Black, and Gerard Pons-Moll. Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs, March 2017.

James Walker, Thomas Zidek, Cory Harbel, Sanghyun Yoon, F. Sterling Strickland, Srinivas Kumar, and Minchul Shin. Soft robotics: A review of recent developments of pneumatic soft actuators. *Actuators*, 9(3), 2020. ISSN 2076-0825. doi: 10.3390/act9010003.

Daniel Weber, Clemens Gühmann, and Thomas Seel. RIANN—A Robust Neural Network Outperforms Attitude Estimation Filters. *AI*, 2(3):444–463, September 2021. ISSN 2673-2688. doi: 10.3390/ai2030028.

Ive Weygers, Manon Kok, Henri De Vroey, Tommy Verbeerst, Mark Versteyhe, Hans Hallez, and Kurt Claeys. Drift-Free Inertial Sensor-Based Joint Kinematics for Long-Term Arbitrary Movements. *IEEE Sensors Journal*, 20(14):7969–7979, 2020. ISSN 1530-437X. doi: 10.1109/JSEN.2020.2982459.

Ive Weygers, Dustin Lehmann, Simon Bachhuber, Daniel Laidig, and Thomas Seel. Do we still need magnetometers for inertial motion tracking? *Proceedings on Automation in Medical Engineering*, 2(1):763–763, March 2023.

D. Wierstra, A. Forster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 18(5):620–634, October 2010. ISSN 1367-0751, 1368-9894. doi: 10.1093/jigpal/jzp049.

Charence Wong, Zhi-Qiang Zhang, Benny Lo, and Guang-Zhong Yang. Wearable Sensing for Solid Biomechanics: A Review. *IEEE Sensors Journal*, 15(5):2747–2760, May 2015. ISSN 1558-1748. doi: 10.1109/JSEN.2015.2393883.

Xinyu Yi, Yuxiao Zhou, and Feng Xu. TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors, May 2021.

Xinyu Yi, Yuxiao Zhou, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Feng Xu. Physical Inertial Poser (PIP): Physics-aware Real-time Human Motion Tracking from Sparse Inertial Sensors, March 2022.

Meng Yuan, Chris Manzie, Malcolm Good, Iman Shames, Lu Gan, Farzad Keynejad, and Troy Robinette. A review of industrial tracking control algorithms. *Control Engineering Practice*, 102:104536, September 2020. ISSN 0967-0661. doi: 10.1016/j.conengprac.2020.104536.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022.

Zheng-Meng Zhai, Mohammadamin Moradi, Ling-Wei Kong, Bryan Glaz, Mulugeta Haile, and Ying-Cheng Lai. Model-free tracking control of complex dynamical trajectories with machine learning. *Nature Communications*, 14(1):5698, September 2023. ISSN 2041-1723. doi: 10.1038/s41467-023-41379-3.

Zhi-Qiang Zhang, Xiao-Li Meng, and Jian-Kang Wu. Quaternion-Based Kalman Filter With Vector Selection for Accurate Orientation Tracking. *IEEE Transactions on Instrumentation and Measurement*, 61(10):2817–2824, October 2012. ISSN 1557-9662. doi: 10.1109/TIM.2012.2196397.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, December 2020. doi: 10.1109/SSCI47803.2020.9308468.

Zhaolong Zheng, Hao Ma, Weichao Yan, Haoyang Liu, and Zaiyue Yang. Training Data Selection and Optimal Sensor Placement for Deep-Learning-Based Sparse Inertial Sensor Human Posture Reconstruction. *Entropy*, 23(5):588, May 2021. ISSN 1099-4300. doi: 10.3390/e23050588.

Part II

Publications

Paper A

RNN-based Observability Analysis for Magnetometer-Free Sparse Inertial Motion Tracking

Authors: Bachhuber, Simon and Weber, Daniel and Weygers, Ive and Seel, Thomas

Published in: International Conference on Information Fusion

Publication date: 09 August 2022

DOI: https://doi.org/10.23919/FUSION49751.2022.9841375

RNN-based Observability Analysis for Magnetometer-Free Sparse Inertial Motion Tracking

Simon Bachhuber*, Daniel Weber†, Ive Weygers*, and Thomas Seel*
* Department of Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany
Email: {simon.bachhuber,ive.weygers,thomas.seel}@fau.de
† Department of Energy and Automation Technology
Technische Universität Berlin, 10587 Berlin, Germany
Email: d.weber.1@tu-berlin.de

***Abstract*—Inertial measurement units are widely used for motion tracking of kinematic chains in numerous applications. While magnetometer-free sensor fusion enables reliably high accuracy in indoor environments and near magnetic disturbances, the use of sparse sensor setups would yield additional advantages in cost, effort, and usability. However, it is unclear which sparse sensor setups can be used to track which motions of which kinematic chains, since observability of the underlying nonlinear dynamics is barely understood to date. We propose a method that utilizes recurrent neural networks (RNNs) and automatically generated training data to assess the observability of the relative pose of kinematic chains in sparse inertial motion tracking (IMT) systems. We apply this method to a range of double-hinge-joint systems that perform fully-exciting random motion. Results show how the degree of observability depends on the kinematic structure and that RNN-based observers can achieve small tracking errors in a large range of sparse and magnetometer-free setups. The proposed methods enable systematic assessment of observability properties in complex nonlinear dynamics and represent a key step toward enabling reliably accurate and non-restrictive IMT solutions.**

I. Introduction

In recent years, technological advances in microelectromechanical systems have resulted in smaller, more affordable inertial measurements units (IMUs). As a result, IMUs are now widely applied in a broad range of application domains [1], which includes autonomous driving [2], aerospace engineering [3], and health applications [4], [5]. In many of these applications IMUs are used to track the motion of some sort of kinematic chain that consists of multiple rigid segments connected by joints. One IMU is attached to each segment, and the 3D accelerometer, 3D gyroscope and 3D magnetometer readings are fused to estimate the orientation and position of all segments.

However, inhomogeneous magnetic fields in indoor environments and in proximity of ferromagnetic material or electric devices severely degrade the accuracy of orientation estimates in real-world scenarios [6]–[9]. To address this challenge, several 6D sensor fusion methods were proposed that omit magnetometer readings but compensate the loss of information by exploiting kinematic constraints, see e.g. [10]–[15]. While such methods enable magnetometer-free relative-pose estimation in kinematic chains, they require a full IMU setup with one IMU per segment.

The use of *sparse* sensor setups, as illustrated in Figure 1, would lead to reduced effort and cost, and thus increased usability, in many applications. Despite the inherent challenges of this approach, promising initial results have been published in recent years at least for magnetometer-dependent 9D fusion, see [16]–[20]. It was found that sparse sensor setups generally lead to non-uniformly observable systems, in which multiple poses or motions generate the same measurements and the relative pose can only be determined when the performed motion provides sufficient excitation. As a consequence, estimation errors may vary largely and unpredictably over time. This effect is even stronger when magnetometer readings are omitted, and it explains why combining the advantages of 6D and sparse IMT has rarely been achieved so far [21].

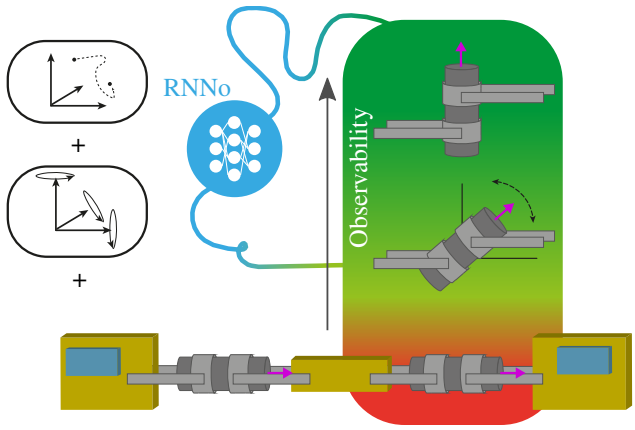


Fig. 1. An RNN-based observer is used to investigate observability in sparse inertial motion tracking systems that perform random translations and rotations. The RNN-based observer's ability to track the relative orientations of the kinematic chain increases as we change the kinematic structure such that the second hinge joint axis is non-parallel to the first hinge joint axis.

In both the 6D and the 9D case, the underlying dynamics are highly nonlinear, and – to the best of our knowledge – no formal observability analysis has been presented to date. This is true with the exception of a single conference paper that considered the special case of a double-hinge-joint sytem

with non-parallel joint axes directions and finds one sufficient but not necessary condition for observability via formulation of an explicit joint constraint [22].

Despite the value of this initial result, it remains generally unclear which sparse sensor setups can be used to track which motions of which kinematic chains and how to systematically answer this question. In the present contribution, we address this question via a neural-network-based approach.

Machine learning has been previously applied to orientation estimation in, e.g. [23], [24], and in particular in combination with sparse IMU placement in, e.g. [17], [20], [25]. Moreover, neural networks have been used for state estimation and observability analysis in [26], [27]. In the present work, we leverage the universal approximation properties of neural networks to answer the question whether the system states can be uniquely inferred from (a sequence of) the measurements of a magnetometer-free sparse IMT system.

More specifically, we propose a method that proves observability by example [28], i.e. by showing that an observer exists. Instead of investigating only observability that is induced by explicitly modeled joint constraints, we propose a purely data-driven approach that trains a recurrent neural network observer on automatically generated training data. We then apply that method to example setups with two IMUs attached to the outer segments of a three-link kinematic chain that performs fully exciting random motions, as illustrated in Figure 1. We show that the learning convergence behaviour is remarkably different in observable versus non-observable systems, and we investigate how the observability properties of the given system are influenced by its kinematic structure.

II. Problem Formulation

In this section, we define and explain the magnetometer-free sparse IMT problem for a general kinematic chain. While the following description is given for a kinematic chain with three segments and two sensors, as illustrated in Figure 2, the considerations directly extend to other sparse setups with a different number of segments and sensors.

Each IMU provides measurements of the three-dimensional angular rate and acceleration, while magnetometer readings are omitted for reasons given in Section I. The relative pose of the kinematic chain, i.e. the relative orientation of each segment with respect to its neighboring segments, shall then be determined from these magnetometer-free and sparse measurements. For the sake of brevity, we assume all sensor-to-segment orientations [14], [29]–[31] to be identity.

Both 6D measurements are combined into one measurement signal $\boldsymbol{y}_t \in \mathbb{R}^{12}$ defined as

$$\boldsymbol{y}_t := (\boldsymbol{\omega}_1(t)^\mathsf{T}, \boldsymbol{\rho}_1(t)^\mathsf{T}, \boldsymbol{\omega}_3(t)^\mathsf{T}, \boldsymbol{\rho}_3(t)^\mathsf{T})^\mathsf{T} \quad \forall t \tag{1}$$

where $\boldsymbol{\omega}_1(t), \boldsymbol{\rho}_1(t)$ and $\boldsymbol{\omega}_3(t), \boldsymbol{\rho}_3(t)$ denote gyroscope and accelerometer measurements at time t for the first IMU (on segment $\mathcal{S}_1$) and the second IMU (on segment $\mathcal{S}_3$), respectively. The relative pose of the kinematic chain is fully determined by the state $\boldsymbol{x}_t \in \mathbb{R}^8$ with

$$\boldsymbol{x}_t := \left(\boldsymbol{x}_t^{(l)\mathsf{T}}, \boldsymbol{x}_t^{(r)\mathsf{T}}\right)^\mathsf{T} := \left({}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q}^\mathsf{T}, {}^{\mathcal{S}_3(t)}_{\mathcal{S}_2(t)}\mathbf{q}^\mathsf{T}\right)^\mathsf{T} \quad \forall t \tag{2}$$

where $\mathcal{S}_1(t)$, $\mathcal{S}_2(t)$ and $\mathcal{S}_3(t)$ are the sensor/segment coordinate systems (or frames) at time t. Then, ${}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q} \in \mathbb{H}$ and ${}^{\mathcal{S}_3(t)}_{\mathcal{S}_2(t)}\mathbf{q} \in \mathbb{H}$ are the two unit quaternions that describe the orientation between segment one and segment two and between segment three and segment two, respectively. Here, $\mathbb{H}$ is the vector space of unit quaternions, a representation of $\mathrm{SU}(2)$.

In light of the variety of different observability definitions in literature (see e.g. [28], [32]), it is important to state that we consider a filtering problem, in which the system state $\boldsymbol{x}_t$ at any instant t must be uniquely inferred from the current and all previous measurements $\boldsymbol{y}_{1:t}$. If and only if this is feasible, we consider the system to be observable.

Although no specific criteria exist, it is known that this observability property depends both on the kinematic structure, including the degrees of freedom and rotation axes of the joints, and on the excitation provided by the performed motion. For the sake of brevity, we eliminate the latter and focus on the former, i.e. we limit all considerations to the ideal case of maximum excitation through random motion. The main research problem is thus to find a method that enables systematic assessment of the given observability property in magnetometer-free sparse IMT setups with different kinematic structures.

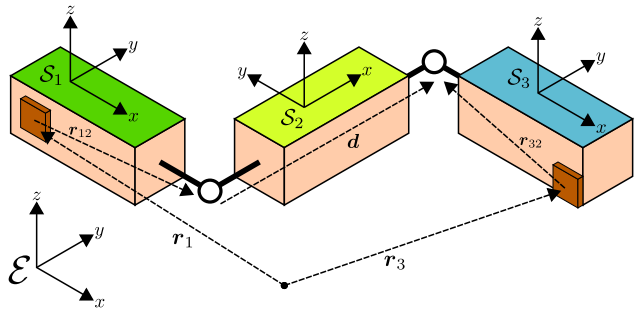


Fig. 2. Kinematic chain with three segments but only two 6D-IMUs (red boxes on first and third segment). The three segments $\mathcal{S}_1, \mathcal{S}_2$ and $\mathcal{S}_3$ are connected by two joints. The vectors ${}_{\mathcal{S}_1}\boldsymbol{r}_{12}, {}_{\mathcal{S}_2}\boldsymbol{d}, {}_{\mathcal{S}_3}\boldsymbol{r}_{32}$ are time-invariant in their local frames.

III. Methods

In this section we propose a method for solving the problem defined in Section II. The method consists of large-scale simulation of random kinematic-chain motions in order to train an RNN-based observer (RNNo) through supervised learning. We first describe how to simulate the motion and derive the training pairs, i.e. the vector of measurements (input) and the state vector (output), from it. Then, we introduce the analytic form of the RNNo. Next, we define a suitable objective function, which we then optimize to train the RNNo. Finally, we establish the relation between the RNNo's estimation capabilities and the observability of the simulated system.

A. Simulation of training data

We start by defining the required frames. The navigation frame is denoted by $\mathcal{E}$, it is time-invariant and its z-axis

is aligned with the Earth's gravity vector. The three (time-dependent) segment frames are given by $\mathcal{S}_1(t)$, $\mathcal{S}_2(t)$, and $\mathcal{S}_3(t)$. The orientations of the segments w.r.t. the navigation frame are then given by the quaternions ${}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}$, ${}^{\mathcal{S}_2(t)}_{\mathcal{E}}\mathbf{q}$, and ${}^{\mathcal{S}_3(t)}_{\mathcal{E}}\mathbf{q}$, respectively. The chain is connected at all times, i.e.

$$ {}_{\mathcal{E}}\boldsymbol{r}_3(t) = {}_{\mathcal{E}}\boldsymbol{r}_1(t) + {}_{\mathcal{E}}\boldsymbol{r}_{12}(t) + {}_{\mathcal{E}}\boldsymbol{d}(t) - {}_{\mathcal{E}}\boldsymbol{r}_{32}(t) \quad \forall t \tag{3} $$

where $\boldsymbol{r}_{12}$ denotes the vector connecting sensor one to the joint that connects segment one and two; similarly $\boldsymbol{r}_{32}$ denotes the vector connecting sensor three to the joint that connects segment three and two; $\boldsymbol{d}$ is the vector between the two joints. Finally, $\boldsymbol{r}_1$ and $\boldsymbol{r}_3$ denotes the position of sensor one and two, respectively. All three segments are assumed to be perfectly rigid. Therefore, as illustrated in Figure 2, the relative vectors that compose the chain are time-invariant in their local frame, i.e.

$$ \begin{aligned} {}_{\mathcal{E}}\boldsymbol{r}_{12}(t) &:= \left({}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}\right) \otimes {}_{\mathcal{S}_1(t)}\boldsymbol{r}_{12} \otimes \left({}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1} \quad \forall t, \\ {}_{\mathcal{E}}\boldsymbol{d}(t) &:= \left({}^{\mathcal{S}_2(t)}_{\mathcal{E}}\mathbf{q}\right) \otimes {}_{\mathcal{S}_2(t)}\boldsymbol{d} \otimes \left({}^{\mathcal{S}_2(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1} \quad \forall t, \\ {}_{\mathcal{E}}\boldsymbol{r}_{32}(t) &:= \left({}^{\mathcal{S}_3(t)}_{\mathcal{E}}\mathbf{q}\right) \otimes {}_{\mathcal{S}_3(t)}\boldsymbol{r}_{32} \otimes \left({}^{\mathcal{S}_3(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1} \quad \forall t \end{aligned} \tag{4} $$

where $\otimes : \mathbb{H} \times \mathbb{H} \to \mathbb{H}$ denotes quaternion multiplication and Euclidean vectors are interpreted as pure quaternions if required. The segment lengths are assumed to be known.

With these assumptions, the motion of the chain is completely determined by, e.g. specifying

- ${}_{\mathcal{E}}\boldsymbol{r}_1(t)$ (translation of sensor one)
- ${}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}$ (rotation of sensor one)
- ${}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q}$, ${}^{\mathcal{S}_3(t)}_{\mathcal{S}_2(t)}\mathbf{q}$ (relative rotation).

We generate ${}_{\mathcal{E}}\boldsymbol{r}_1(t)$ and ${}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}$ by randomly sampling position and orientation changes as well as the corresponding time intervals from uniform distributions, such that velocities and angular velocities are bounded from above. Cosine interpolation is used, and random distortions are added to avoid any regular interpolation patterns that the network might learn to exploit. Figure 3 shows a randomly chosen example motion.

Double hinge joint systems impose the following model for inter-segment rotations

$$ \begin{aligned} {}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q} &:= \left(\cos\tfrac{\alpha^{(l)}(t)}{2}, \sin\tfrac{\alpha^{(l)}(t)}{2}\, {}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)\,\intercal}\right)^{\intercal} \quad \forall t, \\ {}^{\mathcal{S}_3(t)}_{\mathcal{S}_2(t)}\mathbf{q} &:= \left(\cos\tfrac{\alpha^{(r)}(t)}{2}, \sin\tfrac{\alpha^{(r)}(t)}{2}\, {}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(r)\,\intercal}\right)^{\intercal} \quad \forall t, \end{aligned} \tag{5} $$

which are parameterized by the joint angles $\alpha^{(l)}(t)$, $\alpha^{(r)}(t) \in \mathbb{R}$ and the joint axes ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)}$, ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(r)} \in \mathbb{R}^3$. We sample $\alpha^{(l)}(t)$ and $\alpha^{(r)}(t)$ randomly using the same routine as above. The components of the joint axes directions are fixed in $\mathcal{S}_2(t)$, and without loss of generality we model the second (or right) joint-axis direction ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(r)}$ relative to the first (or left) joint-axis direction ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)}$ as

$$ {}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(r)} := \boldsymbol{R}_z(\phi)\, {}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)} \quad \forall t \tag{6} $$

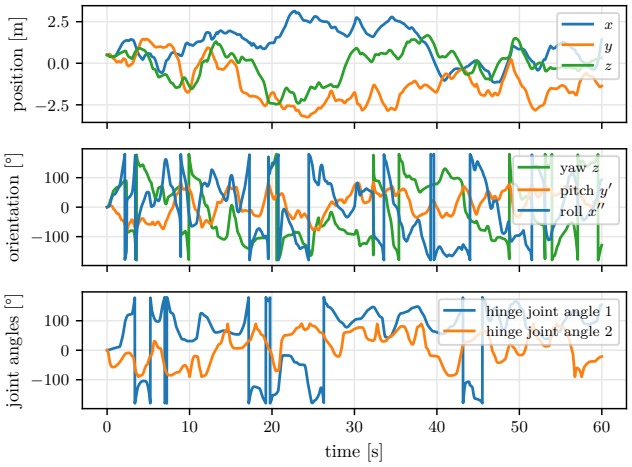


Fig. 3. Example motion for a kinematic chain with two hinge joints and perpendicular joint axes directions. The chain motion is specified using the position (top row) and orientation (middle row) of sensor one as well as both hinge joint angles (bottom row). The motion provides full random excitation.

where $\boldsymbol{R}_z(\phi)$ represents the rotation matrix with angle $\phi \in [0, 90°]$ around the unit-length z-vector. The joint axes directions are parallel for $\phi = 0$ and perpendicular for $\phi = 90°$. Variation of the left joint axis direction in combination with the angle ϕ maps out all possible double-hinge joint systems, or an equivalent representation thereof.

To enable supervised learning of the RNNo, we generate training pairs that correspond to input and ground truth output.

1) State vector: The state vector $\boldsymbol{x}_t$ from (2), required as ground truth output, is given by, e.g.

$$ \begin{aligned} {}_{\mathcal{S}_2(t)}\left({}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q}\right) &:= \left({}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1} \otimes {}^{\mathcal{S}_2(t)}_{\mathcal{E}}\mathbf{q} \quad \forall t, \\ {}_{\mathcal{S}_2(t)}\left({}^{\mathcal{S}_3(t)}_{\mathcal{S}_2(t)}\mathbf{q}\right) &:= \left({}^{\mathcal{S}_3(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1} \otimes {}^{\mathcal{S}_2(t)}_{\mathcal{E}}\mathbf{q} \quad \forall t. \end{aligned} \tag{7} $$

2) Measurement model: The measurement vector $\boldsymbol{y}_t$ from (1), required as input, is composed of simulated gyroscope and accelerometer measurements. The gyroscope of sensor one measures (compare [33])

$$ {}_{\mathcal{S}_1(t)}\boldsymbol{\omega}_1(t) :\cong \underbrace{2\, \frac{\mathrm{d}\left({}^{\mathcal{S}_1(t')}_{\mathcal{E}}\mathbf{q}\right)}{\mathrm{d}t'}\bigg|_t \otimes \left({}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1}}_{\text{true measurement}} + s_{\text{error}} \underbrace{\left(\epsilon_{\text{gyro}}(t) + \xi_{\text{gyro}}\right)}_{\text{noise and bias}} \quad \forall t \tag{8} $$

and the accelerometer of sensor one measures

$$ {}_{\mathcal{S}_1(t)}\boldsymbol{\rho}_1(t) := \underbrace{\left({}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}\right)^{-1} \otimes \left(\frac{\mathrm{d}^2\left({}_{\mathcal{E}}\boldsymbol{r}_1(t')\right)}{\mathrm{d}t'^2}\bigg|_t + {}_{\mathcal{E}}\boldsymbol{g}\right) \otimes {}^{\mathcal{S}_1(t)}_{\mathcal{E}}\mathbf{q}}_{\text{true measurement}} + s_{\text{error}} \underbrace{\left(\epsilon_{\text{acc}}(t) + \xi_{\text{acc}}\right)}_{\text{noise and bias}} \quad \forall t \tag{9} $$

	GRU-layers	Linear-layers	Parameter count
shallow	100, 100	50, 25	100 632
medium	300, 200	100, 50, 50, 25, 25	612 032
complex	400, 300	200, 100, 50, 50, 25, 25	1 216 536

TABLE I
STACKING BLOCKS WITH VARYING NUMBER OF HIDDEN NODES GIVES RISE TO THE DIFFERENT NETWORK COMPLEXITIES OF THE RNN-BASED OBSERVER.

where ${}_{\mathcal{E}}\boldsymbol{g} = (0, 0, 9.81)^\intercal \mathrm{m\,s^{-2}}$ is the gravity vector. In both latter equations $s_{\text{error}} \in \mathbb{R}$ is a scaling factor and ϵ, ξ are random variables representing noise and bias, respectively. Sensor two attached to segment three measures analogously to sensor one.

B. Architecture of the RNNo

The RNNo is responsible for estimating the system state (7) from a time-series of measurements (8), (9). We propose the following network $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathbb{R}^{t\times 12} \to \mathbb{H}^2$

$$\begin{aligned}\boldsymbol{y}_{1:t} &\to \underbrace{\text{GRU} \to \text{Layernorm} \to \text{Elu}}_{\text{repeat } n_g \text{ times}} \\ &\to \underbrace{\text{Linear} \to \text{Relu}}_{\text{repeat } n_l \text{ times}} \to \text{Linear} \\ &\to \text{Split} \to \text{Normalize} \to \hat{\boldsymbol{x}}_t \quad \forall t.\end{aligned}$$

We differentiate between three different network complexities (shallow, medium, complex) by stacking a varying number of GRU blocks and Linear blocks, with a varying number of hidden nodes per block, as summarized in Table I. The network architecture is motivated by [23] where a similar RNNo is used for attitude estimation. The choice of activation functions is based on hyperparameter optimization, where the set of candidate functions stems from analysis given in [34]. We use layer normalization (Layernorm) as described in [35] to aid recurrent network training. The eight network outputs are split and normalized to Euclidean norm of one to create two unit quaternions that represent the inter-segment orientations.

C. Objective Function

Supervised learning requires a scalar objective function that captures and quantifies the estimation goal. It is given by the expectation of the squared error signal

$$\mathcal{J}(\boldsymbol{\theta}) := \mathop{\mathbb{E}}_{(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})} \left[\sum_{i\in\{l,r\}} \sum_{t=1}^{T} e\left(\boldsymbol{x}_t^{(i)}, \hat{\boldsymbol{x}}_t^{(i)}\right)^2 \right] \tag{10}$$

where $e : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$ is a scalar error signal, which is given by

$$e(\mathbf{q}, \hat{\mathbf{q}}) := \operatorname{angle}\left(\mathbf{q} \otimes \hat{\mathbf{q}}^{-1}\right).$$

The estimated state is given by

$$\hat{\boldsymbol{x}}_t := \left(\hat{\boldsymbol{x}}_t^{(l)\intercal}, \hat{\boldsymbol{x}}_t^{(r)\intercal}\right)^\intercal := \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{y}_{1:t})$$

where $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathbb{R}^{t\times 12} \to \mathbb{H}^2$ is the RNNo, and the lower index t is the discrete-time index. The angle operation $\operatorname{angle} : \mathbb{H} \to \mathbb{R}$ extracts the smallest angle of rotation between a unit quaternion and the identity quaternion. It is given by

$$\operatorname{angle}(\mathbf{q}) := 2\arccos(\mathbf{q}[1]) \cong 2\arctan\left(\frac{\sqrt{1-\mathbf{q}[1]^2}}{\mathbf{q}[1]}\right)$$

where $\mathbf{q}[i]$ is used to denote the i-th component of the unit quaternion $\mathbf{q}$. The equivalent expression utilising $\arctan$ is preferred due to numerical advantages in the context of gradient-based optimization.

D. Network Training

The simulated data (cf. Section III-A) is used to compute the objective function (cf. Section III-C), which we can minimize in an iterative manner to obtain a trained RNNo (cf. Section III-B).

The parameter vector $\boldsymbol{\theta}$ is iteratively updated by generating a batch of training data, computing the objective function, and performing an optimization step. We refer to this process as one batch generation and denote the parameter vector after, e.g., 100 batch generations by $\boldsymbol{\theta}_{100}$. However, due to the usage of truncated backpropagation through time (TBPTT), one batch generation incorporates multiple parameter updates. At every batch generation we estimate the expectation in (10) as an arithmetic mean of the error over 2048 sequences. Every sequence contains 60 s of data at 100 Hz, resulting in sequences of length $T = 6000$. The 2048 sequences are obtained by sampling 1024 sequences from the simulated distribution and adding additional 1024 sequences with $\alpha^{(l)}$ and $\alpha^{(r)}$ interchanged. This approach avoids overfitting, a common issue in supervised machine learning [36], as we are able to sample indefinitely many training- and validation data points from an identical distribution. Note that after 1000 batch generations the network has already seen approximately four years of chain motion ($\frac{1000\cdot 2048\cdot 60\,\mathrm{s}}{365\cdot 24\cdot 3600\,\mathrm{s\ year^{-1}}}$). Data simulation at this scale is possible due to utilization of GPUs. The JAX library [37] is used to implement an efficient, JIT-compiled, GPU-accelerated data simulation. We also bypass CPU-to-GPU transfer overhead. The network is implemented using the Haiku library by DeepMind [38], which is itself built on top of JAX.

For optimization of the objective function we use the Adam optimizer combined with constant and adaptive gradient clipping, a decaying learning rate, lookahead, and TBPTT. Details are given in Table II. We use a decaying learning rate for faster convergence. To overcome the exploding/vanishing gradient problem common in the training of RNNs, TBPTT is applied. Borrowing the notation from [39], we utilize TBPTT(10 s, 10 s), i.e. gradients are stopped and applied after 10 s instead of the total length of 60 s. This results in every batch generation corresponding to six parameter updates, or in 1500 batch generations corresponding to 9000 parameter updates. Finally, recall from Section III-B that the network output is normalized to represent two unit quaternions. To encourage the network to normalize naturally, we regularize on the absolute deviation to unity of the normalization operation with $\lambda = 0.1$ as proposed by [24].

gradient clipping	0.3
norm-based gradient clipping	0.25
cosine decaying learning rate	3×10^{-3} to 3×10^{-10} over 9000 updates
lookahead	step size = 0.7 every 6 updates
adam optimizer	$b_1 = 0.99$, $b_2 = 0.999$

TABLE II
SUMMARY OF DIFFERENT 1ST-ORDER OPTIMIZATION TECHNIQUES THAT ARE DEPLOYED FOR OPTIMIZATION OF THE OBJECTIVE FUNCTION.

E. Assessing observability

As discussed in Section I, it is generally intractable to formally analyze the observability for the complex nonlinear dynamics of sparse IMT. A central idea of the present contribution is to gauge observability of such systems through the learning progress and performance of an RNNo. Specifically, we quantify the RNNo's performance using the following measures. The root-mean-squared angle error (RMSE) between the estimated relative orientation $\hat{\boldsymbol{x}}_t^{(i)}$ of the i (left or right) joint and the corresponding ground truth $\boldsymbol{x}_t^{(i)}$ is given by

$$\mathcal{R}_k^{(i)} := \mathcal{R}^{(i)}(\boldsymbol{\theta}_k) := \left(\mathop{\mathbb{E}}_{(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})} \left[\frac{1}{T} \sum_{t=1}^{T} e\left(\boldsymbol{x}_t^{(i)}, \hat{\boldsymbol{x}}_t^{(i)}\right)^2 \right] \right)^{0.5} \quad \forall i \in \{l, r\}$$

and the 90-percentile of the absolute angle error is given by

$$\mathcal{Q}_k^{(i)} := \mathcal{Q}^{(i)}(\boldsymbol{\theta}_k) := \mathop{\mathbb{E}}_{(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})} \left[\text{percentile}_t \left[\left| e\left(\boldsymbol{x}_t^{(i)}, \hat{\boldsymbol{x}}_t^{(i)}\right) \right| \right] \right] \quad \forall i \in \{l, r\}.$$

Here, $\boldsymbol{\theta}_k$ denotes the parameter vector after $k \in \mathbb{N}$ batch generations, and the expectation is estimated (at every batch generation) by sampling 2048 additional sequences, i.e. training data is not reused. We refer to these measures after 1400 batch generations as the residual error.

We then use the following arguments to investigate whether some given sparse IMT system $\mathcal{A}$ is observable for maximally exciting motion (cf. Section II). Moreover, we propose to use the RNNo's performance as a measure of the degree of observability, which – similar to the concept of observability Gramians – enables quantitative comparison of two observable sparse IMT systems.

Argument 1 (Non-observability) *If $\mathcal{A}$ is non-observable, then the proposed RNNo cannot converge to low residual error, even if the amount of training data is increased, even if the parameter count of the RNN is increased, and even if the noise and bias levels are reduced.*

Argument 2 (Observability) *If an RNNo trained with data from $\mathcal{A}$ converges to a small residual error, then observability of $\mathcal{A}$ is proven by example [28]. In such a case, this residual error should exhibit some dependence on the RNN's parameter count, the amount of training data, and the noise and bias levels.*

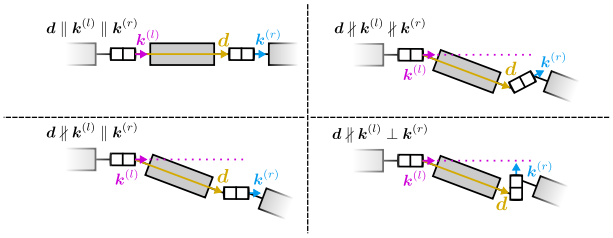


Fig. 4. Four different double-hinge-joint systems corresponding to different realizations of the joint axes directions.

Argument 3 (Degrees of observability) *Let $\mathcal{A}, \mathcal{B}$ be observable as defined in Argument 2. If the same RNNo achieves a lower residual error when trained on data from $\mathcal{A}$ than when trained on data from $\mathcal{B}$, then $\mathcal{A}$ is said to have a higher degree of observability than $\mathcal{B}$.*

IV. OBSERVABILITY ANALYSIS

In this section we investigate the observability of different realizations of the general IMT problem of Section II using the proposed method from Section III.

We start off by introducing a double-hinge-joint system that is guaranteed to be non-observable and use it to exemplify Argument 1. Then, using Argument 2 and Argument 3, we show that all other double-hinge-joint systems are observable but not to the same degree. Finally, we showcase the RNNo's estimation capabilities in more complex systems involving joints with higher degrees of freedom.

For double-hinge-joint systems, the simulation is parameterized by the chain dimensions defined in (4), and by choosing the distribution of the noise and bias random variables defined in the measurement model of (9), (8). Their numerical values are given in Table III. Finally, different kinematic structures are considered by varying the direction of the left joint axis ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)}$ and the angle $\phi \in [0, 90°]$ as defined in (6). We relate to the following four joint axes combinations using the following abbreviations:

- $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$: Middle segment is parallel to the left joint axis, which is parallel to the right joint axis, i.e. ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)} = (1, 0, 0)^\intercal$ and $\phi = 0$,
- $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$: Middle segment is not parallel to left joint axis, which is parallel to right joint axis, i.e. ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)} = \frac{1}{\sqrt{3}}(1, 1, 1)^\intercal$ and $\phi = 0°$,
- $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$: Middle segment is not parallel to left joint axis, which is not parallel to right joint axis, i.e. ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)} = \frac{1}{\sqrt{3}}(1, 1, 1)^\intercal$ and $\phi \in (0, 90°)$,
- $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$: Middle segment is not parallel to left joint axis, which is perpendicular to right joint axis, i.e. ${}_{\mathcal{S}_2(t)}\boldsymbol{k}^{(l)} = \frac{1}{\sqrt{3}}(1, 1, 1)^\intercal$ and $\phi = 90°$.

These four kinematic structures are illustrated in Figure 4.

A. Observability as a binary system property

In the case $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$, exchanging joint angles $\alpha^{(l)}$ and $\alpha^{(r)}$ leads to a different state $\boldsymbol{x}_t$ while the mea-

$\mathcal{S}_1(t)\boldsymbol{r}_{12}$	$(0.5, 0.5, 0.5)^\intercal \mathrm{m}$
$\mathcal{S}_2(t)\boldsymbol{d}$	$(0.5, 0, 0)^\intercal \mathrm{m}$
$\mathcal{S}_3(t)\boldsymbol{r}_{32}$	$(0.5, 0.5, 0.5)^\intercal \mathrm{m}$
gyroscope noise $\epsilon_{\text{gyro}}(t)$	$\mathcal{N}\left(0, \left(1^\circ\,\mathrm{s}^{-1}\right)^2\right)$
gyroscope bias ξ_{gyro}	$\mathcal{U}\left(-1^\circ\,\mathrm{s}^{-1}, 1^\circ\,\mathrm{s}^{-1}\right)$
accelerometer noise $\epsilon_{\text{acc}}(t)$	$\mathcal{N}\left(0, \left(0.5\,\mathrm{m\,s}^{-2}\right)^2\right)$
accelerometer bias ξ_{acc}	$\mathcal{U}\left(-0.5\,\mathrm{m\,s}^{-2}, 0.5\,\mathrm{m\,s}^{-2}\right)$

TABLE III
DIMENSIONS OF SIMULATED KINEMATIC CHAIN AND SIMULATED MEASUREMENT ERRORS OF THE TWO IMUS.

system \ s_{error}	2.0	1.0	0.1
$\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$	1.60 ± 0.07	1.25 ± 0.01	1.17 ± 0.09
$\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$	3.37 ± 0.24	2.34 ± 0.01	1.66 ± 0.05
$\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$	61.61 ± 0.25	61.61 ± 0.25	61.61 ± 0.25

TABLE IV
IMPACT OF NOISE AND BIAS LEVELS ON RESIDUAL ERROR FOR DOUBLE-HINGE-JOINT SYSTEMS. MEAN AND STANDARD DEVIATION OF THE RMSE $\mathcal{R}^{(l)}_{1400:1500}$ IN DEGREES.

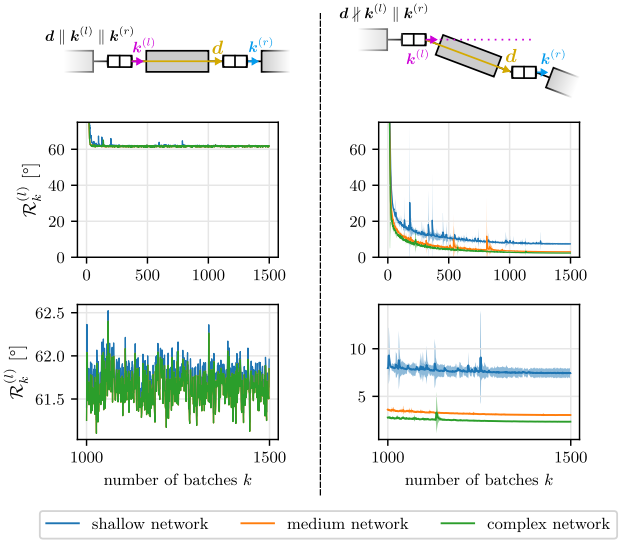


Fig. 5. RMSE $\mathcal{R}_k^{(l)}$ as a function of the number of (processed) batches k. Second-row plots zoom into last 500 batch generations, right-column plots show mean values ± 2 empirical standard deviations of 3 seeds. Errors saturate regardless of network complexity for the non-observable case $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ *(left)*, while errors decrease arbitrarily (up to some residual error due to noise and bias) for the observable case $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ *(right)*.

surement $\boldsymbol{y}_t$ remains unchanged. Therefore, it is a non-observable system. The same statement cannot be made about $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$. Figure 5 compares the estimation error $\mathcal{R}_k^{(l)}$ between $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ and $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ as a function of the number of batch generations k. The error of the non-observable system stagnates at a much higher residual error. Increasing the network's parameter count has no effect. This is in complete contrast to the right subplots, where the residual error is not only lower to begin with but also decreases as we increase the network's complexity.

Figure 6 compares one exemplary time series of one joint angle estimation error obtained by the complex network after 1500 batch generations for $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ versus $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$, which clearly shows large error spikes only for the non-observable system. In Tables IV and V we find that neither decreasing the noise and bias scale parameter s_{error} nor increasing the batch size has an effect on the residual error for $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$. The opposite behaviour can be observed for $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$. These findings confirm Argument 1, and it follows from Argument 2 that $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ is an observable system.

B. Degrees of observability: Observability as a non-binary system property

In the previous section we have established that $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ is observable. A similar argument can be made to show that $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$ is observable. In fact, the rightmost column of Table V shows that $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$ reaches a lower residual error than $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$. From Argument 3 it follows that $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$ has a higher degree of observability

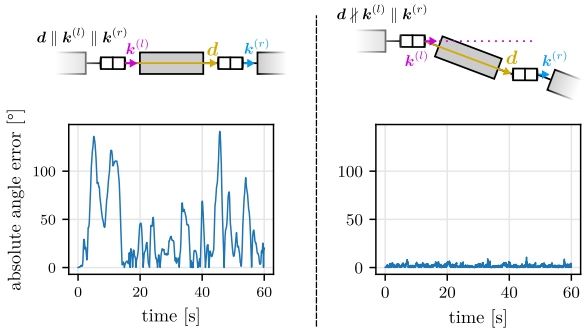


Fig. 6. Absolute angle error of the relative-orientation estimate between segment one and two as a function of time for a randomly sampled motion example. The left subplot shows error spikes above 100° due to non-observability, whereas the complex network shows very good performance at all times for the observable case in the right subplot.

system \ batch size	256	1024	2048
$\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$	1.33 ± 0.04	7.49 ± 6.17	1.25 ± 0.01
$\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$	7.15 ± 4.53	2.44 ± 0.06	2.34 ± 0.01
$\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$	61.64 ± 0.77	61.58 ± 0.38	61.61 ± 0.25

TABLE V
IMPACT OF BATCH SIZE ON RESIDUAL ERROR FOR DOUBLE-HINGE-JOINT SYSTEM. MEAN AND STANDARD DEVIATION OF $\mathcal{R}^{(l)}_{1400:1500}$ IN DEGREES.

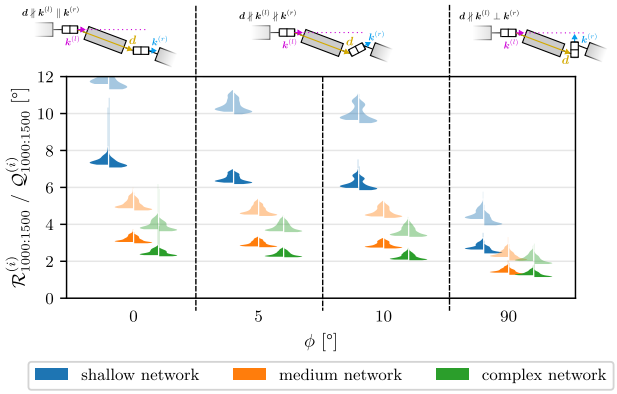


Fig. 7. Estimated distribution of the RMSE $\mathcal{R}^{(i)}_{1000:1500}$ and 90-percentile $\mathcal{Q}^{(i)}_{1000:1500}$ (low opacity) of inter-segment orientations i (left/right split) as estimated by the RNN-based observer. For each of the three network complexities, the residual error increases as ϕ decreases, i.e. as the two joint axes of the double-hinge-joint system align.

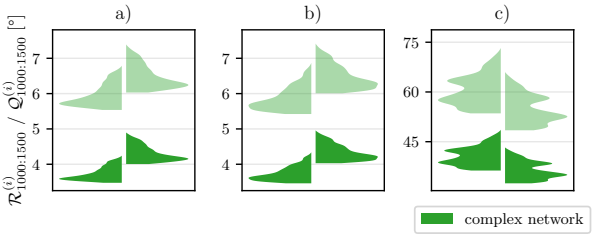


Fig. 8. Estimated distribution of the RMSE $\mathcal{R}^{(i)}_{1000:1500}$ and 90-percentile $\mathcal{Q}^{(i)}_{1000:1500}$ (low opacity) of inter-segment orientations i (left/right split) as estimated by the RNN-based observer for three different three-segment kinematic chains: *a)* biaxial joint and hinge joint with linearly dependent joint axes; *b)* biaxial joint and hinge joint with linearly independent joint axes; *c)* two triaxial joints, i.e. no restrictions on relative motions.

than $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$. Both systems represent an edge case: $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$ ($\phi = 0$) and $\boldsymbol{d} \nparallel \boldsymbol{k}^{(l)} \perp \boldsymbol{k}^{(r)}$ ($\phi = 90°$). In Figure 7 we also consider two intermediate values, namely $\phi = 5°$ and $\phi = 10°$. Figure 7 shows the estimated distribution (obtained via kernel density estimation) of $\mathcal{R}^{(i)}_{1000:1500}$ and $\mathcal{Q}^{(i)}_{1000:1500}$, i.e. using the last 500 batch generations. Note that errors in either joint, left or right, are almost identical. Systems with ϕ closer to 90° have a higher degree of observability. Note that every double-hinge-joint system, except for $\boldsymbol{d} \parallel \boldsymbol{k}^{(l)} \parallel \boldsymbol{k}^{(r)}$, is observable but not to the same degree.

C. Joints with higher degrees of freedom

In this work, double-hinge-joint systems served as a first example to demonstrate the ability of the RNNo-based approach to facilitate observability assessment. The presented approach can straightforwardly be applied to any kinematic chain that differs in, e.g. the degrees of freedom of joints, for a broad range of IMT problems. In Figure 8 we explore this idea by considering systems that involve joints with higher degrees of freedom. The RNNo achieves low estimator errors for kinematic chains composed of a hinge joint and a biaxial joint, whilst estimation errors are significantly higher for a system with two spherical joints.

V. Conclusion

In this work we presented a novel RNN-based method for observability assessment in complex nonlinear systems. The purely data-driven approach trains a recurrent neural network observer on automatically generated training data and thereby either proves observability by finding a suitable observer – or shows that the states cannot be inferred from the measurements, regardless of network complexity, training data volume, and noise/bias levels.

We successfully apply the proposed method to a range of magnetometer-free sparse IMT systems with three kinematic segments and two IMUs. Our results confirm prior findings on corner cases and demonstrate how observability properties of such analytically intractable systems depend on the kinematic structure. For this example use case, we find that all double-hinge-joint systems, except for one corner case configuration, yield observability with estimation errors in the range of $\approx 2°$ or less for fully exciting random motions.

The proposed method enables, for the first time, a systematic assessment of observability properties in sparse IMT systems. It can be used to answer the pressing question which sparse sensor setups can be used to track which motions of which kinematic chains and thus represents an important first step towards combining the advantages of sparse sensor setups and magnetometer-free sensor fusion.

Beyond the purpose of assessing observability, the RNN-based observer that is generated by the proposed method is a promising candidate for actually solving the state estimation task. Since it is designed on noise- and bias-affected simulation data, it can be expected to yield reasonable performance also on experimental data, at least for rigid mechanical setups. Our present research aims at answering this question.

Future research will also aim at using the proposed approach to investigate the observability properties of other sparse IMT setups than those considered in the present contribution. This will include setups with different numbers of segments and sensors and different combinations of joints, both for magnetometer-free fusion and for 9D IMUs.

Acknowledgements

The authors gratefully acknowledge the compute resources and support provided by NHR@FAU.

References

[1] T. Seel, M. Kok, and R. S. McGinnis, "Inertial Sensors - Applications and Challenges in a Nutshell," *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–5, 2020. DOI: 10.3390/s20216221.

[2] V. R. Marco, J. Kalkkuhl, and T. Seel, "Nonlinear observer with observability-based parameter adaptation for vehicle motion estimation," *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 60–65, 2018. DOI: 10.1016/j.ifacol.2018.09.091.

[3] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *IEEE/RSJ 2008*, 2008, pp. 340–345. DOI: 10.1109/IROS.2008.4650766.

[4] C. Wong, Z. Q. Zhang, B. Lo, and G. Z. Yang, *Wearable Sensing for Solid Biomechanics: A Review*, 2015. DOI: 10.1109/JSEN.2015.2393883.

[5] A. Buke, F. Gaoli, W. Yongcai, S. Lei, and Y. Zhiqi, *Healthcare algorithms by wearable inertial sensors: A survey*, 2015. DOI: 10.1109/CC.2015.7114054.

[6] W. De Vries, D. Veeger, C. Baten, and F. van der Helm, "Magnetic distortion in motion labs, implications for validating inertial magnetic sensors," *Gait & posture*, vol. 29, pp. 535–541, 2009. DOI: 10.1016/j.gaitpost.2008.12.004.

[7] E. Le Grand and S. Thrun, "3-Axis magnetic field mapping and fusion for indoor localization," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 358–364, 2012. DOI: 10.1109/MFI.2012.6343024.

[8] K. P. Subbu, B. Gozick, and R. Dantu, "LocateMe: Magnetic-fields-based indoor localization using smartphones," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, 2013. DOI: 10.1145/2508037.2508054.

[9] Y. Shu, C. Bo, G. Shen, C. Zhao, L. Li, and F. Zhao, "Magicol: Indoor Localization Using Pervasive Magnetic Field and Opportunistic WiFi Sensing," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1443–1457, 2015. DOI: 10.1109/JSAC.2015.2430274.

[10] D. Lehmann, D. Laidig, and T. Seel, "Magnetometer-free motion tracking of one-dimensional joints by exploiting kinematic constraints," in *Automation in Medical Engineering*, 2020. DOI: 10.18416/AUTOMED.2020.

[11] D. Laidig, D. Lehmann, M.-A. Bégin, and T. Seel, "Magnetometer-free Realtime Inertial Motion Tracking by Exploitation of Kinematic Constraints in 2-DoF Joints," in *International Conference Engineering Medicine and Biology Society*, vol. 2019, 2019, pp. 1233–1238. DOI: 10.1109/EMBC.2019.8857535.

[12] D. Lehmann, D. Laidig, R. Deimel, and T. Seel, "Magnetometer-free inertial motion tracking of arbitrary joints with range of motion constraints," in *IFAC-PapersOnLine*, vol. 53, 2020, pp. 16 016–16 022. DOI: 10.1016/j.ifacol.2020.12.401.

[13] M. Kok, J. D. Hol, and T. B. Schön, "An optimization-based approach to human body motion capture using inertial sensors," *IFAC Proceedings Volumes*, vol. 47, pp. 79–85, 2014. DOI: 10.3182/20140824-6-ZA-1003.02252.

[14] B. Taetz, G. Bleser, and M. Miezal, "Towards Self-Calibrating Inertial Body Motion Capture," *19th International Conference on Information Fusion*, 2016. DOI: 10.48550/arXiv.1606.03754.

[15] I. Weygers, M. Kok, H. De Vroey, *et al.*, "Drift-Free Inertial Sensor-Based Joint Kinematics for Long-Term Arbitrary Movements," *IEEE Sensors Journal*, vol. 20, no. 14, pp. 7969–7979, 2020. DOI: 10.1109/JSEN.2020.2982459.

[16] T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll, "Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs," *Comput. Graph. Forum*, vol. 36, no. 2, pp. 349–360, 2017. DOI: 10.1111/cgf.13131.

[17] Y. Huang, M. Kaufmann, E. Aksan, M. J. Black, O. Hilliges, and G. Pons-Moll, "Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time," in *SIGGRAPH Asia 2018*, Association for Computing Machinery, Inc, 2018. DOI: 10.1145/3272127.3275108.

[18] L. W. F. Sy, N. H. Lovell, and S. J. Redmond, "Estimating Lower Limb Kinematics Using a Lie Group Constrained Extended Kalman Filter with a Reduced Wearable IMU Count and Distance Measurements," *Sensors*, vol. 20, no. 23, 2020. DOI: 10.3390/s20236829.

[19] L. Sy, M. Raitor, M. D. Rosario, *et al.*, "Estimating Lower Limb Kinematics Using a Reduced Wearable Sensor Count," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 4, pp. 1293–1304, 2021. DOI: 10.1109/TBME.2020.3026464.

[20] Z. Zheng, H. Ma, W. Yan, H. Liu, and Z. Yang, "Training Data Selection and Optimal Sensor Placement for Deep-Learning-Based Sparse Inertial Sensor Human Posture Reconstruction," *Entropy*, vol. 23, 2021. DOI: 10.3390/e23050588.

[21] A. Grapentin, D. Lehmann, A. Zhupa, and T. Seel, "Sparse Magnetometer-Free Real-Time Inertial Hand Motion Tracking," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, vol. 2020-September, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 94–100. DOI: 10.1109/MFI49285.2020.9235262.

[22] K. Eckhoff, M. Kok, S. Lucia, and T. Seel, "Sparse Magnetometer-free Inertial Motion Tracking – A Condition for Observability in Double Hinge Joint Systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 16 023–16 030, 2020. DOI: 10.1016/j.ifacol.2020.12.403.

[23] D. Weber, C. Gühmann, and T. Seel, "RIANN—A Robust Neural Network Outperforms Attitude Estimation Filters," *AI*, vol. 2, no. 3, pp. 444–463, 2021. DOI: 10.3390/ai2030028.

[24] D. Pavllo, D. Grangier, and M. Auli, "QuaterNet: A Quaternion-based Recurrent Model for Human Motion," 2018.

[25] H. T. Butt, B. Taetz, M. Musahl, M. A. Sanchez, P. Murthy, and D. Stricker, "Magnetometer robust deep human pose regression with uncertainty prediction using sparse body worn magnetic inertial measurement units," *IEEE Access*, vol. 9, pp. 36 657–36 673, 2021. DOI: 10.1109/ACCESS.2021.3062545.

[26] W. Xiang, "Interval Observer Design of Dynamical Systems with Neural Networks," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '21, New York, NY, USA: Association for Computing Machinery, 2021. DOI: 10.1145/3447928.3456662.

[27] W. Kang, L. Xu, and H. Zhou, *The Observability in Unobservable Systems*, 2022. DOI: 10.48550/arXiv.2201.04186.

[28] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering*. Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[29] F. Olsson, M. Kok, T. Seel, and K. Halvorsen, "Robust Plug-and-Play Joint Axis Estimation Using Inertial Sensors," *Sensors*, vol. 20, no. 12, 2020. DOI: 10.3390/s20123534.

[30] L. Pacher, C. Chatellier, R. Vauzelle, and L. Fradet, "Sensor-to-Segment Calibration Methodologies for Lower-Body Kinematic Analysis with Inertial Sensors: A Systematic Review," *Sensors*, vol. 20, no. 11, 2020. DOI: 10.3390/s20113322.

[31] V. Rodrigo Marco, J. Kalkkuhl, J. Raisch, and T. Seel, "A Novel IMU Extrinsic Calibration Method for Mass Production Land Vehicles," *Sensors*, vol. 21, no. 1, 2021. DOI: 10.3390/s21010007.

[32] Besancon G., *Nonlinear Observers and Applications*. Springer Berlin, Heidelberg, 2007. DOI: 10.1007/978-3-540-73503-8.

[33] B. Graf, "Quaternions and dynamics," *arXiv e-prints*, arXiv:0811.2889, 2008.

[34] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "A Comprehensive Survey and Performance Analysis of Activation Functions in Deep Learning," *CoRR*, vol. abs/2109.14545, 2021.

[35] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," Tech. Rep., 2016. DOI: 10.48550/arXiv.1607.06450.

[36] B. Ghojogh and M. Crowley, "The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial," *ArXiv*, 2019. DOI: 10.48550/arXiv.1905.12787.

[37] J. Bradbury, R. Frostig, P. Hawkins, *et al.*, *JAX: composable transformations of Python+NumPy programs*, 2022.

[38] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin, *Haiku: Sonnet for JAX*, 2021.

[39] R. J. Williams and J. Peng, "An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories," Tech. Rep., 1990, pp. 490–501.

Paper B

Plug-and-Play Sparse Inertial Motion Tracking With Sim-to-Real Transfer

Authors: Bachhuber, Simon and Lehmann, Dustin and Doschky, Eva and Koelewijn, Anne D. and Seel, Thomas and Weygers, Ive

Published in: IEEE Sensors Letters (Volume 7)

Publication date: 21 August 2023

DOI: https://doi.org/10.1109/LSENS.2023.3307122

Sensor signal processing

Plug-and-Play Sparse Inertial Motion Tracking With Sim-to-Real Transfer

Simon Bachhuber[1*], Dustin Lehmann[2], Eva Dorschky[1], Anne D. Koelewijn[1], Thomas Seel[3], and Ive Weygers[1]

[1] *Department Artificial Intelligence in Biomedical Engineering, FAU Erlangen-Nürnberg, 91052 Erlangen, Germany*
[2] *Technical University Berlin, 10587 Berlin, Germany*
[3] *Institute of Mechatronic Systems, Leibniz Universität Hannover, 30167 Hannover, Germany*
[*] *Graduate Student Member, IEEE*

Manuscript received 18 June 2023; revised 9 August 2023; accepted 10 August 2023. Date of publication 21 August 2023; date of current version 8 September 2023.

Abstract— Inertial measurement units (IMUs) are used for inertial motion tracking (IMT) in a growing number of applications as sensor fusion methods are being advanced in three directions: magnetometer-free IMT methods that eliminate the effect of magnetic disturbances; sparse IMT approaches that lead to reduced setup complexity; and automatic self-calibration of sensor-to-segment positions or orientations. In this letter, we propose an approach that combines all three achievements and, for the first time, enables plug-and-play, magnetometer-free, and sparse IMT. This is accomplished by training a recurrent neural-network-based observer (RNNo) on just-in-time generated simulated motion data of kinematic chains. We demonstrate that domain-specific training data augmentations lead to a trained RNNo which zero shot generalizes to previously unseen experimental data and, thus, overcomes the sim-to-real gap. The trained RNNo achieves a tracking error of < 4 degrees when estimating the relative pose of a three-segment kinematic chain with two hinge joints. The proposed method offers a novel simulation-data-driven approach for solving complex sparse sensing problems while assuring robust and plug-and-play generalizability to experimental data.

Index Terms—Sensor signal processing, inertial measurement units (IMUs), magnetometer-free, recurrent neural networks, sensor fusion, sparse sensing.

I. INTRODUCTION

Recently, inertial measurement units (IMUs) have become smaller and less expensive. Consequently, they are now used in numerous application domains, ranging from autonomous driving [1], and aerospace engineering [2], [3], to health applications [4], [5], [6], [7]. In many of these applications, IMUs are used to track the motion of all components of some sort of kinematic chain that consists of rigid segments connected by joints. Typically, one IMU per segment is attached, and all gyroscope, accelerometer, and magnetometer readings are fused to estimate the pose of the chain.

The applicability of inertial motion tracking (IMT) is often limited by the three requirements of magnetometer data, full IMU setups, and calibration poses. Relying on magnetometer data limits applicability as ferromagnetic material or electronic devices, omnipresent in indoor, real-world environments, distort the magnetic field [8], and degrade subsequent pose estimates. To overcome this, several magnetometer-free methods have been proposed [9], [10]. However, they require a full IMU setup with one IMU per segment, whereas fewer sensors than segments (sparse sensor setup), as shown in Fig. 1, would lead to reduced effort and cost, and thus increased usability. Methods tailored specifically to sparse sensor setups have been developed [11], [12] that typically utilize extensive models of the system to constrain the solution to physically feasible motion states. Finally, the requirement of either precise knowledge of sensor-to-segment position and/or orientation, or of calibration motion to identify the latter [13], [14], limits the plug-and-play capabilities of IMT.

The accomplishment of *simultaneously* alleviating even two out of those three requirements has rarely been achieved. The combination

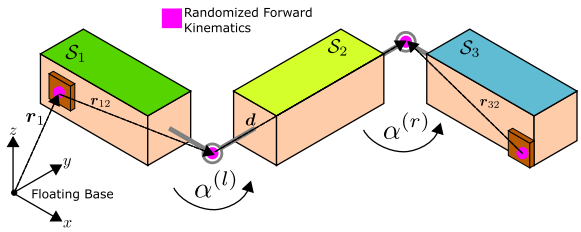


Fig. 1. Challenging IMT problem of a kinematic chain with three segments $\mathcal{S}_{1-3}$, and a sparse sensor setup consisting of only two-segment-aligned 6D-IMUs (brown boxes) on the outer segments. The segments are connected by two hinge joints with known and nonparallel joint axes' directions and joint angles $\alpha^{(l/r)}$. The physical dimensionality of the chain is assumed to be unknown and defined by $\boldsymbol{r}_{12}, \boldsymbol{d}$, and $\boldsymbol{r}_{32}$.

of magnetometer-free IMT with plug-and-play capabilities has been investigated in [9] and [10], and the combination of magnetometer-free and sparse IMT has only been achieved in [15] and [16] by requiring very specific chain configurations and knowledge on the physical dimensions of the chain, which inherently restricts the method's plug-and-play capabilities. The latter combination is especially challenging as the motion states may then be non- or only partially observable, i.e., different movements can generate the same IMU sensor data. First results on the observability properties of such IMT problems have been published in [17] and [18].

In this contribution, for the first time, we alleviate all three requirements simultaneously to achieve a first step toward magnetometer-free, sparse IMT with plug-and-play capabilities with respect to the physical dimensions of the kinematic chain and validate our approach in experiment. More specifically, we present a novel data-driven magnetometer-free approach that is able to track all relative motion states of three-segment two hinge joint systems (cf. Fig. 1) from

Corresponding author: Simon Bachhuber (e-mail: simon.bachhuber@fau.de).
Associate Editor: F. Falcone.
Digital Object Identifier 10.1109/LSENS.2023.3307122

measurements of only two IMUs, one on each of the outer segments, despite large uncertainties in the sensor-to-segment position and physical dimensionality of the chain. Our approach utilizes a recurrent neural-network-based observer (RNNo) [17] together with novel nonrestrictive priors in the form of domain-specific augmentations during in silico training, to effectively solve the sim-to-real gap. The proposed method is, furthermore, demonstrated to converge quickly and achieve a long-term stable tracking error of < 4 degrees for an unseen experimental kinematic chain in a plug-and-play manner.

II. PROBLEM FORMULATION

We consider a kinematic chain consisting of three segments with frames (coordinate systems) $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ connected by two hinge joints with known and orthogonal joint axes' directions. Note that we expect that the presented approach will still be valid for nonorthogonal nonparallel joint axes, as shown in [17]. We define plug-and-play capabilities as the requirement of unknown sensor-to-segment positions and unknown physical dimensionality of the chain and where only broad value ranges of the parameters that define the chain's geometry are known. However, we assume that sensor-to-segment orientations are known and, in the following, do not distinguish between sensor and segment frame.

As depicted in Fig. 1, only the two outer segments are equipped with IMUs, and each IMU contains gyroscope and accelerometer. Despite the sparse sensor setup, the complete pose of the kinematic chain, i.e., the relative orientation of each segment with respect to its neighboring segments, shall be determined for all time instances.

The resulting estimation problem is defined as

$$\begin{aligned} &\textbf{Estimate} \quad \boldsymbol{x}(t) = \left({}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q}^\intercal, {}^{\mathcal{S}_3(t)}_{\mathcal{S}_2(t)}\mathbf{q}^\intercal\right)^\intercal \in \mathbb{R}^8 \\ &\quad \textbf{from} \quad \boldsymbol{y}(t' < t) \quad \forall t \\ &\quad \text{where} \quad y(t) = \left(\boldsymbol{\omega}_1(t)^\intercal, \boldsymbol{\rho}_1(t)^\intercal, \boldsymbol{\omega}_3(t)^\intercal, \boldsymbol{\rho}_3(t)^\intercal\right)^\intercal \in \mathbb{R}^{12} \end{aligned} \tag{1}$$

where $\boldsymbol{\omega}_1(t)$, $\boldsymbol{\rho}_1(t)$ and $\boldsymbol{\omega}_3(t)$, $\boldsymbol{\rho}_3(t)$ denote the angular velocity and acceleration measurements at time t of the first IMU (attached to segment $\mathcal{S}_1$) and the second IMU (attached to segment $\mathcal{S}_3$), respectively. We describe orientations with unit quaternions, where, for example, ${}^{\mathcal{S}_1(t)}_{\mathcal{S}_2(t)}\mathbf{q} \in \mathbb{R}^4$ describes the orientation between segment $\mathcal{S}_1$ and segment $\mathcal{S}_2$. Finally, we assume that the kinematic chain is in motion most of the time such that there is some degree of excitation.

III. RECURRENT NEURAL NETWORKS FOR SPARSE IMT

In this section, we propose a method that solves the problem defined in (1) by training an RNNo on a virtually endless stream of just-in-time generated training data of simulated motion, where all ground truth state information are readily available. The trained RNNo then zero shot generalizes to data from an experimental realization of the sparse IMT problem (1).

RNNo is a recurrent neural-network-based observer that is obtained by stacking gated recurrent unit (GRU) and feedforward layers to create a neural network that solves (1) by mapping $y(t' < t) \rightarrow x(t)\ \forall t$. For optimization, we utilize the Adam optimizer, gradient clipping, lookahead, and truncated backpropagation through time to train the RNNo on data generated by the random chain motion generator (RCMG). The RCMG simulates random chain motion and outputs the sequences $x(t)$, $y(t)\ \forall t$, as defined in (1), where each sequence is 60s in duration. At each training step, 512 sequences are stacked to build a large batch size. Additional details regarding the RNNo and the optimization strategy can be found in [17], and in the following, we propose three augmentations to the RCMG to enrich the training data to effectively close the sim-to-real gap.

A. Range of Motion

As a first augmentation to the RCMG, we utilize the fact that segments are solid objects that cannot move through one another by enforcing that both hinge joint angles are within $[-\pi, \pi)$ at all times. We include this augmentation to encourage the RNNo to learn to predict physically feasible state trajectories. In order to generate random chain motion, the RCMG needs routines to sample both hinge joint angle trajectories $\alpha^{(l/r)}(t)$. We propose two routines for sampling in the presence of the range of motion restriction $\alpha^{(l)}(t), \alpha^{(r)}(t) \in \mathbb{R} \in [-\pi, \pi)\ \forall t$. Both routines start with the previous (or initial) hinge joint angle configuration $\alpha_{\text{prev}} = \alpha(t_{\text{prev}}) \in [-\pi, \pi)$ at timepoint $t_{\text{prev}} \geq 0$. We then sample the *next* hinge joint angle α_t at timepoint t using the following algorithm.

Random Joint Angle Trajectory Routine used by the RCMG to generate random hinge joint angle trajectories and to simulate training data for RNNo.

Require: $t_{\text{prev}} \geq 0$ {previous timepoint}
Require: $\Delta t^< \geq 0$ {lower bound for time delta between t and t_{prev}}
Require: $\Delta t^> > \Delta t^<$ {upper bound for time delta between t and t_{prev}}
Require: $\alpha_{\text{prev}} \in [-\pi, \pi)$ {previous angle value}
Require: $\mathrm{d}\alpha^< \geq 0$ {lower bound for absolute angular velocity}
Require: $\mathrm{d}\alpha^> > \mathrm{d}\alpha^<$ {upper bound for absolute angular velocity}
1: $\Delta t \leftarrow \mathcal{U}(\Delta t^<, \Delta t^>)$ {sample uniformly time delta between t and t_{prev}}
2: $\bar{p} \leftarrow 0.5(1 - \frac{\alpha_{\text{prev}}}{\pi})$ {average parity, i.e., average decision outcome whether $\alpha_t - \alpha_{\text{prev}}$ will increase or decrease}
3: $p \leftarrow 2\texttt{sampleBernoulli}(\bar{p}) - 1$ {actual decision outcome}
4: $\alpha_t^< \leftarrow \texttt{clip}(\alpha_{\text{prev}} + p\mathrm{d}\alpha^<\Delta t, -\pi, \pi)$ {lower bound for α_t}
5: $\alpha_t^> \leftarrow \texttt{clip}(\alpha_{\text{prev}} + p\mathrm{d}\alpha^>\Delta t, -\pi, \pi)$ {upper bound for α_t}
6: **if** $\alpha_t^< > \alpha_t^>$ **then**
7: $\alpha_t^<, \alpha_t^> \leftarrow \alpha_t^>, \alpha_t^<$ {swap if lower > upper}
8: **end if**
9: $\alpha_t \leftarrow \mathcal{U}(\alpha_t^<, \alpha_t^>)$
10: $t \leftarrow t_{\text{prev}} + \Delta t$

The tuple of $\alpha_{\text{prev}}, t_{\text{prev}}, \alpha_t,$ and t is then cosine interpolated and resampled to match the desired sampling rate (here: 100 Hz). We refer to the routine as written above by the name "balanced." Replacing line 2 in the above algorithm by $\bar{p} \leftarrow 0.5$ yields the second routine, which we refer to as "unbalanced." Note that "balanced" will result in a tendency for hinge joint angles to move away from π.

For both routines, the tuple $\Delta t^<, \Delta t^>, \mathrm{d}\alpha^<, \mathrm{d}\alpha^>$ can be interpreted as the parameters that define the motion distribution for both hinge joint angles. Note that these parameters have an intuitive physical meaning. As an example, the reader can consider one arm stretched out, with the palm facing upward, and then moving the palm toward the head. For the elbow joint, typical values for Δt and $\mathrm{d}\alpha$ would be $\approx$ 1s and $\approx 120\frac{\text{deg}}{\text{s}}$.

TABLE 1. Randomized Geometries of the RCMG

	${}_{\mathcal{S}_1(t)}\boldsymbol{r}_{12}$		${}_{\mathcal{S}_2(t)}\boldsymbol{d}$		${}_{\mathcal{S}_3(t)}\boldsymbol{r}_{23}$	
	Lower	Upper	Lower	Upper	Lower	Upper
$x \sim \mathcal{U}(.,.)$	0.05	0.25	0.1	0.35	0.05	0.25
$y \sim \mathcal{U}(.,.)$	−0.05	0.05	−0.02	0.02	−0.05	0.05
$z \sim \mathcal{U}(.,.)$	−0.05	0.05	−0.02	0.02	−0.05	0.05

RCMG simulates kinematic chains with various physical dimensions and sensor-to-segment positions (cf., Fig. 1) by drawing the translation vectors from the following uniform distributions (All in m).

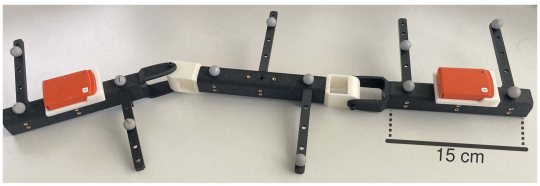


Fig. 2. 3D-printed kinematic chain used for recording experimental data. Three segments (black) connected by two hinge joints (white) with nonparallel joint axes. A sparse set of two IMUs (orange) are attached to the outer segments. Gray spheres are OMC markers.

B. Randomized Forward Kinematics

The RCMG randomizes at which node of the kinematic chain the floating base attaches before performing forward kinematics for generating training data. The global translation can attach at one of the four points in the kinematic chain, as illustrated in Fig. 1. Similarly, the global rotation can attach at one of the three segments. We add this augmentation to avoid the RNNo learning asymmetry in the IMU measurements. To motivate this, consider the following example: Naively, we may always attach the floating base to the first IMU (as depicted by $\boldsymbol{r}_1$ in Fig. 1). Then, only the specific force measurements of the second IMU will include artifacts due to centrifugal forces. RNNo might then dutifully learn this pattern but reduce the RNNos ability to generalize.

C. Randomized Geometry

The RCMG randomizes the chain geometry (physical dimensionality of the kinematic chain and sensor-to-segment positions) before simulation. The geometry that influences the measurements when it is changed is completely defined by the three 3D vectors $\boldsymbol{r}_{12}$, $\boldsymbol{d}$, and $\boldsymbol{r}_{32}$, as illustrated in Fig. 1. We randomize the geometry for every sequence generated by the RCMG. We draw each of the nine parameters according to Table 1, where $\mathcal{U}$(lower, upper) denotes a, on both limits inclusive, uniform distribution. Comparing Table 1 with Fig. 2, we can see that for, e.g., ${}_{\mathcal{S}_2(t)}\boldsymbol{d}$ (vector connecting both hinge joints), the longitudinal x-component is randomized across a generous value range from 0.1 to 0.35m, which includes the true experimental length of $\approx$ 0.2m. This augmentation encourages the RNNo to implicitly estimate the chain geometry for broad applicability, even under large geometric uncertainties in practice.

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate the proposed method (cf. Section III) on a suitable experimental realization of the sparse IMT problem (cf. Section II). In the following, we will first describe the experimental setup, and then validate the RNNos long-term stability and initial convergence. Finally, we will present an exhaustive ablation study that presents the contribution of each proposed augmentation individually.

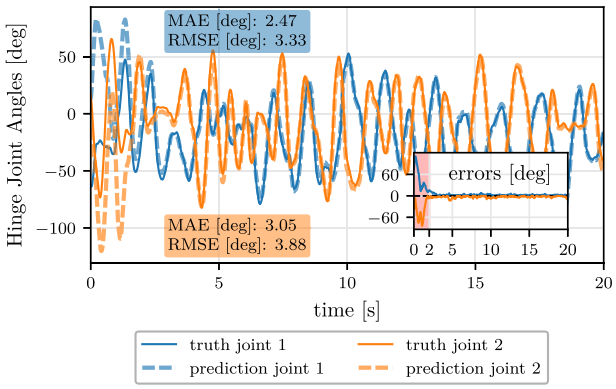


Fig. 3. Estimation performance of the trained RNNo on one exemplary sequence. After a short initial convergence, the RNNo closely tracks both hinge joint angles. Mean absolute error (MAE) and root-mean-square error (RMSE) are calculated after convergence (2s) for each sequence.

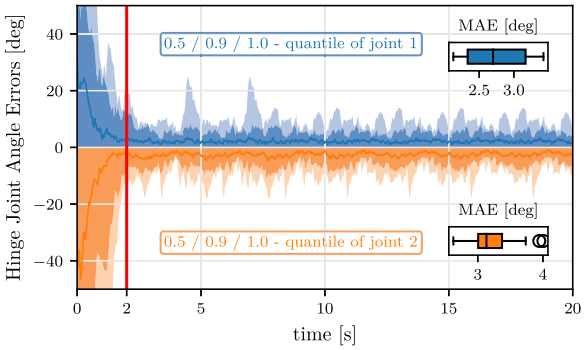


Fig. 4. Initial convergence performance of the trained RNNo across all 15 sequences. RNNo consistently convergences within < 2s after convergence errors remain low at all times.

A. Experimental Setup

To evaluate the RNNos performance, we use an experimental setup that is compliant with the sparse IMT problem considered here (cf. Section II). To this end, we have performed a measurement session of $\approx$ 120s to assess both long-term stability, as well as initial convergence. The 3D-printed kinematic chain has been moved randomly and is in motion most of the time. Fig. 2 showcases the 3D-printed kinematic chain that was used to record the experimental data, i.e., the magnetometer-free IMU measurements and ground truth pose measurements, using an optical motion capture (OMC) system. As preprocessing, we have downsampled the OMC measurements from 120 to 100Hz using spherical linear interpolation, and time synchronization with IMU measurements (that already record natively at 100Hz) is achieved by cross correlation of the norms of the measured (IMU) and calculated angular velocities (OMC). Reference orientations are obtained by spanning an orthogonal coordinate system using three markers per segment.

B. Initial Convergence

To assess the RNNos ability to quickly converge to a correct estimate, we have extracted 15 sequences of 20s duration. In these sequences, the initial orientation of the kinematic chain is unknown. Fig. 3 shows one exemplary sequence, and Fig. 4 showcases the performance across all sequences. The RNNo consistently converges to

TABLE 2. Effect of Each Augmentation on the Performance of the Trained RNNo (RoM: Range of Motion, RandForKin: Randomized Forward Kinematics, and RandGeo: Randomized Geometry)

Augmentation 1 RoM	Augmentation 2 RandForKin	Augmentation 3 RandGeo	MAE [deg] (in experiment)
✗	✗	✗	73.6 ± 27.0
✗	✗	✓	40.3 ± 12.0
✗	✓	✗	32.3 ± 5.2
✗	✓	✓	15.9 ± 1.7
✓(unbalanced)	✗	✗	38.8 ± 16.0
✓(unbalanced)	✗	✓	19.6 ± 13.0
✓(unbalanced)	✓	✗	19.2 ± 11.0
✓(unbalanced)	✓	✓	10.1 ± 5.0
✓(balanced)	✗	✗	10.8 ± 2.5
✓(balanced)	✗	✓	5.04 ± 0.80
✓(balanced)	✓	✗	8.54 ± 1.50
✓(balanced)	✓	✓	**3.25 ± 0.29**

Standard deviations utilize five seeds for the RNNos parameter initialization.

hinge joint angle estimates with low estimation errors within < 2s, and errors remain low after convergence at all times. This showcases the RNNos applicability despite unknown initial orientations, unknown sensor-to-segment positions, and unknown physical chain dimensions, and it highlights the RNNos plug-and-play capabilities.

C. Long-Term Stability

To assess long-term stability, we apply the RNNo to a 70s continuous experimental estimation task. Excluding 2s for initial convergence, the RNNo was able to reliably track both hinge joint angles with the following errors: MAE 2.74 / 3.15, RMSE 3.95 / 4.26, and 95%-percentile 7.96 / 9.07 degrees (joint 1/joint 2). This shows that the trained RNNo can successfully track both hinge joint angles even on longer sequences.

D. Ablation Study

Finally, we present the contribution of each proposed augmentation individually (cf., Section III) on the performance of the trained RNNo. Table 2 lists the MAE in degrees across both hinge joints (as depicted on the right side of Fig. 4) for all combinations of disabled augmentations. With all three augmentations disabled, the RNNo does not generalize, resulting in an experimental MAE > 70 degrees. Only if all three augmentations are enabled can the RNNo achieve an experimental MAE < 4 degrees, despite the magnetometer-free and sparse sensor setup, and which is in line with errors typically only achieved with state-of-the-art sensor fusion methods for full IMU setups (one sensor per segment) [19].

V. CONCLUSION

In this contribution, we have successfully shown that a recurrent neural network that is trained in simulation can generalize in practice to enable sparse and magnetometer-free motion tracking of experimental and unseen kinematic chains. This plug-and-play IMT capability is achieved by domain-specific augmentations that are applied during the training of an RNNo to effectively close the sim-to-real gap. After rapid initial convergence, the trained RNNo can achieve long-term stable tracking errors of < 4 degrees. The results indicate that RNNos can be used for plug-and-play IMT in magnetically disturbed environments with minimal efforts, and thus achieve unprecedented high applicability and usability, at unprecedented low cost.

The presented approach constitutes a proof-of-concept of how complex sensing problems can effectively be solved by first proving observability in simulation and utilizing a neural-network-based observer, and then realizing generalization in practical settings by suitable augmentations to the simulation. Interesting directions for future work are: conducting an extensive experimental validation, including a variety of motion aspects; extending the RNNo to joint setups that include a higher degree of freedom joints; and enabling RNNo to generalize across joint axes and sensor-to-segment orientations.

REFERENCES

[1] V. R. Marco, J. Kalkkuhl, and T. Seel, "Nonlinear observer with observability-based parameter adaptation for vehicle motion estimation," *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 60–65, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318317518

[2] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 340–345.

[3] M. W. Givens and C. Coopmans, "A survey of inertial sensor fusion: Applications in sUAS navigation and data collection," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2019, pp. 1054–1060.

[4] I. H. López-Nava and A. Muñoz-Meléndez, "Wearable inertial sensors for human motion analysis: A review," *IEEE Sensors J.*, vol. 16, no. 22, pp. 7821–7834, Nov. 2016.

[5] P. Picerno, "25 years of lower limb joint kinematics by using inertial and magnetic sensors: A review of methodological approaches," *Gait Posture*, vol. 51, pp. 239–246, 2017.

[6] D. Y. Ku and A. Patel, "Kinematic state estimation using multiple DGPS/MEMS-IMU sensors," *IEEE Sensors Lett.*, vol. 4, no. 12, Dec. 2020, Art. no. 5501604.

[7] M. H. Rahmani, R. Berkvens, and M. Weyn, "Chest-worn inertial sensors: A survey of applications and methods," *Sensors*, vol. 21, no. 8, 2021, Art. no. 2875. [Online]. Available: https://www.mdpi.com/1424-8220/21/8/2875

[8] W. De Vries, H. Veeger, C. Baten, and F. van der Helm, "Magnetic distortion in motion labs, implications for validating inertial magnetic sensors," *Gait Posture*, vol. 29, no. 4, pp. 535–541, 2009.

[9] B. Taetz, G. Bleser, and M. Miezal, "Towards self-calibrating inertial body motion capture," in *Proc. IEEE 19th Int. Conf. Inf. Fusion*, 2016, pp. 1751–1759. [Online]. Available: http://arxiv.org/abs/1606.03754

[10] D. Laidig, I. Weygers, and T. Seel, "Self-calibrating magnetometer-free inertial motion tracking of 2-DoF joints," *Sensors*, vol. 22, no. 24, 2022, Art. no. 9850.

[11] T. Von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll, "Sparse inertial poser: Automatic 3D human pose estimation from sparse IMUs," in *Computer Graphics Forum*, vol. 36, no. 2. Hoboken, NJ, USA: Wiley, 2017, pp. 349–360.

[12] L. Sy et al., "Estimating lower limb kinematics using a reduced wearable sensor count," *IEEE Trans. Biomed. Eng.*, vol. 68, no. 4, pp. 1293–1304, Apr. 2021.

[13] M. Nazarahari and H. Rouhani, "Semi-automatic sensor-to-body calibration of inertial sensors on lower limb using gait recording," *IEEE Sensors J.*, vol. 19, no. 24, pp. 12465–12474, Dec. 2019.

[14] D. Graurock, T. Schauer, and T. Seel, "Automatic pairing of inertial sensors to lower limb segments—A plug-and-play approach," *Curr. Directions Biomed. Eng.*, vol. 2, no. 1, pp. 715–718, 2016.

[15] A. Grapentin, D. Lehmann, A. Zhupa, and T. Seel, "Sparse magnetometer-free real-time inertial hand motion tracking," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst.*, 2020, pp. 94–100.

[16] X. Yi et al., "Physical inertial poser (pip): Physics-aware real-time human motion tracking from sparse inertial sensors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 13167–13178.

[17] S. Bachhuber, D. Weber, I. Weygers, and T. Seel, "RNN-based observability analysis for magnetometer-free sparse inertial motion tracking," in *Proc. 25th Int. Conf. Inf. Fusion*, 2022, pp. 1–8.

[18] K. Eckhoff, M. Kok, S. Lucia, and T. Seel, "Sparse magnetometer-free inertial motion tracking a condition for observability in double hinge joint systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 16023–16030, Jan. 2020.

[19] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, and D. Stricker, "Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion," *Sensors*, vol. 17, no. 6, 2017, Art. no. 1257. [Online]. Available: https://www.mdpi.com/1424-8220/17/6/1257

Paper C

Recurrent Inertial Graph-Based Estimator (RING): A Single Pluripotent Inertial Motion Tracking Solution

Authors: Bachhuber, Simon and Weygers, Ive and Lehmann, Dustin and Dombrowski, Mischa and Seel, Thomas

Published in: Transactions on Machine Learning Research

Publication date: October 2024

URL: `https://openreview.net/forum?id=h2C3rkn0zR`

Recurrent Inertial Graph-Based Estimator (RING): A Single Pluripotent Inertial Motion Tracking Solution

Simon Bachhuber *simon.bachhuber@fau.de*
Department of Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

Ive Weygers *ive.weygers@fau.de*
Department of Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

Dustin Lehmann *dustin.lehmann@tu-berlin.de*
Control Systems Group
Technical University Berlin

Mischa Dombrowski *mischa.dombrowski@fau.de*
Department of Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

Thomas Seel *thomas.seel@imes.uni-hannover.de*
Institute of Mechatronics Systems
Leibniz Universität Hannover

Reviewed on OpenReview: *https://openreview.net/forum?id=h2C3rkn0zR*

Abstract

This paper introduces a novel ML-based method for Inertial Motion Tracking (IMT) that fundamentally changes the way this technology is used. The proposed method, named RING[1] (Recurrent Inertial Graph-Based Estimator), provides a pluripotent, problem-unspecific plug-and-play IMT solution that, in contrast to conventional IMT solutions, eliminates the need for expert knowledge to identify, select, and parameterize the appropriate method. RING's pluripotency is enabled by a novel online-capable neural network architecture that uses a decentralized network of message-passing, parameter-sharing recurrent neural networks, which map local IMU measurements and nearest-neighbour messages to local orientations. This architecture enables RING to address a broad range of IMT problems that vary greatly in aspects such as the number of attached sensors, or the number of segments in the kinematic chain, and even generalize to previously unsolved IMT problems, including the challenging combination of magnetometer-free and sparse sensing with unknown sensor-to-segment parameters. Remarkably, RING is trained solely on simulated data, yet evaluated on experimental data, which indicates its exceptional ability to zero-shot generalize from simulation to experiment, while outperforming several state-of-the-art problem-specific solutions. For example, RING can, for the first time, accurately track a four-segment kinematic chain (which requires estimating four orientations) using only two magnetometer-free inertial measurement units. This research not only makes IMT more powerful and less restrictive in established domains ranging from biomechanics to autonomous systems, but also opens its application to new users and fields previously untapped by motion tracking technology. Code and data is available here.

[1] one to track them all

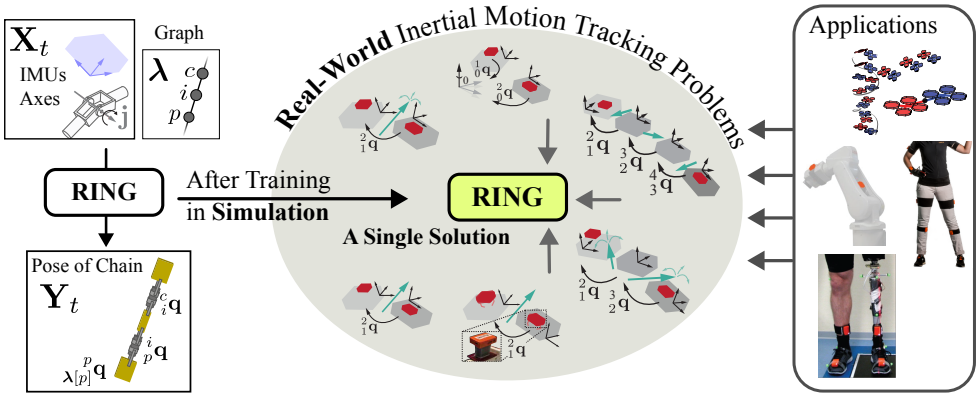


Figure 1: RING is a ML-based method that provides a versatile, pluripotent IMT solution applicable across a broad range of challenging IMT problems, designed for use without the need for expert knowledge. Remarkably, RING is trained solely on simulated data, yet zero-shot generalizes to real-world experiments and outperforms several problem-specific state-of-the-art solutions.

1 Introduction

In the domain of multi-agent systems, structural policies have shown great potential for the control of complex agents; meanwhile, Recurrent Neural Networks (RNNs) are an established choice for sequential data. The potential of their combination for the analysis of structural sequential data has rarely been investigated and exploited. This combination seems particularly promising for state estimation in graph-structured systems, such as, for example, for IMT of Kinematic Chains (KCs).

The need for reliable and accurate estimation of the orientation, attitude, or pose of articulated objects in three-dimensional (3D) space spans across various application domains ranging from aerospace engineering (Euston et al., 2008; Givens & Coopmans, 2019) to health applications (Buke et al., 2015; López-Nava & Muñoz-Meléndez, 2016; Seel et al., 2020). Inertial Measurement Units (IMUs), which typically comprise a 3D accelerometer, a 3D gyroscope, and a 3D magnetometer, have become smaller and less expensive within the last two decades and have therefore rapidly become the most promising technology for accurate, reliable, and inexpensive motion tracking in rigid bodies and KCs, especially since camera-based systems are typically more expensive, more restrictive, and suffer from occlusion (von Marcard et al., 2017; Huang et al., 2018).

However, fusing the available measurement signals to estimate the desired orientations requires advanced IMT algorithms that typically need to overcome a combination or all of the following three main IMT challenges (Seel et al., 2020): (1) inhomogeneous magnetic fields in indoor environments and in proximity of ferromagnetic material or electric devices; (2) sensor-to-segment calibration, i.e. identifying the joint position and axis orientations in local sensor coordinates; (3) solving sparse problems in which some segments of the KC are not equipped with a sensor, to improve the usability and reduce costs.

In recent years, numerous highly specialized methods have been proposed to address these challenges. Magnetometer-free methods have been developed to estimate the relative orientation between two adjacent segments by exploiting different kinematic constraints (Kok et al., 2014; Laidig et al., 2017; Lehmann et al., 2020; 2024). Moreover, numerous general-purpose magnetometer-free attitude estimators have been proposed (Mahony et al., 2008; Madgwick, 2010; Seel & Ruppin, 2017; Weber et al., 2021; Laidig & Seel, 2023). Several algorithms were developed to achieve sensor-to-segment calibration for specific kinematics with full sensor setups (Taetz et al., 2016; McGrath et al., 2018; Olsson et al., 2020). Finally, a variety of sparse IMT methods have been developed that either use a limited number of sensors while still depending on magnetometers (von Marcard et al., 2017; Huang et al., 2018; Sy et al., 2020; 2021; Zheng et al., 2021) or are magnetometer-free (Grapentin et al., 2020; Yi et al., 2021; 2022; Bachhuber et al., 2023; Van Wouwe et al., 2023).

In summary, there exists a plethora of methods, each tailored to a very specific application, such as magnetometer-free tracking of a single-degree-of-freedom (1-DoF) joint (Lehmann et al., 2020), or human

pose estimation from six IMUs, as detailed in Yi et al. (2021). To apply IMT, the user must successfully identify the method that is suitable for the given problem and typically specify various parameters such as joint axes directions and sensor placement. Therefore, the user must be an expert in the field of IMT, which strongly limits the use of IMUs in many application domains. To make matters even worse, a given problem might require a nontrivial combination of methods which may exclude each other, e.g., the tracking of a sparse three-segment KC currently requires known joint axes directions (Bachhuber et al., 2023), but the method that estimates the joint axes directions does not allow for a sparse sensor setup (Olsson et al., 2020).

What if, instead of a plethora of methods, we had a single *pluripotent* method that can be used for, e.g., magnetometer-free tracking of 1-DoF joints, and the tracking of sparse sensor setups with known or even unknown joint axes directions? What if we had *one to track them all*?

In this work, we demonstrate that ML methods can be used to, for the first time, achieve this goal. We propose a method, named RING, that combines a novel neural network architecture, named RINGCell, with an elaborate training data simulation, the Random Chain Motion Generator (RCMG). The key insight that enables the pluripotency of RING is that a given system in an Inertial Motion Tracking Problem (IMTP) can be viewed as a graph where nodes represent segments and edges represent a single DoF. Then, a shared set of parameters can be applied on a decentralized, per-node level, where only local IMU measurements are observed and only the local estimation problem is solved, i.e., the orientation relative to the parent is estimated. Information exchange between nodes is enabled by passing messages along the edges of the graph.

We show that RING can be used to plug-and-play solve a range of challenging magnetometer-free sparse or non-sparse motion tracking problems with a single trained neural network and that it even solves previously unsolved challenging IMTPs, such as, e.g., the tracking of a triple-hinge-joint system with only two IMUs. We demonstrate that although RING is trained solely on simulated data, it zero-shot generalizes to experimental data and aligns with various state-of-the-art (SOTA) results.

2 Related Works

As mentioned above, there exists a plethora of highly specialized methods in the field of IMT, but there is no single pluripotent solution that solves a variety of IMTPs. Even the use of ML methods for IMT has so far only led to specific solutions for single IMTPs. RNNs have been used in Weber et al. (2021) to achieve SOTA attitude estimation, and in Bachhuber et al. (2023) to successfully track a specific sparse KC. Deep learning has also been used for human motion capture, where the full-body pose is estimated from typically six or more IMUs, and previous work has shown promising results (von Marcard et al., 2017; Huang et al., 2018; Zheng et al., 2021; Yi et al., 2021; 2022; Van Wouwe et al., 2023; Puchert & Ropinski, 2023). However, while addressing a challenging problem, these methods are limited to human motion capture with one specific sensor setup and assume statistical patterns of human motion (von Marcard et al., 2017), or full-body biomechanical models (Yi et al., 2022) to constrain the estimated pose.

From a methodological viewpoint, RING uses a decentralized network of message-passing RNNs with shared parameters that are trained via supervised learning. The concept of decentralized networks, communication, and collaboration is at the heart of multi-agent systems, and the means of communication can either be prescribed (Panait & Luke, 2005; Wang et al., 2019) or, more recently, learned (Sukhbaatar et al., 2016; Foerster et al., 2016; Wang et al., 2018; Pathak et al., 2019; Huang et al., 2020). In deep Reinforcement Learning (RL), feedforward networks have been used to parameterize structured policies that pass messages along the edges of a graph in Sukhbaatar et al. (2016); Foerster et al. (2016); Wang et al. (2018); Huang et al. (2020), and distinct advantages of message-passing have been shown. In particular, in Huang et al. (2020) it was investigated whether centralised control can emerge from decentralized policies, and they show that it is possible to learn a global policy that achieves locomotion across various agent morphologies. It is interesting to note that policies must collaborate in order to achieve a global task, e.g., locomotion, and are motivated by a global reward. In the present work, the decentralized RNNs must collaborate by exchanging information in order to solve the task of motion tracking, and are motivated by a decentralized loss function. The advantages of communication and global coordination emerging in a decentralized structure were also investigated in Sukhbaatar et al. (2016); Foerster et al. (2016). In particular, the work of Sukhbaatar et al. (2016) uses supervised learning instead of RL for learning communication protocols.

3 Preliminaries

3.1 Notation

We use a typical notation with scalars denoted by x, vectors by $\boldsymbol{x}$, matrices (or higher dimensional tensors) by $\mathbf{X}$, and quaternions (or higher dimensional arrays of quaternions) by $\mathbf{q}$. Additionally, note that the symbol $\mathbb{1}$ defines either the unity element of a given space, or the indicator function, such that, e.g., $\mathbb{1}_0(i)$ is one for $i = 0$ and zero else. The symbol $\otimes$ is used to denote the direct product of vector spaces, and additionally denotes quaternion multiplication, see Definition B.2. Further details can be found in Appendix A.1.

3.2 An Inertial Motion Tracking Problem

We define an IMTP as the task of estimating the trajectory of the complete rotational state of an Articulated Rigid-Body System (ARBS) from inertial data. Conceptually, an ARBS is a collection of multiple rigid (or assumed to be rigid) bodies that are interconnected via joints that allow for relative motion between these bodies. Let there be a singleton, inertial reference coordinate system named *base* (sometimes *world*, or *Earth*), then an orientation of a body relative to the base is referred to as an absolute orientation. An orientation of a body relative to the coordinate system of another body is referred to as a relative orientation. Then, assuming that no additional information about the types of joints is provided, to fully describe the rotational state of an ARBS that consists of N bodies at a single moment in time, N orientations must be specified. In this work, we will utilize $N - 1$ relative orientations and one absolute orientation.

An IMTP is solved by estimating the rotational state from at most N IMUs that are connected to the bodies of the ARBS (at most one IMU per body), and that provide 3D measurements of angular rates, specific forces, and the magnetic field density in their local sensor coordinates. A magnetometer-free method solves an IMTP without the use of the magnetometer, and such IMUs are referred to as 6D IMUs. In this case, the rotational state of the ARBS can only be estimated up to an absolute heading error, that is, up to an arbitrary rotation around the gravity (or vertical) direction, and the one absolute orientation is referred to as the absolute attitude. This is because both accelerometer and gyroscope measurements are invariant under a rotation around the vertical direction. Finally, note that if at least one body of the ARBS does not have an IMU attached, then the IMTP is said to be sparse.

3.3 Graph Connectivity

The topology of an ARBS is represented by a Connectivity Graph (CG) (Featherstone, 2008; 2010), which is an undirected graph where the nodes represent the bodies that constitute the ARBS and the edges represent its joints.

Before the CG can be encoded, the bodies must be numbered. We adopt the broadly adopted standard numbering scheme and notation from Featherstone (2008; 2010) which for an ARBS with N bodies proceeds as follows:

1. The base is assigned the number 0 and it serves as the root node.
2. The remaining bodies are consecutively numbered from 1 to N, so that each body has a higher number than its parent.

The CG may then be encoded via a parent array $\boldsymbol{\lambda} \in \mathbb{N}^N$ where $\boldsymbol{\lambda}[i]$ is the body number of the parent of body i. Additionally, we define the function $\mu(i)$ to return the set of the body numbers of the children of body i, that is

$$\mu_\lambda(i) = \{j \mid \boldsymbol{\lambda}[j] = i \quad \forall j = 1 \dots N\}\,. \tag{1}$$

Definition 3.1. The body i of an ARBS with parent array $\boldsymbol{\lambda}$ is said to be an outer body if it has no children bodies, i.e., $\mu_\lambda(i) = \{\}$, or if its parent is the base, i.e., $\boldsymbol{\lambda}[i] = 0$; otherwise it is said to be an inner body.

As an example, consider an ARBS that is a three-segment KC (see Figure 2). There, the three bodies are numbered increasingly from top-to-bottom, and then, for this numbering, the parent is given by $\boldsymbol{\lambda} = (0, 1, 2)^\intercal$.

Note that if the inner body (middle segment) is assumed to connect to the base, then the parent array is different and given by $\boldsymbol{\lambda} = (0, 1, 1)^\intercal$.

3.4 Minimal and Maximal Coordinates

We refer to the generalized coordinates position vector as the minimal coordinates, denoted by $\boldsymbol{q}$. It fully captures the kinematic state of the ARBS using a minimal amount of coordinates. The size of $\boldsymbol{q}$ depends on the DoFs of the joints in the ARBS. In this work, we will consider ARBSs that can move freely in space (without any constraints). Therefore, in the corresponding CG, the edges that connect to the base are 6-DoF free joints and all the remaining edges represent single DoF.

Definition 3.2. For an ARBS with N bodies with a CG given by $\boldsymbol{\lambda}$ where the joints that connect to the base are 6-DoF and 1-DoF otherwise, the size of $\boldsymbol{q}$ is given by $N_q = \sum_i^N 7\mathbb{1}_0(\boldsymbol{\lambda}[i]) + (1 - \mathbb{1}_0(\boldsymbol{\lambda}[i]))$. Similarly, the size of the velocity of minimal coordinates $\dot{\boldsymbol{q}}$ is given by $N_{\dot{q}} = \sum_i^N 6\mathbb{1}_0(\boldsymbol{\lambda}[i]) + (1 - \mathbb{1}_0(\boldsymbol{\lambda}[i]))$.

For an ARBS with N bodies, we refer to the set of all euclidean positions and rotational states from the base to all bodies in the system as the maximal coordinates, denoted by $\boldsymbol{t} \in \left(\mathbb{H} \otimes \mathbb{R}^3\right)^N$. The size of $\boldsymbol{t}$ depends only on the number of bodies in the system.

3.5 Representing Orientations

We represent 3D orientations with quaternions $\mathbf{q}$ due to various advantages over equivalent representations using orthogonal matrices or Euler angles (Kuipers, 2002). In particular, we use ${}^0_1\mathbf{q} \in \mathbb{H}$ to denote the absolute orientation from the base to the body one's coordinate system. Similarly, we use ${}^1_2\mathbf{q}$ to denote the relative orientation from body one to body two. In Appendix B, we define various utilized quaternion-related operations.

3.6 Loss Function For Orientations

In order to train and evaluate ML methods that predict orientations (represented by quaternions), a suitable metric function must be identified. We can compare the difference between a ground truth orientation $\mathbf{q}$ and the corresponding predicted orientation $\hat{\mathbf{q}}$ by computing the angle of the smallest rotation that makes ground truth and prediction identical. It is given by $\texttt{angle}(\mathbf{q} \otimes \hat{\mathbf{q}}^*)$ where $\otimes$ denotes quaternion multiplication, * denotes the complex conjugate, and `angle` extracts the rotation angle of a quaternion (see Appendix B.5). Then, we use the following mean-squared-error function $\texttt{loss} : \mathbf{q} \in \mathbb{H}, \hat{\mathbf{q}} \in \mathbb{H} \to \mathbb{R}_{\geq 0}$ to calculate a scalar error between two quaternions, given by

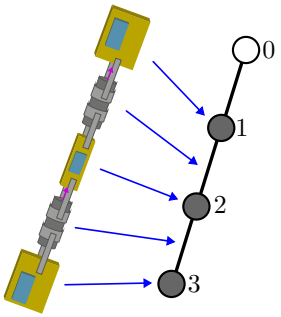
$$\texttt{loss}(\mathbf{q}, \hat{\mathbf{q}}) = \texttt{angle}(\mathbf{q} \otimes \hat{\mathbf{q}}^*)^2. \tag{2}$$

The loss function for a single orientation in eq. (2) is then used to compute the mean-squared-error for the entire rotational state of the KC in eq. (10).

4 Method

In this section, we define the problem under consideration and the proposed method which consists of three components:

- A virtually infinite, simulated training data set (see Section 4.2).
- A novel, online-capable neural network architecture, named RINGCell, purpose-built for state estimation in ARBS (see Section 4.3). In contrast to typical RNNs that map a fixed number of input features to a fixed number of output features using a centralized logic, RINGCell leverages the graph-structure of ARBS and employs a decentralized, message-passing logic with a shared set of parameters. This design maps a fixed number of input features *per body* to a fixed number of output features *per body*, and it allows RINGCell to adapt to the size and topology of the ARBS without the need for retraining.



$$
\begin{aligned}
\texttt{sys} = \{ & \\
& \boldsymbol{n}, \boldsymbol{\lambda} \in \mathbb{N}^N, \\
& \mathbf{J} \in \mathbb{R}^{N\times 3}, \\
& \mathbf{R} \in \mathbb{R}^{2N\times 3}, \\
& \mathbf{K}, \boldsymbol{\Gamma} \in \mathbb{R}^{N\times 6}, \\
& T_s \in \mathbb{R}, \\
\} &
\end{aligned}
$$

Figure 2: System object example (see Definition 4.2) for a three-segment KC with $N = 3$ bodies and parent array $\boldsymbol{\lambda}_3 = (0, 1, 2)^\intercal$.

- A powerful IMT solution, named RING, that can solve a broad range of IMTPs in the real world (see Section 4.4). RING is obtained by training RINGCell on the simulated training data.

Finally, Section 4.5 provides a software implementation of the entire method and, more specifically, RING. Most notably, we provide a single file that trains the here benchmarked version of RING from scratch *without requiring any files that contain real-world training data.*

4.1 Problem Formulation: A Class of Inertial Motion Tracking Problems

In this work, we consider the following class of IMTPs. We consider an N-segment KC where $N \in \{1, 2, 3, 4\}$ with unknown physical geometry and with segments that are interconnected via hinge joints. The axes directions of these hinge joints may be *known or unknown*. For each inner body *at most one* and for each outer body exactly one 6D IMU, with unknown sensor-to-body position, is *rigidly or nonrigidly* attached to each body. For each body and with the KC at rest, the constant sensor-to-body orientation is assumed to be known. The initial pose of the ARBS is assumed to be unknown. Then, the task is to estimate, for every timestep, the rotational state of the KC (consisting of N orientations) from the available IMU measurements and the available joint axes directions. Note that the task requires providing accurate estimates at each timestep, thus necessitating an online-capable solution focused on online, real-time processing (filtering) as opposed to retrospective data processing (smoothing). To summarize, two IMTPs of this class of IMTPs can differ in: 1) the number of bodies N in the KC, 2) the number of attached IMUs, 3) the number of known joint axes directions, and 4) whether IMUs are attached rigidly or nonrigidly. A broad range of IMTPs from this class of IMTPs are illustrated in Figure 1 within the grey ellipsoid.

Definition 4.1. The parent arrays $\boldsymbol{\lambda}_N$ of the N-segment KC where $N \in \{1, 2, 3, 4\}$ are given by, without loss of generality, $\boldsymbol{\lambda}_1 = (0)^\intercal, \boldsymbol{\lambda}_2 = (0, 1)^\intercal, \boldsymbol{\lambda}_3 = (0, 1, 2)^\intercal, \boldsymbol{\lambda}_4 = (0, 1, 2, 3)^\intercal$, respectively.

4.2 Training Data: The RCMG Algorithm

The RING network is trained on data obtained from simulated random motion of one-, two-, three-, and four-segment KCs. The procedure that generates this training data (from only PseudoRNG) is called the Random Chain Motion Generator (RCMG) (Bachhuber et al., 2022). The RCMG procedure (see Algorithm 1) has three main steps that execute consecutively:

1. Randomized KC (see Section 4.2.1). A randomized KC is drawn to manipulate the downstream simulation and achieve various forms of domain randomization that enable to bridge the sim-to-real gap from simulation (training) to real-world (testing).
2. Random Motion Generation (see Section 4.2.2). The KC is simulated to randomly move in space and the maximal coordinates of all bodies relative to the base are computed for every timestep.
3. Get $\mathbf{X}, \mathbf{Y}$ Data (see Section 4.2.3). From the maximal coordinates of all bodies, the IMU, joint axes, and pose data is computed, and returned as training data.

Internally, the RCMG procedure uses a system object `sys` (see Definition 4.2) which is the collection of various attributes such as, e.g., a joint axes array $\mathbf{J} \in \mathbb{R}^{N\times 3}$.

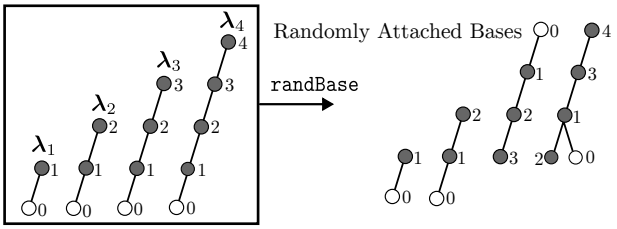


Figure 3: Each sequence of training data is simulated using the RCMG. In the first step of the RCMG a randomized KC is created which involves randomizing the node in the KC to which the base attaches. Afterwards, the nodes in the graph are numbered according to Section 3.3. For example here, for $\boldsymbol{\lambda}_4$, the new numbering array is $\boldsymbol{n} = (2, 1, 3, 4)^\intercal$, and the new parent array $\boldsymbol{\lambda} = (0, 1, 1, 3)^\intercal$.

Definition 4.2. We define a system object `sys` with N bodies as the collection of the following attributes:

- a numbering array of integers $\boldsymbol{n} \in \mathbb{N}^N$ which stores a permutation of the numbers going from $1 \dots N$,
- a parent array of integers $\boldsymbol{\lambda} \in \mathbb{N}^N$,
- a joint axes array $\mathbf{J} \in \mathbb{R}^{N\times 3}$ that contains the hinge joint axis direction,
- a relative-to-parent position array $\mathbf{R} \in \mathbb{R}^{2N\times 3}$ that contains the position vector of the body's coordinate system relative to its parent (expressed in the parent's coordinate system). In Section 4.2.1, it is outlined that for each of the N bodies there is a second IMU body. In the $\mathbf{R}$ array, the first N values are used to specify the position vector between two non-IMU bodies (segment-to-segment positions, physical geometry of the KC), while the last N values are used to specify the position vector from non-IMU body to the corresponding IMU body (segment-to-body positions, IMU attachment). This is done to independently randomize both the physical geometry and the IMU attachment, forcing the network to learn to generalize to scenarios such as an arm robot with unknown segment lengths, as well as to calibrate for an unknown IMU attachment,
- an array of stiffness parameters for N free joints $\mathbf{K} \in \mathbb{R}^{N\times 6}$,
- an array of damping parameters for N free joints $\mathbf{\Gamma} \in \mathbb{R}^{N\times 6}$,
- a float representing the sampling time $T_s \in \mathbb{R}$, here always 0.01 s.

An exemplary system object is shown in Figure 2.

4.2.1 Step 1) Randomized Kinematic Chain

In order to enrich the simulated training data, several forms of domain randomization are achieved by drawing a KC with randomized system attributes.

The first domain randomization is the randomization of all downstream forward kinematics applications, and additionally, randomization of the absolute random translation and orientation in the generation of random motion. This has been shown to improve the training data such that the trained network more effectively closes the sim-to-real gap (Bachhuber et al., 2023) This is achieved by re-attaching the bases randomly and afterwards, the nodes in the graph are re-numbered according to Section 3.3. An example of this is shown in Figure 3. Secondly, the N randomized hinge joint axes $\mathbf{J}$ are drawn. Thirdly, the position array $\mathbf{R}$ is randomized by drawing values from uniform ranges.

Finally, the stiffness $\mathbf{K}$ and damping $\mathbf{\Gamma}$ arrays are randomized. These values are used to model nonrigidly attached IMUs by connecting additional nodes that are connected via passive spring-damper free joints. For each node i in the CG, we add an additional IMU node with body number $i + N$ that is a child of node i and that is connected to node i via a passive spring-damper free joint. The stiffness $\mathbf{K}[i]$ and damping $\mathbf{\Gamma}[i]$

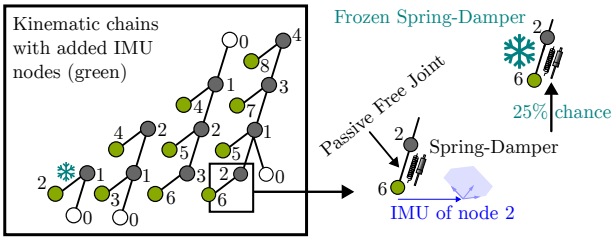


Figure 4: For each node i in the KCs, there exists a second IMU node (that is not counted in the body count N of the system; here in green) with body number $i + N$. The IMU node is connected via a passive spring-damper free joint to the original node in order to simulate nonrigidly-attached IMUs. There is a 25% chance that the damping and stiffness parameters of the passive free joint are chosen such that the IMU is effectively rigidly attached and the passive free joint is frozen.

parameters are used during the forward simulation of the KC as the parameters of the 3D spring-damper dynamics between node i and (IMU) node $i + N$. For each node i, the stiffness $\mathbf{K}[i]$ and damping $\mathbf{\Gamma}[i]$ parameters are randomized in a way such that, either the IMU is effectively rigidly attached, or such that the IMU moves relative to the segment node i (which then models nonrigid attachment).

4.2.2 Step 2) Random Motion Generation

As a second step, the RCMG procedure generates random motion of the previously obtained randomized KC. Random motion is obtained by drawing a random reference trajectory for all joints in the KC. The generation of such randomized reference trajectories is influenced and constrained by various parameters, e.g., upper limits on angular velocities or lower limits on the amount of motion (to avoid jittering). Additional details on the reference trajectory generation can be found in Appendix C.1. Afterwards, a dynamical forward simulation is performed where joint forces are computed using PD control such that the random reference is tracked. Note that the additionally added nodes to model nonrigid IMU attachment are passive free joints and as such they are not actuated. Finally, the trajectories of maximal coordinates of all N bodies and N IMU bodies, given by $\mathbf{T} \in \left(\mathbb{H} \otimes \mathbb{R}^3\right)^{2N \times T}$ (from base to body), are computed.

4.2.3 Step 3) Get $\mathbf{X}, \mathbf{Y}$ Data

In the last step, the training tuples of $\mathbf{X}, \mathbf{Y}$ are computed from the trajectories of maximal coordinates and afterwards returned. They are:

- $\mathbf{X} \in \mathbb{R}^{T \times N \times 9}$, where $\mathbf{X}[:, i, :6]$ is the simulated 6D IMU data for body i as measured in its IMU body $i + N$ (but for each inner body, there is a $\frac{2}{3}$ chance that the IMU data gets dropped out and replaced by zeros), and where $\mathbf{X}[:, i, 6:]$ is the joint axis direction $\mathbf{J}[i]$ of the hinge joint between body i and its parent $\boldsymbol{\lambda}[i]$ and zeros if the parent is the base (but for each body, there is a $\frac{1}{2}$ chance that the joint axis direction data gets dropped out and replaced by zeros), and

- $\mathbf{Y} \in \mathbb{H}^{T \times N}$ where $\mathbf{Y}[:, i]$ is the timeseries of: 1) absolute attitudes ${}^{i}_{0}\mathbf{q}(t)$ if the parent $\boldsymbol{\lambda}[i]$ is the base; 2) relative orientations ${}^{i}_{p}\mathbf{q}(t)$ from body i to its parent $p = \boldsymbol{\lambda}[i]$, and

where T is the number of timesteps (here, 60 s at 100 Hz, thus $T = 6000$), and N is the number of bodies in the KC. IMU and joint axes data is dropped out in order to force RING to learn to solve IMTPs with sparse IMU placement (an inner body may not have an IMU attached to it), and learn to solve IMTPs with require sensor-to-segment calibration. Finally, multiple sequences are stacked to build a training batch. These input and output arrays ($\mathbf{X}$ and $\mathbf{Y}$) directly align with the provided software implementation of RING, see Appendix 4.5.

Algorithm 1 `RCMG` (Generate One Training Data Pair)

Input: $\boldsymbol{\lambda}_N$
Output: $\mathbf{X} \in \mathbb{R}^{T \times N \times 9}$, $\mathbf{Y} \in \mathbb{H}^{T \times N}$
$\texttt{sys} \leftarrow \texttt{randSys}\,(\boldsymbol{\lambda}_N)$ {see Algorithm 2}
$\mathbf{T} \leftarrow \texttt{randMotion}\,(\texttt{sys})$ {see Algorithm 3}
$\mathbf{X}, \mathbf{Y} \leftarrow \texttt{getXY}(\texttt{sys}, \boldsymbol{\lambda}_N, \mathbf{T})$ {see Algorithm 4}

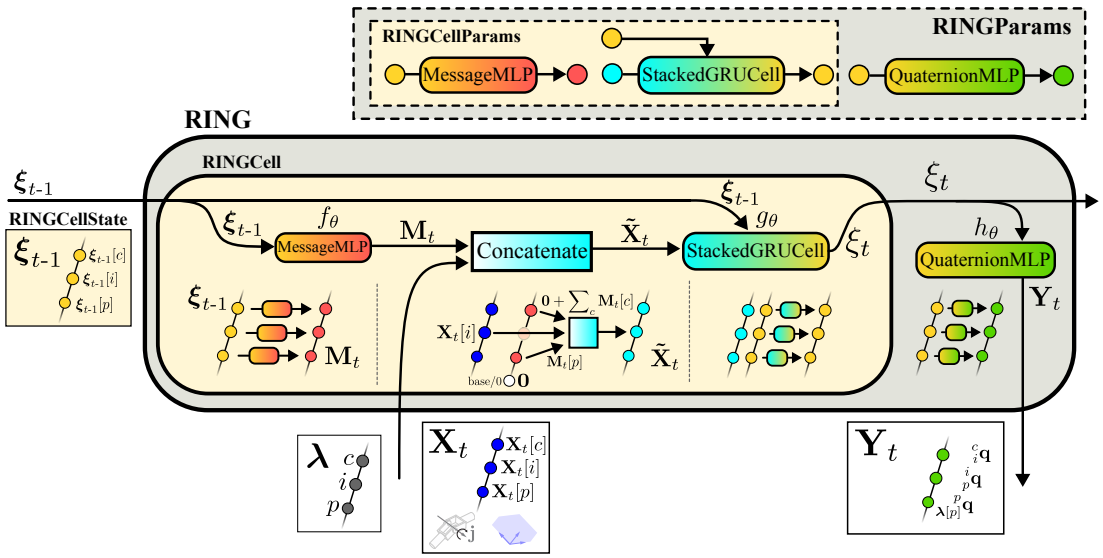


Figure 5: The architecture of the plug-and-play IMT solution RING. It consists of the RNN cell RINGCell and a final MLP head that returns one quaternion per node in the graph. RING is trained to estimate child-to-parent orientations from the available local IMU and joint axis data and nearest neighbour messages. To this end, RINGCell applies a shared set of parameters on a decentralized, per-node level and passes messages along the edges of the graph to allow for information exchange. Note that while the parameters are shared, the hidden states are not shared across nodes. This architecture allows RING to apply a single set of parameters across a broad range of IMTPs, which may vary in aspects such as the number of segments, and it ultimately enables RING's remarkable pluripotency.

4.3 The Architecture of RING

RING is based on a decentralized network of message-passing RNNs which allows for information exchange along the edges of a graph (as given by the CG). RING can be applied to an arbitrary CG (see Section 3.3) and it maps per-node-input to per-node-output. *Most importantly, the parameters of RING are shared across the nodes of the graph such that the number of parameters of RING does not depend on the given CG.* The hidden state, however, is not shared across the nodes. This allows for the training of a single RING network which then solves a variety of IMTPs (see Section 4.1).

The introduction of RING stems from the requirement of acquiring a single pretrained network that can solve different IMTPs with a varying number of inputs and outputs, e.g., estimate two orientations for a two-segment KC, and three orientations for a three-segment KC. This is possible because on a node (or segment) level, we can guarantee static input (at most one IMU, at most a known joint axis direction) and output shapes (one orientation relative to parent). But the per-node-output cannot be estimated from only the per-node-input as the orientation relative to the parent depends on the orientation state of the parent, and orientation w.r.t. the base cannot be estimated from 6D IMU data, additionally, the segment may not have an IMU attached. Thus, we have to allow information exchange between nodes and propose a scheme which again gives rise to local static input and output shapes of messages. Finally, we claim that the estimation of the entire pose in a hierarchical approach is inherently natural and provides an advantageous structural prior to subsequent parameter learning which we explore in Section 5.4.

RING consists of a novel RNN cell, named RINGCell, and a final Multi-Layer Perceptron (MLP) head such that the network's per-node-output is a single unit quaternion (per timestep). Note that the network head is independent of the RINGCell and may easily be replaced to suit different needs.

Let $\boldsymbol{\lambda}$ be a CG with $N \in \mathbb{N}$ nodes, let $F \in \mathbb{N}$ be the number of input features per node, let $H \in \mathbb{N}$ be the half-hidden state dimensionality, and let $M \in \mathbb{N}$ be the dimensionality of the latent messages passed inside the cell based on the edges in the CG. Then, let $\boldsymbol{\xi}_{t\text{-}1} \in \mathbb{R}^{N \times 2H}$ be the hidden state of the RINGCell from the previous timestep $t-1$, and let $\mathbf{X}_t \in \mathbb{R}^{N \times F}$ be the F inputs for all N nodes at time t. Then, the next hidden state $\boldsymbol{\xi}_t$ is obtained by

$$\boldsymbol{\xi}_t = \texttt{ringCell}\left(\boldsymbol{\xi}_{t\text{-}1}, \mathbf{X}_t, \boldsymbol{\lambda}\right) \quad \forall t \tag{3}$$

with $\boldsymbol{\xi}_0 = \mathbf{0}$.

Internally, a RINGCell has the parameters of

- a Message-MLP-network $f_\theta : \mathbb{R}^H \to \mathbb{R}^M$ (three layers, single hidden layer, hidden layer size H, ReLU activations, no final activation), and
- a Stacked-GRUCell-network $g_\theta : \mathbb{R}^{2H} \times \mathbb{R}^{2M+F} \to \mathbb{R}^{2H}$ which consists of the sequence of Gated-Recurrent-Unit(GRU)Cell, LayerNorm, GRUCell (Cho et al., 2014). Note that θ is symbolic for the whole set of parameters of the Stacked-GRUCell-network and that it is different to the parameters of f_θ.

Note that the two GRUCells each have a hidden state dimensionality of H, thus the hidden state of the Stacked-GRUCell-network is of dimensionality of $2H$.

Then, a RINGCell has three consecutive steps, $\forall i = 1 \dots N$:

1. Messages $\mathbf{M}_t \in \mathbb{R}^{N \times M}$ are computed.

$$\mathbf{M}_t[i] = f_\theta\left(\boldsymbol{\xi}_{t\text{-}1}[i, H{:}]\right) \tag{4}$$

2. Messages are passed and latent input $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times (2M+F)}$ computed.

$$\tilde{\mathbf{X}}[i] = \texttt{concat}\left(\mathbf{M}_t\big[\boldsymbol{\lambda}[i]\big], \mathbf{0} + \sum_{c \in \mu_{\boldsymbol{\lambda}}(i)} \mathbf{M}_t[c], \mathbf{X}_t[i]\right) \tag{5}$$

where $\mathbf{M}_t\big[\boldsymbol{\lambda}[i]\big]$ is $\mathbf{0}$ if $\boldsymbol{\lambda}[i] = 0$.

3. Hidden state is updated.

$$\boldsymbol{\xi}_t[i] = g_\theta\left(\boldsymbol{\xi}_{t\text{-}1}[i], \tilde{\mathbf{X}}[i]\right) \tag{6}$$

The architecture of RING is finished by piping the hidden state $\boldsymbol{\xi}_t$ through a final network head, the Quaternion-MLP that combines

- a Layernorm, and a MLP-network $h_\theta : \mathbb{R}^H \to \mathbb{R}^4$ (three layers, single hidden layer, hidden layer size H, ReLU activations, no final activation).

The Quaternion-MLP has two consecutive steps, $\forall i = 1 \dots N$:

1. Unnormalized output $\tilde{\mathbf{Y}} \in \mathbb{R}^{N\times 4}$ is computed.

$$\tilde{\mathbf{Y}}[i] = h_\theta\Big(\texttt{layernorm}\left(\boldsymbol{\xi}_t[i, H{:}]\right)\Big) \tag{7}$$

2. Normalize to allow interpretation as unit quaternions. One unit quaternion per node. Final RING output $\hat{\mathbf{Y}}_t \in \mathbb{H}^N$.

$$\hat{\mathbf{Y}}_t[i] = \frac{\tilde{\mathbf{Y}}[i]}{\sqrt{\sum_{j=1}^4 \tilde{\mathbf{Y}}[i,j]^2}} \tag{8}$$

Note that the normalizing operation, due to its square-root operation, requires special care to allow for successful backpropagation. Also note that the employed loss function, see Section 3.6, is based on a arctan expression, which does not require any special care, in contrast to seemingly equivalent expressions to extract the angle from a quaternion that are based on arccos.

Finally, by combining the equations (3), (7), (8) and unrolling the RNN in time, we can view the entire RING network as a function that maps the timeseries of available input data $\mathbf{X} \in \mathbb{R}^{T\times N\times F}$ and CG $\boldsymbol{\lambda} \in \mathbb{N}^N$ to the timeseries of predicted output data $\hat{\mathbf{Y}} \in \mathbb{H}^{T\times N}$, i.e.

$$\hat{\mathbf{Y}} = \texttt{ring}_\theta\left(\mathbf{X}, \boldsymbol{\lambda}\right) \tag{9}$$

where the parameters of RING are given by the set $\{f_\theta, g_\theta, h_\theta\}$. The hyperparameters of RING are $H \in \mathbb{N}$ and $M \in \mathbb{N}$. Note that the set of parameters of RING is influenced by the hyperparameters of RING and the number of input features per node F, but *not* by the GC $\boldsymbol{\lambda}$ or the number of bodies. For example, a single RING network can be used for predicting the orientations of both two-segment or three-segment KCs even though the number of bodies and, consequently, the dimensionality of input and output arrays is different, e.g., $\mathbf{X} \in \mathbb{R}^{T\times 2\times F}$ compared to $\mathbf{X} \in \mathbb{R}^{T\times 3\times F}$.

4.4 RING: A Single IMT Solution to a variety of IMTPs

In this section, we use the *simulated* training data from Section 4.2 to train a network based on the RINGCell architecture (see Section 4.3). The trained network is then called RING and, with a single set of parameters, RING shows its pluripotency in a range of *experimental* scenarios, from simple single-joint tracking to complex four-segment KCs, without reliance on magnetometers as will be shown in Section 5.

For each of the four KCs $\boldsymbol{\lambda}_N$ (see Definition 4.1), we use the RCMG (see Algorithm 1) to simulate 512 input-output pairs and stack them to create a single batch of training data. This batch of training data is used to update the parameters of the RING network (see Section 4.3), with $H = 400$ and $M = 200$ (total parameter count: 2 337 404) by minimizing the mean-squared-(orientation-estimation)-error, i.e.,

$$\min_\theta \frac{1}{10T} \sum_{N\in\{1,2,3,4\}} \mathop{\mathbb{E}}_{\mathbf{X},\mathbf{Y}\sim\texttt{RCMG}(\boldsymbol{\lambda}_N)} \sum_{t=1}^{T}\sum_{i=1}^{N} \texttt{loss}\left(\mathbf{Y}, \texttt{ring}(\mathbf{X}, \boldsymbol{\lambda}_N)\right)[t,i] \tag{10}$$

where `RCMG` is given by Algorithm 1, `loss` is given by eq. (2), `ring` is given by eq. (9), and where the expectation $\mathbb{E}$ is estimated using 512 draws.

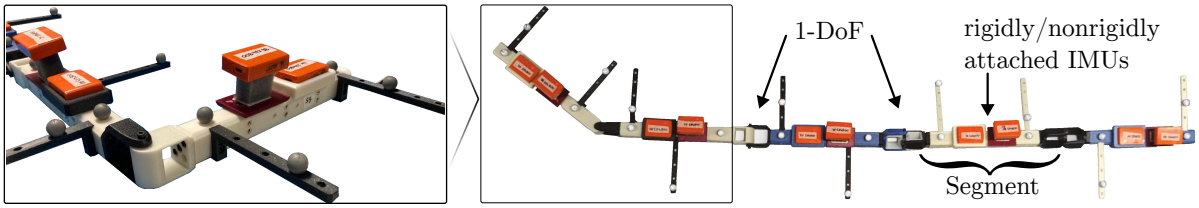


Figure 6: Experimental five-segment KC with ten IMUs (orange boxes) and 20 OMC markers (grey spheres). It uses a single spherical joint followed by three hinge joints, each oriented along the x, y, and z axes, respectively. Each segment of the KC was equipped with two IMUs: one firmly attached to the segment and another affixed nonrigidly using foam padding. The experimental KC is moved in space, and the inertial and optical data is recorded to validate RING and compare it to SOTA methods across a broad range of IMTPs.

We use the LAMB optimizer (You et al., 2020) combined with global (threshold = 0.1) and adaptive (threshold = 0.2) gradient clipping, a cosine decaying learning rate (initial learning rate = 1×10^{-3}, and Truncated Backpropagation Through Time (TBTT). Borrowing the notation from Williams & Peng (1990), we utilize TBTT(10 s, 10 s), i.e., the gradients are stopped and applied after 10 s instead of the total length of 60 s. This results in every batch generation (or episode) corresponding to six parameter updates. We train for 5000 episodes using a single node with eight A40 GPUs (each 48 gigabytes of VRAM). Due to the large amounts of training data, overfitting is seemingly impossible and any form of early stopping did not prove to be required. This has been also confirmed with a separate validation dataset. Training has been stopped after 5000 episodes as the loss has no longer improved.

For practical applications, the inference time and computational requirements of RING are more important than its training requirements. To this end, a theoretical and empirical analysis of the time complexity of RING is conducted in Appendix D, and the findings are summarized here. The theoretical time complexity to advance the prediction of RING by one timestep is $\mathcal{O}(N \times H \times (H + M + F))$. This translates in practice to an efficient NN that enables real-world online application even on low-end hardware. For example, on a single-core Intel Xeon at 2 GHz, RING can comfortably enable motion tracking of a four-segment KC at more than 500 Hz, which is well above typical IMU sampling rates that range from 90 to 286 Hz (Laidig et al., 2021).

4.5 Openly-available Code and Data

The code and experimental validation data is hosted at `https://github.com/simon-bachhuber/ring_supplementary_material` and the repository contains implementations of the RCMG, RINGCell, and RING as decoupled components. Additionally, it includes the code of SOTA methods and validation data to create the experimental validation results (AMAEs and RMAEs) of RING and the SOTA methods that are discussed in Section 5. Finally, we also provide files to retrain RING from scratch, without requiring any real-world training data files. The software, most notably, uses the JAX and Haiku frameworks (Bradbury et al., 2018; Hennigan et al., 2020). The ease-of-use of RING is demonstrated through a code example in the Appendix E.

5 Experimental Validation of RING

In this section, we evaluate the accuracy of RING with one common set of pretrained parameters (see Section 4.4) across a broad range of *experimental* IMTPs (see Section 4.1). *In general, RING shows remarkable pluripotency in zero-shot generalization to real-world experiments across diverse IMTPs. This underscores its broad applicability.*

In Section 5.1, we describe the experimental setup used to evaluate RING on real-world data. We show that RING successfully solves multiple previously solved challenging IMTPs and competes with the current SOTA methods (Section 5.2). Impressively, RING further achieves accurate tracking in even more challenging IMTPs, including two IMTPs that have not been solved before (Section 5.3).

5.1 Experimental Setup and Evaluation Metric

Experimental evaluation is conducted on a five-segment KC which is shown in Figure 6. The KC is constructed by a concatenation of a single spherical joint followed by three hinge joints, each oriented along the x, y, and z axes, respectively. Each segment of the KC was equipped with two IMUs (MTw Awinda, Xsens, Enschede, the Netherlands): one firmly attached to the segment and another affixed nonrigidly using foam padding. This foam-attachment of IMU was conducted to investigate the reduction of motion artifacts. The left panel in Figure 6 shows a close-up of one segment of the KC with one exemplary nonrigidly attached IMU. Each segment is equipped with at least three noncollinearly placed reflective markers that are used to obtain ground truth orientations from a twelve camera Optical Motion Capture (OMC) system (OptiTrack Prime x22, NaturalPoint Inc., Corvalis, USA).

Two distinct trials were conducted, involving random movements of the five-segment KC, by introducing manual alterations of the KC pose by two experienced scientists. The first trial, spanning a duration of 66 s, featured a diverse range of motions, extending from very slow to notably rapid movements. Moreover, the second trial, with a length of 68 s, not only encompassed a variety of motions but also incorporated random intervals of complete stillness, wherein the KC remained motionless.

All trials were preprocessed in the following way: NaN values were removed, an offline time-synchronization was employed via cross-correlation between measured (IMUs) and approximated angular velocities (OMC), and all collected data was resampled to a uniform sampling rate of 100 Hz. Additionally, we correct for any small misalignment between the rigidly attached IMUs' local coordinate systems and the corresponding segments' body coordinate systems as spanned by the OMC markers. Simultaneously, we align the OMC's reference coordinate system with the earth reference coordinate system (the base), as observed by the IMUs. For more comprehensive details regarding these preprocessing steps and the software implementations utilized, readers are referred to the study Laidig et al. (2021).

The rich experimental data obtained from the five-segment KC enables us to perform evaluations on subsets of data and the KC, which represent a variety of different IMTPs. For example, for validation of a one-segment KC, we use the recorded data of all five segments of the five-segment KC independently for evaluation, which effectively increases the amount of validation data by a factor of five. Similarly, for validation of a two-segment KC with a hinge joint, we use three different sub-KCs with a joint axis direction along the x, y, and z axes (the two-segment sub-KC of segment one and segment two is excluded due to its spherical joint). Similarly, for a three-segment KC with double hinge joints, we use two different sub-KCs, and for a four-segment KC with triple hinge joint we use the sub-KC of segment two to five.

Assessing KC pose estimation performance of RING in comparison with existing SOTA and thus solving the IMTP under consideration, requires the usage of a suitable evaluation metric. As already discussed in Section 3.6, the expression $\texttt{angle}(\mathbf{q} \otimes \hat{\mathbf{q}}^*)$ can be used to compare the difference between a ground truth orientation $\mathbf{q}$ and the corresponding predicted orientation $\hat{\mathbf{q}}$. Thus, in order to compare the timeseries of ground truth $\mathbf{Y} \in \mathbb{H}^{T \times N}$ and predicted pose $\hat{\mathbf{Y}} \in \mathbb{H}^{T \times N}$ of an N-segment KC with GC $\boldsymbol{\lambda}_N$, we compute the Attitude-Mean-Absolute-(orientation)-Error (AMAE) and Relative-Mean-Absolute-(orientation)-Error (RMAE) which are given by

$$\texttt{AMAE}\left(\mathbf{Y}, \hat{\mathbf{Y}}\right) = \frac{1}{T} \sum_{t=500}^{T} \left| \texttt{angle}\left(\texttt{zeroHead}\left(\mathbf{Y}[t,1]\right) \otimes \texttt{zeroHead}\left(\hat{\mathbf{Y}}[t,1]\right)^* \right) \right| \tag{11}$$

$$\texttt{RMAE}\left(\mathbf{Y}, \hat{\mathbf{Y}}\right) = \frac{1}{TN} \sum_{t=500}^{T} \sum_{i=2}^{N} \left| \texttt{angle}\left(\mathbf{Y}[t,i] \otimes \hat{\mathbf{Y}}[t,i]^* \right) \right| \tag{12}$$

where $|.|$ denotes the absolute value and `zeroHead` removes the heading component (see Definition B.7). Note that initial 5 s (equaling to an index of 500 at 100 Hertz) of each timeseries were deliberately excluded from the AMAE and RMAE calculations. This decision was made to ensure that the recorded errors accurately reflected the method's performance after convergence.

From Section 3.2, recall that magnetometer-free IMT estimates one absolute attitude, and $N-1$ relative orientations. This difference is captured by the two metrices AMAE and RMAE. Mathematically, AMAE

reports zero angle error if the ground truth orientation and estimated orientation are equal up to an arbitrary heading difference. Whereas, RMAE reports zero angle error if and only if both orientations are exactly identical. The implications are that a low AMAE indicates that the inclination of the system is correctly estimated but the entire system might still be rotated with an arbitrary yaw angle. A low RMAE means that the entire internal pose of the system is accurately estimated.

5.2 RING Unifies Prior Work

5.2.1 Attitude Estimation

Attitude estimation in real-world environments using inertial sensors is a vital prerequisite for a wide range of applications, including tracking human movement and enabling autonomy in air and land vehicles. The attitude estimation problem is defined as estimating the orientation of an object with respect to the horizontal plane with a single sensor (Weber et al., 2021).

The widely employed and long-standing methods from Madgwick (2010); Mahony et al. (2008); Seel & Ruppin (2017) have recently been outperformed with SOTA approaches by Weber et al. (2021); Laidig & Seel (2023). Table 1 shows that RING aligns with SOTA performance in attitude estimation, as indicated by the experimental AMAEs. More specifically, the AMAE of the attitude for Madgwick (2010) is $(2.25 \pm 0.81)^\circ$, for Laidig & Seel (2023) is $(1.61 \pm 1.04)^\circ$, for Weber et al. (2021) is $(2.06 \pm 1.03)^\circ$, for Mahony et al. (2008) is $(2.09 \pm 0.87)^\circ$, for Seel & Ruppin (2017) is $(2.56 \pm 0.93)^\circ$, and for RING is $(2.13 \pm 0.91)^\circ$.

5.2.2 Magnetometer-free Tracking of 1-DoF Joint

IMT of connected segments without relying on magnetometers is desirable for numerous applications where the magnetic field is typically disturbed, including control of industrial robotic manipulators, interconnected drones in automated warehouses, and human motion analysis in hospital environments.

Here, we consider the IMTP of tracking the relative motion between two segments from two 6D IMUs, assuming a single hinge joint with a known axis direction connects both segments.

Overcoming the use of magnetometers is typically achieved by combining SOTA approaches for attitude estimation and utilizing joint-specific constraints that exploit knowledge of the hinge joint axis direction (Laidig et al., 2017; Lehmann et al., 2020). Additionally, a magnetometer-reliant method uses 9D VQF (Laidig & Seel, 2023) for both IMUs independently and does not exploit any kinematic constraint between the two body parts. The RMAEs are reported in Table 1, and they are: for the VQF-based baseline $(19.36 \pm 8.02)^\circ$, for Lehmann et al. (2020) $(4.15 \pm 2.05)^\circ$, for Laidig et al. (2017) $(3.32 \pm 2.12)^\circ$, and for RING $(3.52 \pm 1.00)^\circ$. This shows that RING aligns with the SOTA methods, which are already a non-trivial combination of two separate methods which, in contrast to applying RING, requires expert knowledge.

5.2.3 Magnetometer-free Tracking of 1-DoF Joint with Unknown Joint Axis Direction

The IMTP of Section 5.2.2 can be made more challenging by assuming that the hinge joint axis direction is not known. Then, it can first be estimated from the IMU data, and then subsequently a method that assumes a known joint axis direction can be applied.

A SOTA method for hinge joint axis direction estimation is given by Olsson et al. (2020). RING out-of-the-box supports an unknown hinge joint axis direction by its versatility to drop out the respective node input and replacing it with zeros.

The experimental RMAEs are given in Table 1; combining Olsson et al. (2020) with Lehmann et al. (2020) results in $(4.06 \pm 2.23)^\circ$, and with Laidig et al. (2017) results in $(3.18 \pm 2.05)^\circ$, and RING achieves $(3.92 \pm 1.40)^\circ$. This shows that RING aligns with SOTA performance, which *comprises three distinct methods*.

5.2.4 Three-Segment Sparse IMT

Only few recent works have achieved the combination of sparse and magnetometer-free IMT. One such challenging IMTP is, the tracking of all relative segment orientations of a three-segment KC, with double

hinge joints and known hinge joint axes directions, by using only two IMUs placed on the outer segments. The SOTA method for this IMTP is given by Bachhuber et al. (2023). The absolute attitude can be easily tracked using any of the methods from Section 5.2.1.

As shown in Table 1, the experimental RMAE using Bachhuber et al. (2023) is $(5.60 \pm 2.35)^\circ$, and using RING the RMAE is reduced to $(4.14 \pm 0.53)^\circ$ and, consequently, RING outperforms the SOTA in this IMTP.

5.3 RING Goes Beyond the SOTA

5.3.1 Motion Artifact Reduction

The efficacy of all SOTA IMT methods heavily rely on a rigid sensor-to-body attachment. In many practical scenarios, such as human motion analysis, this assumption is rapidly violated, resulting in strongly degraded estimation accuracies, attributed to a model-reality mismatch.

Here, we consider the IMTP of Section 5.2.2, however here, the IMUs are not rigidly attached to the two segments, while the estimation target remains the relative orientation between the two segments (and not between the coordinate systems of the two IMUs), and the absolute attitude of one of the segments.

Currently, there exist no method that does not make the assumption of rigid IMU attachment. Still, the methods from Section 5.2.1 can be applied to estimate the absolute attitude, and we use the most accurate estimator VQF (Laidig & Seel, 2023) for comparison purposes. The methods from Section 5.2.2 are applied to estimate the relative orientation. To compensate for the violation of the rigid-IMU-attachment assumption, we additionally apply an intuitive low-pass filter (LPF) step to the estimated absolute attitude and relative orientation for each baseline method to suppress unwanted high frequency artifacts. The cutoff frequency was grid searched and we report only the best result for each baseline. Alternatively, the use of the LPF on the estimated orientations prior to computing the relative orientation did not yield better performance.

In Table 1, column 5.3.1A reports the experimental AMAE in the attitude estimate, and demonstrates RING's superior performance of $(7.59 \pm 2.85)^\circ$ over the combination of VQF+LPF with an AMAE of $(9.19 \pm 2.31)^\circ$. Similarly, column 5.3.1B reports experimental RMAEs, they are: Lehmann et al. (2020) achieves $(8.00 \pm 2.78)^\circ$, Laidig et al. (2017) achieves $(7.00 \pm 1.57)^\circ$, and RING achieves $(5.56 \pm 2.33)^\circ$. This shows that RING outperforms the SOTA methods. Note that the reduced AMAE shows that RING has learned to fuse the information of the second segment's IMU into the attitude estimation of the first segment.

5.3.2 Three-Segment Sparse IMT with Unknown Joint Axes Directions

The IMTP considered in Section 5.2.4 can be made even more challenging by not assuming known joint axes directions.

To the best of the authors' knowledge, there currently exists no method that is applicable in such a challenging IMTP. RING can solve this IMTP with only a modest increase in error (Table 1), given the increased complexity of the task. Note that the direction of the joint axis cannot be estimated using Olsson et al. (2020) such that the method from Section 5.2.4 may be applied, as Olsson et al. (2020) does not allow for a sparse sensor setup which inherently requires a pluripotent approach such as RING. We report a RMAE value of $(5.37 \pm 0.71)^\circ$ for RING in this challenging IMTP.

5.3.3 Four-Segment Sparse IMT: 3-DoFs between IMUs

Increasing sensor sparsity will naturally make an IMTP problem more complex. A KC configuration with four segments and only two 6D IMUs results in three DoFs (three consecutive hinge joints) between the two outer-segment 6D IMUs and it represents the limit of accurate sparse IMT. Despite the complexity, the estimation target remains to capture all three relative orientations.

To the best of the authors' knowledge, there currently is no method that is applicable in such a challenging IMTP. RING can solve this problem formulation sufficiently well. When assuming known joint axes directions for the three hinge joints, RING achieves a RMAE of $(6.78 \pm 1.41)^\circ$. An exemplary trial is shown in Figure 7.

Table 1: Motion tracking accuracy (in degrees) of RING compared to various SOTA methods across a variety of IMTPs. While previous methods are problem-specific and Not Applicable (NA) to many IMTPs, RING is the only method that accurately solves all problems. All columns report the RMAE, see eq. (12), as metric, except for the columns 5.2.1 and 5.3.1A which report AMAE as metric, see eq. (11).

IMTPs Method	5.2.1	5.2.2	5.2.3	5.2.4	5.3.1a	5.3.1b	5.3.2	5.3.3
(1)	2.06 ± 1.03	NA	NA	NA	$\geq(5)$	NA	NA	NA
(2)	2.25 ± 0.81	$\geq(5)$	$\geq(5)$	NA	$\geq(5)$	NA	NA	NA
(3)	2.09 ± 0.87	$\geq(5)$	$\geq(5)$	NA	$\geq(5)$	NA	NA	NA
(4)	2.56 ± 0.93	$\geq(5)$	$\geq(5)$	NA	$\geq(5)$	NA	NA	NA
(5)	1.61 ± 1.04	$\rightarrow$	19.3 ± 8.02	NA	9.20 ± 2.31	24.9 ± 17.6	NA	NA
(5)+(6)	$\uparrow$	3.32 ± 2.12	NA	NA	$\uparrow$	7.00 ± 1.57	NA	NA
(5)+(7)	$\uparrow$	4.15 ± 2.05	NA	NA	$\uparrow$	8.00 ± 2.78	NA	NA
(5)+(6)+(8)	$\uparrow$	$\rightarrow$	3.18 ± 2.05	NA	$\uparrow$	8.50 ± 2.60	NA	NA
(5)+(7)+(8)	$\uparrow$	$\rightarrow$	4.06 ± 2.23	NA	$\uparrow$	7.90 ± 2.48	NA	NA
(9)	NA	NA	NA	5.60 ± 2.35	NA	NA	NA	NA
RING	2.13 ± 0.91	3.52 ± 1.00	3.92 ± 1.40	4.14 ± 0.53	7.59 ± 2.85	5.56 ± 2.33	5.37 ± 0.71	6.78 ± 1.41

Methods: Weber et al. (2021)(1), Madgwick (2010)(2), Mahony et al. (2008)(3), Seel & Ruppin (2017)(4), Laidig & Seel (2023)(5), Laidig et al. (2017)(6), Lehmann et al. (2020)(7),Olsson et al. (2020)(8), Bachhuber et al. (2023)(9)
$\geq (i)$ refers to the AMAE or RMAE of being expected to be larger or equal than for method (i)
$\uparrow$ or $\rightarrow$ indicate that the AMAE or RMAE is equal to the AMAE or RMAE of the cell above or to the cell to the right

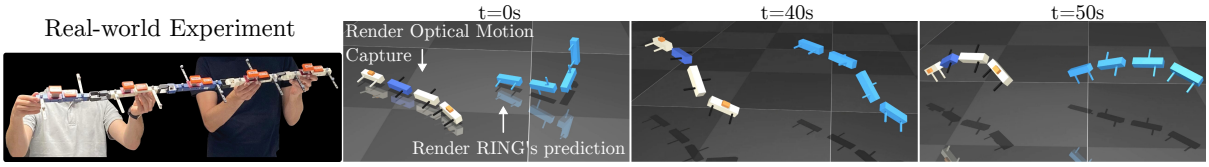


Figure 7: Exemplary frames that showcase RING's performance for the IMTP that involves a four-segment KC with sparse IMU attachments and known joint axes directions (see Section 5.3.3). It is a remarkable first that RING can accurately estimate the four orientations (one absolute and three relative orientations) from only two magnetometer-free IMUs. A video of the trial is available here.

When assuming unknown joint axes directions, RING achieves $(13.66 \pm 3.07)°$. This shows that with unknown joint axes directions, this IMTP pushes the limits of observability (Bachhuber et al., 2022).

5.4 The Decentralized Approach of RING Provides an Advantageous Structural Prior

In this section, we showcase a second advantage of the decentralized approach of RING over a centralized approach that is typically employed, e.g., by stacking multiple LSTM- or GRU-Cells. First, recall that RING is based on a decentralized network of message-passing RNNs which allows for the training of a single set of parameters despite a varying number of bodies in the KC and, while the latter aspect is the core motivation behind this architecture it is, interestingly, not the sole motivation behind the decentralized approach. The second motivation is that the estimation of the entire pose in RING's decentralized approach provides an advantageous structural prior compared to an RNN that utilizes a centralized approach. For this purpose, we compare RING to the RNN-based Observer (RNNO), a deep GRU network with intermediate Layernorm layers, as it was proposed in Bachhuber et al. (2023). RNNO maps all available input data to the entire targeted pose data and does not utilize the specific graph structure of the IMTP. RNNO has been proposed as the solution of a specific IMTP (see Section 5.2.4) and, for this IMTP, both RING and RNNO achieve similar performance. However, this is not the case if the IMTP becomes vastly more complex. Consider, e.g., the IMTP of Section 5.3.3, which is arguably the most challenging IMTP under consideration in this work. We have trained RNNO on the subset of training data of RING that corresponds to this IMTP. Despite

Table 2: Motion tracking accuracy of RING solving the IMTP defined in Section 5.2.3 on external datasets. RING provides consistently low errors demonstrating robustness across different IMU hardware.

Dataset	IMU Hardware	T[s]	Native Rate [Hz]	RMAE [°]
Ours	Xsens MTw Awinda	402	40	3.92 ± 1.40
RepoIMU (1)	microIMU (2)	390	90	3.44 ± 0.06
OpenAXES (3)	Bosch BMI160 + Analog Devices ADXL355	227	≈ 125	2.52 ± 0.29

Szczęsna et al. (2016) (1), Jędrasiak et al. (2013) (2), Webering et al. (2023) (3)

varying several hyperparameters, the best reported RMAE of RNNO is $(18.72 \pm 5.12)°$. This is significantly higher compared to RING's RMAE of $(6.78 \pm 1.41)°$, even though RING is maintaining its applicability to a broad range of IMTPs, and is not purpose-trained for a single IMTP. This showcases that the decentralized approach of RING provides an advantage even if only the solution of a single, specific IMTP is required.

5.5 RING's Robustness to Different IMUs

IMUs in general, and especially across vendors, differ in properties like noise density and bias offset. RNN-based inertial sensor fusion has been demonstrated to generalize across different sensor hardware (Weber et al., 2021). Nonetheless, to ensure broad real-world applicability, we investigate RING's robustness to different IMU hardware. First, we analyze how RING's performance scales as noise and bias properties are incrementally increased. Then, we evaluate RING on openly-available, real-world datasets that use IMUs from different vendors.

To simulate reasonable noise density and bias offset ranges, we constructed a worst-case IMU by combining the worst properties of various IMU manufacturers, as summarized in Table 3. Then, those worst-case noise and bias values are incrementally (in seven equidistant steps) increased from 0 to 120%, and a corresponding amount of simulated noise and bias is added to our real-world IMU data (see Section 5.1), which yields seven modified datasets. RING and all SOTA methods are validated on the modified dataset, and this procedure is repeated using ten different seeds. The AMAEs and RMAEs of all methods are plotted for all IMTPs as a function of the seven steps in Figure 9. The figure shows that, whilst the performance of all methods (as expected) slightly worsens as noise and bias are increased, RING maintains accuracy comparable to SOTA methods in all IMTPs and, especially in two-segment KC tracking, substantially outperforms them.

The RepoIMU dataset (Szczęsna et al., 2016) contains real-world IMU and ground truth data from passive motions of a swinging pendulum. It uses non-commercial micro IMUs (Jędrasiak et al., 2013) with a focus on low cost and small size over accuracy. The OpenAXES Robot Dataset contains data from motions of a robot arm drawing different shapes at different speeds. IMUs are attached to each segment, and the ground truth is known from the robot's encoders. Table 2 reports RING's performance for both external datasets.

6 Discussion

We have shown that a single RNN, named RING, can accurately solve a broad range of IMTPs even if two IMTPs do not have same input-output dimensionality. This pluripotent behavior originates from a decentralized architecture that provides an advantageous structural prior compared to the more typical centralized approach which, in contrast to RING, has the additional limitation that it can only be utilized if the solution of a single IMTP is sufficient. In summary, for the set of IMTPs under consideration (see Section 4.1), we have shown that RING can enable accurate IMT for one-, two-, and three-segment KCs as long as the IMUs are rigidly attached. Most notably, this includes an IMTP that is sparse, magnetometer-free, and requires sensor-to-segment calibration. If IMUs are nonrigidly attached, then RING is shown to provide accurate orientation estimates for a two-segment KC. RING can achieve accurate IMT for a four-segment KC, provided that the IMUs are rigidly attached and sensor-to-segment calibration is not required. It is a remarkable first, that RING can track the pose of the four-segment KC (which requires in total four distinct orientations) using only two IMUs. Note that RING does not require problem specific priors, although their

introduction, i.e., a known joint axes direction, or an additionally attached IMU is straightforwardly possible as additional input data to improve the motion tracking accuracy even further. This can be seen in Table 1 where, e.g., for a three-segment KC, providing joint axes directions decreases the mean RMAE from 5.37° (Section 5.3.2) to 4.14 (Section 5.2.4).

7 Conclusion

In the presented work, we combined ideas from the domain of multi-agent systems with RNNs to propose an architecture based on a decentralized network of message-passing, parameter-sharing RNNs. We have successfully exploited this combination for the analysis of structural sequential data and solved a challenging state estimation problem, which is IMT of KCs, by letting the decentralized network map local IMU measurements and nearest-neighbour messages to local orientations. In particular, we introduced RING, a *pluripotent* IMT solution that, unlike all previous, problem-specific approaches, enables plug-and-play non-expert use. RING outperforms a range of problem-specific SOTA solutions and even generalizes to previously unsolved scenarios, including the challenging combination of magnetometer-free and sparse sensing with unknown sensor-to-segment parameters. Remarkably, RING demonstrates the ability to zero-shot generalize to experimental scenarios, despite being trained solely on simulated data. For example, RING can, for the first time, accurately track a real-world four-segment kinematic chain (which requires estimating four orientations) using only two magnetometer-free IMUs.

RING's pluripotency greatly simplifies the application of IMT by eliminating the need for expert knowledge to identify, select, and fine-tune problem-specific methods. This is expected to not only make IMT more powerful and less restrictive in established domains but also to facilitate the accessibility of IMT technology by non-expert users and broadens its applicability to previously untapped domains.

References

Simon Bachhuber, Daniel Weber, Ive Weygers, and Thomas Seel. RNN-based Observability Analysis for Magnetometer-Free Sparse Inertial Motion Tracking. In *2022 25th International Conference on Information Fusion (FUSION)*, pp. 1–8, Linköping, Sweden, July 2022. IEEE. ISBN 978-1-73774-972-1. doi: 10.23919/FUSION49751.2022.9841375. URL https://ieeexplore.ieee.org/document/9841375/.

Simon Bachhuber, Dustin Lehmann, Eva Dorschky, Anne D. Koelewijn, Thomas Seel, and Ive Weygers. Plug-and-Play Sparse Inertial Motion Tracking With Sim-to-Real Transfer. *IEEE Sensors Letters*, 7(10): 1–4, October 2023. ISSN 2475-1472. doi: 10.1109/LSENS.2023.3307122. URL https://ieeexplore.ieee.org/document/10225275.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Ao Buke, Fang Gaoli, Wang Yongcai, Song Lei, and Yang Zhiqi. Healthcare algorithms by wearable inertial sensors: A survey. *China Communications*, 12(4):1–12, April 2015. ISSN 1673-5447. doi: 10.1109/CC.2015.7114054. URL https://ieeexplore.ieee.org/abstract/document/7114054.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, September 2014. URL http://arxiv.org/abs/1406.1078.

Mark Euston, Paul Coote, Robert Mahony, Jonghyuk Kim, and T. Hamel. A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2008. doi: 10.1109/IROS.2008.4650766.

Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, New York, 2008. ISBN 978-0-387-74314-1 978-0-387-74315-8.

Roy Featherstone. A Beginner's Guide to 6-D Vectors (Part 2) [Tutorial]. *IEEE Robotics & Automation Magazine*, 17(4):88–99, December 2010. ISSN 1070-9932, 1558-223X. doi: 10.1109/MRA.2010.939560. URL https://ieeexplore.ieee.org/document/5663690/.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html.

Matthew W. Givens and Calvin Coopmans. A Survey of Inertial Sensor Fusion: Applications in sUAS Navigation and Data Collection. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1054–1060, June 2019. doi: 10.1109/ICUAS.2019.8798225. URL https://ieeexplore.ieee.org/document/8798225.

Aaron Grapentin, Dustin Lehmann, Ardjola Zhupa, and Thomas Seel. Sparse Magnetometer-Free Real-Time Inertial Hand Motion Tracking. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 94–100, September 2020. doi: 10.1109/MFI49285.2020.9235262. URL https://ieeexplore.ieee.org/document/9235262.

Tom Hennigan, Trevor Cai, Tamara Norman, Lena Martens, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control, July 2020. URL http://arxiv.org/abs/2007.04976.

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics*, 37(6):185:1–185:15, December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275108. URL https://dl.acm.org/doi/10.1145/3272127.3275108.

Karol Jędrasiak, Krzysztof Daniec, and Aleksander Nawrat. The low cost micro inertial measurement unit. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 403–408, June 2013. doi: 10.1109/ICIEA.2013.6566403. URL https://ieeexplore.ieee.org/document/6566403.

Manon Kok, Jeroen D. Hol, and Thomas B. Schön. An optimization-based approach to human body motion capture using inertial sensors. *IFAC Proceedings Volumes*, 47(3):79–85, January 2014. ISSN 1474-6670. doi: 10.3182/20140824-6-ZA-1003.02252. URL https://www.sciencedirect.com/science/article/pii/S147466701641596X.

J.B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 2002. ISBN 978-0-691-10298-6. URL https://press.princeton.edu/books/paperback/9780691102986/quaternions-and-rotation-sequences.

Daniel Laidig and Thomas Seel. VQF: Highly accurate IMU orientation estimation with bias estimation and magnetic disturbance rejection. *Information Fusion*, 91:187–204, March 2023. ISSN 1566-2535. doi: 10.1016/j.inffus.2022.10.014. URL https://www.sciencedirect.com/science/article/pii/S156625352200183X.

Daniel Laidig, Thomas Schauer, and Thomas Seel. Exploiting kinematic constraints to compensate magnetic disturbances when calculating joint angles of approximate hinge joints from orientation estimates of inertial sensors. In *2017 International Conference on Rehabilitation Robotics (ICORR)*, pp. 971–976, July 2017. doi: 10.1109/ICORR.2017.8009375. URL https://ieeexplore.ieee.org/document/8009375.

Daniel Laidig, Marco Caruso, Andrea Cereatti, and Thomas Seel. BROAD—A Benchmark for Robust Inertial Orientation Estimation. *Data*, 6(7):72, July 2021. ISSN 2306-5729. doi: 10.3390/data6070072. URL https://www.mdpi.com/2306-5729/6/7/72.

Dustin Lehmann, Daniel Laidig, and Thomas Seel. Magnetometer-free motion tracking of one-dimensional joints by exploiting kinematic constraints. *Proceedings on Automation in Medical Engineering*, 1(1):027–027, February 2020. URL https://www.journals.infinite-science.de/index.php/automed/article/view/335.

Dustin Lehmann, Daniel Laidig, Simon Bachhuber, Thomas Seel, and Ive Weygers. Magnetometer-Free Inertial Motion Tracking of Kinematic Chains, April 2024. URL https://papers.ssrn.com/abstract=4785077.

Irvin Hussein López-Nava and Angélica Muñoz-Meléndez. Wearable Inertial Sensors for Human Motion Analysis: A Review. *IEEE Sensors Journal*, 16(22):7821–7834, November 2016. ISSN 1558-1748. doi: 10.1109/JSEN.2016.2609392. URL https://ieeexplore.ieee.org/document/7567551.

Sebastian O H Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. 2010.

Robert Mahony, Tarek Hamel, and Jean-Michel Pflimlin. Nonlinear Complementary Filters on the Special Orthogonal Group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, June 2008. ISSN 1558-2523. doi: 10.1109/TAC.2008.923738. URL https://ieeexplore.ieee.org/document/4608934.

Timothy McGrath, Richard Fineman, and Leia Stirling. An Auto-Calibrating Knee Flexion-Extension Axis Estimator Using Principal Component Analysis with Inertial Sensors. *Sensors*, 18(6):1882, June 2018. ISSN 1424-8220. doi: 10.3390/s18061882. URL https://www.mdpi.com/1424-8220/18/6/1882.

Fredrik Olsson, Manon Kok, Thomas Seel, and Kjartan Halvorsen. Robust Plug-and-Play Joint Axis Estimation Using Inertial Sensors. *Sensors*, 20(12):3534, January 2020. ISSN 1424-8220. doi: 10.3390/s20123534. URL https://www.mdpi.com/1424-8220/20/12/3534.

Liviu Panait and Sean Luke. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, November 2005. ISSN 1573-7454. doi: 10.1007/s10458-005-2631-2. URL https://doi.org/10.1007/s10458-005-2631-2.

Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c26820b8a4c1b3c2aa868d6d57e14a79-Abstract.html.

Patrik Puchert and Timo Ropinski. A3GC-IP: Attention-oriented adjacency adaptive recurrent graph convolutions for human pose estimation from sparse inertial measurements. *Computers & Graphics*, 117:96–104, December 2023. ISSN 0097-8493. doi: 10.1016/j.cag.2023.09.009. URL https://www.sciencedirect.com/science/article/pii/S0097849323002327.

Thomas Seel and Stefan Ruppin. Eliminating the Effect of Magnetic Disturbances on the Inclination Estimates of Inertial Sensors**This work was conducted within the research project BeMobil, which is supported by the German Federal Ministry of Research and Education (FKZ 16SV7069K). *IFAC-PapersOnLine*, 50(1):8798–8803, July 2017. ISSN 2405-8963. doi: 10.1016/j.ifacol.2017.08.1534. URL https://www.sciencedirect.com/science/article/pii/S2405896317321201.

Thomas Seel, Manon Kok, and Ryan S. McGinnis. Inertial Sensors—Applications and Challenges in a Nutshell. *Sensors*, 20(21):6221, January 2020. ISSN 1424-8220. doi: 10.3390/s20216221. URL https://www.mdpi.com/1424-8220/20/21/6221.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 2252–2260, Red Hook, NY, USA, December 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9.

Luke Sy, Michael Raitor, Michael Del Rosario, Heba Khamis, Lauren Kark, Nigel H. Lovell, and Stephen J. Redmond. Estimating Lower Limb Kinematics Using a Reduced Wearable Sensor Count. *IEEE transactions on bio-medical engineering*, 68(4):1293–1304, April 2021. ISSN 1558-2531. doi: 10.1109/TBME.2020.3026464.

Luke Wicent F. Sy, Nigel H. Lovell, and Stephen J. Redmond. Estimating Lower Limb Kinematics Using a Lie Group Constrained Extended Kalman Filter with a Reduced Wearable IMU Count and Distance Measurements. *Sensors (Basel, Switzerland)*, 20(23):6829, November 2020. ISSN 1424-8220. doi: 10.3390/s20236829.

Agnieszka Szczęsna, Przemysław Skurowski, Przemyslaw Pruszowski, Damian Pęszor, Marcin Paszkuta, and Konrad Wojciechowski. Reference Data Set for Accuracy Evaluation of Orientation Estimation Algorithms for Inertial Motion Capture Systems. volume 9972, pp. 509–520, September 2016. ISBN 978-3-319-46417-6. doi: 10.1007/978-3-319-46418-3_45.

Bertram Taetz, Gabriele Bleser, and Markus Miezal. Towards Self-Calibrating Inertial Body Motion Capture, June 2016. URL http://arxiv.org/abs/1606.03754.

Tom Van Wouwe, Seunghwan Lee, Antoine Falisse, Scott Delp, and C. Karen Liu. Diffusion Inertial Poser: Human Motion Reconstruction from Arbitrary Sparse IMU Configurations, August 2023. URL http://arxiv.org/abs/2308.16682.

Timo von Marcard, Bodo Rosenhahn, Michael J. Black, and Gerard Pons-Moll. Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs, March 2017. URL http://arxiv.org/abs/1703.08014.

He Wang, Wenwu Yu, Guanghui Wen, and Guanrong Chen. Fixed-Time Consensus of Nonlinear Multi-Agent Systems With General Directed Topologies. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(9):1587–1591, September 2019. ISSN 1558-3791. doi: 10.1109/TCSII.2018.2886298. URL https://ieeexplore.ieee.org/document/8574981.

Tingwu Wang, Renjie Liao, and Sanja Fidler. NerveNet: Learning Structured Policy with Graph Neural Networks. In *ICLR*, 2018.

Daniel Weber, Clemens Gühmann, and Thomas Seel. RIANN—A Robust Neural Network Outperforms Attitude Estimation Filters. *AI*, 2(3):444–463, September 2021. ISSN 2673-2688. doi: 10.3390/ai2030028. URL https://www.mdpi.com/2673-2688/2/3/28.

Fritz Webering, Christian Fahnemann, and Nils Stanislawski. OpenAXES example robot dataset, 2023. URL https://data.uni-hannover.de/dataset/a5637b98-acb4-4487-bf1b-a32b42425be4.

Ronald J. Williams and Jing Peng. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 2(4):490–501, December 1990. ISSN 0899-7667. doi: 10.1162/neco.1990.2.4.490. URL https://ieeexplore.ieee.org/document/6797135.

Xinyu Yi, Yuxiao Zhou, and Feng Xu. TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors, May 2021. URL http://arxiv.org/abs/2105.04605.

Xinyu Yi, Yuxiao Zhou, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Feng Xu. Physical Inertial Poser (PIP): Physics-aware Real-time Human Motion Tracking from Sparse Inertial Sensors, March 2022. URL http://arxiv.org/abs/2203.08528.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes, January 2020. URL http://arxiv.org/abs/1904.00962.

Zhaolong Zheng, Hao Ma, Weichao Yan, Haoyang Liu, and Zaiyue Yang. Training Data Selection and Optimal Sensor Placement for Deep-Learning-Based Sparse Inertial Sensor Human Posture Reconstruction. *Entropy*, 23(5):588, May 2021. ISSN 1099-4300. doi: 10.3390/e23050588. URL https://www.mdpi.com/1099-4300/23/5/588.

A Preliminaries

A.1 Notation

- Scalars are lowercase or uppercase, italic, non-bold, e.g., $x \in \mathbb{R}$, or, e.g., $N \in \mathbb{N}$.
- (Column) vectors are lowercase, italic, bold, e.g., $\boldsymbol{x} \in \mathbb{R}^3$, and, e.g., $\boldsymbol{x} = (1, 2, 3)^\intercal$.
- Matrices (or higher) are uppercase, upright, bold, e.g., $\mathbf{X}$ and $\mathbf{X} \in \mathbb{R}^{3\times4}$.
- Individual quaternions are denoted with $\mathbf{q} \in \mathbb{H}$. One-dimensional arrays of quaternions with $\mathbf{q}$, e.g., $\mathbf{q} \in \mathbb{H}^5$. Two dimensional arrays of quaternions with $\mathbf{q}$, e.g., $\mathbf{q} \in \mathbb{H}^{5\times5}$.
- (Programming) functions and structures are written in typewriter typestyle, e.g., `sys`.
- The equal symbol = is overloaded and used for definitions, assignments, and comparison, and the context defines the current meaning.
- The symbol $\mathbf{0}$ defines an arbitrarily large array of zeros that is automatically broadcasted to the required dimensionality.
- The symbol $\mathbb{1}$ defines either the unity element of a given space, or the indicator function, such that, e.g., $\mathbb{1}_0(i)$ is one for $i = 0$ and zero else.
- The symbol $\otimes$ is used to denote the direct product of vector spaces, and additionally denotes quaternion multiplication, see Definition B.2.

A.2 Array Indexing and Slicing

Vectors and matrices are array-like objects that can be indexed and sliced dynamically. Indexing starts with 1 and slicing is inclusive on both sides. For example, let $\mathbf{X} \in \mathbb{R}^{3\times4}$, then $\mathbf{X}[1] \in \mathbb{R}^4$, or $\mathbf{X}[1{:}2] \in \mathbb{R}^{2\times4}$.

Additionally, we define the following auto-completion rules by example, such that $\mathbf{X}[{:}2]$ is equivalent to $\mathbf{X}[1{:}2]$, and $\mathbf{X}[2{:}]$ is equivalent to $\mathbf{X}[2{:}3]$, and $\mathbf{X}[{:}]$ is equivalent to $\mathbf{X}[1{:}3]$. Finally, multiple dimensions can be indexed or sliced simultaneously and are separated by a comma, e.g., $\mathbf{X}[{:}, 3{:}] \in \mathbb{R}^{3\times2}$.

B Quaternion Algebra

Definition B.1. We use $\mathbb{H}$ to denote the space of all unit quaternions, and we denote a unit quaternion with $\mathbf{q} = q_w + q_x i + q_y j + q_z k$.

Definition B.2. Let $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{H}$ be two unit quaternions, then we use $\otimes$ to denote quaternion multiplication of the two unit quaternions, that is

$$\begin{aligned}\mathbf{q}_1 \otimes \mathbf{q}_2 = (&q_{1w}q_{2w} - q_{1x}q_{2x} - q_{1y}q_{2y} - q_{1z}q_{2z}) + (q_{1w}q_{2x} + q_{1x}q_{2w} + q_{1y}q_{2z} - q_{1z}q_{2y})i \\ &+ (q_{1w}q_{2y} - q_{1x}q_{2z} + q_{1y}q_{2w} + q_{1z}q_{2x})j + (q_{1w}q_{2z} + q_{1x}q_{2y} - q_{1y}q_{2x} + q_{1z}q_{2w})k\end{aligned}$$

Note that the space of unit quaternions $\mathbb{H}$ in combination with quaternion multiplication $\otimes$ forms a closed group, i.e., $(\mathbf{q}_1 \otimes \mathbf{q}_2) \in \mathbb{H} \quad \forall \mathbf{q}_1, \mathbf{q}_2$.

Definition B.3. The inverse of a quaternion $\mathbf{q}^{-1}$ is given by the complex conjugate denoted by $\mathbf{q}^*$.

Definition B.4. The quaternion that corresponds to a certain rotation around an axis $\boldsymbol{j} = (j_x, j_y, j_z)^\intercal \in \mathbb{R}^3$ by an angle $\alpha \in \mathbb{R}$ is given by $\mathbf{q} = \texttt{quat}(\boldsymbol{j}, \alpha) = \cos(\frac{\alpha}{2}) + (j_x \sin(\frac{\alpha}{2}))i + (j_y \sin(\frac{\alpha}{2}))j + (j_z \sin(\frac{\alpha}{2}))k$.

Definition B.5. Extracting the angle α from a given quaternion $\mathbf{q}$ (the inverse operation of B.4) can be done using $\texttt{angle}(\mathbf{q}) = 2 \arctan\left(\frac{\sqrt{q_x^2+q_y^2+q_z^2}}{q_w}\right)$.

Definition B.6. We define the projection `project` of a quaternion $\mathbf{q}$ onto a (primary) axis $\mathbf{J}$ as the decomposition into two quaternions, the primary rotation $\mathbf{q}_p$ and the residual rotation $\mathbf{q}_r$, such that $\mathbf{q} = \mathbf{q}_r \otimes \mathbf{q}_p$ while the angle of the residual rotation is minimized. This can be done with

1. $\alpha_p \leftarrow \arctan\left(\frac{j_x q_x + j_y q_y + j_z q_z}{q_w}\right)$
2. $\mathbf{q}_p \leftarrow \texttt{quat}(\boldsymbol{j}, \alpha_p)$
3. $\mathbf{q}_r \leftarrow \mathbf{q} \otimes \mathbf{q}_p^*$

Definition B.7. We define the function `zeroHead` as the function that maps a quaternion $\mathbf{q}$ to the quaternion $\mathbf{q}_r$ with zero heading component and returns $\mathbf{q}_r$. It can be obtained via

1. $\mathbf{q}_p, \mathbf{q}_r \leftarrow \texttt{project}\,(\mathbf{q}, (0,0,1)^\intercal)$

where `project` is given in Definition B.6.

Definition B.8. The function `randQuat` that returns a random quaternion, uniform on the sphere, can be obtained by drawing i.i.d. four numbers from a normal distribution, interpreting them as the components q_w, q_x, q_y, q_z of a quaternion, and then normalizing the quaternion to obtain a unit quaternion.

Definition B.9. The function $\texttt{rotate}(\mathbf{q}, \boldsymbol{r})$ applies a quaternion $\mathbf{q}$ to a vector $\boldsymbol{r} \in \mathbb{R}^3$. If the quaternion is interpreted as ${}^0_1\mathbf{q}$ (from 0 to 1) and the vector is expressed using the unit-vectors of coordinate system 0, then the function `rotate` returns the same vector but using the unit-vectors of coordinate system 1. Let $\boldsymbol{r} = (r_x, r_y, r_z)^\intercal$, then the $\texttt{rotate}(\mathbf{q}, \boldsymbol{r})$ function is given by $(\mathbf{q} \otimes (0, r_x, r_y, r_z)^\intercal \otimes \mathbf{q}^*)\,[1{:}]$.

C Training Data: The RCMG Algorithm

Algorithm 2 `randSys` (RCMG First Step)

Input: $\boldsymbol{\lambda}_N$
Output: `sys`
$\texttt{sys} \leftarrow \texttt{initSys}(\boldsymbol{\lambda}_N)$ {allocate empty structure}
$\texttt{sys} \leftarrow \texttt{randBase}(\texttt{sys}, \boldsymbol{\lambda}_N)$ {see Definition C.1}
for $i = 1$ **to** N **do**
  $\texttt{sys}.\mathbf{J}[i] = \texttt{rotate}(\texttt{randQuat}(), \hat{\boldsymbol{e}}_{\boldsymbol{x}})$ {random hinge joint axis direction; unused if $\texttt{sys}.\boldsymbol{\lambda}[i] = 0$}
  for $d = 1$ **to** 3 **do**
    $\texttt{sys}.\mathbf{R}[i, d] = \texttt{randSegmentToSegment}(d)$ {see Definition C.2}
    $\texttt{sys}.\mathbf{R}[i + N, d] = \texttt{randSensorToSegment}(d)$ {see Definition C.2}
  end for
  {IMU of node 1 of the standard system is always rigidly attached, as there is no second IMU whose measurements may be fused to effectively eliminate motion due to the nonrigid attachment.}
  if $\texttt{sys}.\boldsymbol{n}[1] = i$ **or** $\texttt{randBernoulli}(0.25)$ **then**
    $\boldsymbol{k} \leftarrow \texttt{getRigidStif}()$ {see Definition C.4}
    $\boldsymbol{\gamma} \leftarrow \texttt{getRigidDamp}()$ {see Definition C.4}
  else
    $\boldsymbol{k} \leftarrow \texttt{randNonRigidStif}()$ {see Definition C.5}
    $\boldsymbol{\gamma} \leftarrow \texttt{randNonRigidDamp}()$ {see Definition C.5}
  end if
  $\texttt{sys}.\mathbf{K}[i] = \boldsymbol{k}$
  $\texttt{sys}.\boldsymbol{\Gamma}[i] = \boldsymbol{k} \cdot \boldsymbol{\gamma}$ {element-wise multiplication}
end for

Algorithm 3 `randMotion` (RCMG Second Step)

Input: `sys`, (`motionConfig`) {`motionConfig` object is used only by `randFreeTraj` and `randHingTraj` and it influences and constraints the random motion (see Appendix C.1)}
Output: $\mathbf{T} \in \left(\mathbb{H} \otimes \mathbb{R}^3\right)^{2N \times T}$
$T \leftarrow \texttt{int}\left(\frac{60}{\texttt{sys.T}_\texttt{s}}\right)$ {# timesteps; duration of training sequences is 60 seconds}
$\mathbf{Q} = \mathbf{0}$ {timeseries of minimal coordinates $\mathbf{Q} \in \mathbb{R}^{N_q \times T}$ of system without IMU bodies; see Definition 3.2}
$a \leftarrow 0$
for $i = 1$ **to** N **do**
  if $\texttt{sys}.\boldsymbol{\lambda}[i] = 0$ **then**
    $b \leftarrow a + 7$
    $\mathbf{Q}[a{:}b] = \texttt{randFreeTraj}\left(\texttt{motionConfig}, T\right)$ {see Definition C.6}
  else
    $b \leftarrow a + 1$
    $\mathbf{Q}[a{:}b] = \texttt{randHingTraj}\left(\texttt{motionConfig}, T\right)$ {see Definition C.6}
  end if
  $a \leftarrow b$
end for
$\tilde{\mathbf{Q}} = \mathbf{0}$ {timeseries of minimal coordinates $\tilde{\mathbf{Q}} \in \mathbb{R}^{T \times (N_q + 7N)}$ of system with IMU bodies; see Definition 3.2}
$\tilde{\boldsymbol{q}} \leftarrow \mathbf{0}$
$\dot{\tilde{\boldsymbol{q}}} \leftarrow \mathbf{0}$ {$\dot{\tilde{\boldsymbol{q}}} \in \mathbb{R}^{N_{\dot{q}} + 6N}$; see Definition 3.2}
for $t = 1$ **to** T **do**
  $\boldsymbol{\tau} \leftarrow \texttt{PDControl}\left(\texttt{sys}, \mathbf{Q}[:, t], \tilde{\boldsymbol{q}}[:N_q]\right)$ {see Definition C.7}
  $\boldsymbol{\tau} \leftarrow \texttt{concat}\left(\boldsymbol{\tau}, \mathbf{0} \in \mathbb{R}^{6N}\right)^\intercal$ {N IMUs' *passive* free joints}
  $\tilde{\boldsymbol{q}}, \dot{\tilde{\boldsymbol{q}}} \leftarrow \texttt{forDyn}\left(\texttt{sys}, \tilde{\boldsymbol{q}}, \dot{\tilde{\boldsymbol{q}}}, \boldsymbol{\tau}\right)$ {see Definition C.8}
  $\tilde{\mathbf{Q}}[t] = \tilde{\boldsymbol{q}}$
end for
$\mathbf{T} \leftarrow \texttt{forKin}\left(\texttt{sys}, \tilde{\mathbf{Q}}\right)$ {see Definition C.8}

Algorithm 4 `getXY` (RCMG Third Step)

Input: `sys`, $\boldsymbol{\lambda}_N$, $\mathbf{T} \in \left(\mathbb{H} \otimes \mathbb{R}^3\right)^{2N \times T}$
Output: $\mathbf{X} \in \mathbb{R}^{N \times 9 \times T}$, $\mathbf{Y} \in \mathbb{H}^{N \times T}$
$\mathbf{X} \leftarrow \mathbf{0}$
$\mathbf{Y} \leftarrow \mathbf{0}$
for $i = 1$ **to** N **do**
  $p \leftarrow \boldsymbol{\lambda}_N[i]$
  $\tilde{i} \leftarrow \texttt{sys.}\boldsymbol{n}[i]$
  $\tilde{p} \leftarrow 0$
  if $p \neq 0$ **then**
    $\tilde{p} \leftarrow \texttt{sys.}\boldsymbol{n}[p]$
  end if
  $O_i \leftarrow \texttt{isOuter}(i, \texttt{sys.}\boldsymbol{\lambda})$ {true if body i is an outer body; see Definition 3.1}
  if O_i **or** `randBernoulli`(0.33) {inner IMU data might not be made available} **then**
    $j \leftarrow \tilde{i} + N$ {body number IMU node}
    $\mathbf{X}[i, :6] = \texttt{simIMU}\left(\mathbf{T}[j], \texttt{sys.T}_\texttt{s}\right)$ {see Definition C.9}
  end if
  if $p \neq 0$ **and** `randBernoulli`(0.5) {for hinge joints, joint axis might not be made available} **then**
    $\mathbf{X}[i, 7:] = \texttt{sys.}\mathbf{J}[\tilde{i}]$
  end if
  ${}^{0}_{p}\mathbf{q} \leftarrow \mathbb{1}$
  if $\tilde{p} \neq 0$ **then**
    ${}^{0}_{p}\mathbf{q} \leftarrow \mathbf{T}[\tilde{p}, :4]$
  end if
  ${}^{0}_{i}\mathbf{q} \leftarrow \mathbf{T}[\tilde{i}, :4]$
  ${}^{i}_{p}\mathbf{q} \leftarrow {}^{0}_{p}\mathbf{q} \otimes {}^{0}_{i}\mathbf{q}^*$ {note that the expression ${}^{0}_{p}\mathbf{q}^* \otimes {}^{0}_{i}\mathbf{q}$ can not be used instead of ${}^{0}_{p}\mathbf{q} \otimes {}^{0}_{i}\mathbf{q}^*$, as it can dramatically reduce the network's ability to learn. The reason is that in the expression ${}^{0}_{p}\mathbf{q} \otimes {}^{0}_{i}\mathbf{q}^*$ the joint axis direction is expressed in the (more meaningful) local coordinate system and not in the base's coordinate system.}
  if $\tilde{p} = 0$ **then**
    ${}^{i}_{p}\mathbf{q} \leftarrow \texttt{zeroHead}\left({}^{i}_{p}\mathbf{q}\right)$ {see Definition B.7}
  end if
  $\mathbf{Y}[i] = {}^{i}_{p}\mathbf{q}$
end for

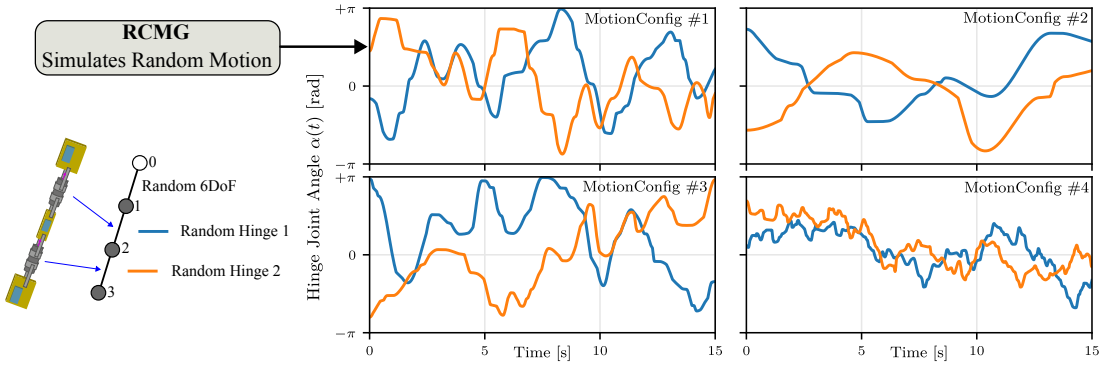


Figure 8: RCMG generates random motion by drawing random trajectories of the minimal coordinates of the system. Exemplary random trajectories of the hinge joint's minimal coordinates are shown for the four different `motionConfig`s (see Appendix C.1). They are drawn using the function `randHingeTraj`, see Definition C.6. RCMG also draws the random trajectory of the minimal coordinates of the free joint, but they are not shown here for simplicity.

C.1 The `motionConfig` Object

In the second step of the RCMG (see Section 4.2.2), random motion of the KC is simulated and, subsequently, training data is generated. Random motion of the KC is obtained by using PD control during a dynamical forward simulation such that the minimal coordinates of the KC track a randomly drawn set of reference trajectories for the minimal coordinates. The functions `randFreeTraj` and `randHingTraj` (see Definition C.6) are used to draw the minimal coordinates reference trajectories in Algorithm 3. Furthermore, the type of motion generated by these functions can be manipulated with a `motionConfig` object which defines various parameters, e.g., upper limits on angular velocities or lower limits on the amount of motion. For an exhaustive list of parameters, the reader is referred to the software implementation (see Appendix 4.5).

In this work, we use in total four different `motionConfig`s to generate training data. For each generated sequence, we randomly and uniformly draw from these four. The four `motionConfig`s are used to ensure that a wide range of different motion patterns are covered in the training data. Exemplary trajectories of the hinge joints' minimal coordinates are shown in Figure 8.

C.2 Support Functions used in Algorithms 2/3/4

In the following, the functions used in Algorithms 2/3/4 will be discussed. And while no pseudo-code is provided for these support functions, it should be noted that the majority of these functions should be understandable despite a textual description only. Additionally, the reader may always refer to the software implementation for additional details (see Appendix 4.5).

Definition C.1. The function `randBase`(`sys`, $\boldsymbol{\lambda}_N$) randomly re-attaches the base of the system `sys` and afterwards the nodes in the graph are re-numbered according to Section 3.3. This process yields a new parent array $\boldsymbol{\lambda}$. Additionally, the permutation of the new numbers of the nodes expressed in the numbering scheme that was used to obtain $\boldsymbol{\lambda}_N$ is captured in the numbering array $\boldsymbol{n}$.

Consider the three-segment KC given in Figure 2. The parent array is given by $\boldsymbol{\lambda}_3 = (0, 1, 2)^\intercal$ and we assume, without loss of generality, the numbering scheme that is shown in the figure. Here, the function `randBase` has three choices for attaching the base. The first is trivial and given by node 1. The second is given by node 2. In this case the parent array is always given by $\boldsymbol{\lambda} = (0, 1, 1)^\intercal$, and two scenarios for the numbering array are possible. They are $\boldsymbol{n} = (2, 1, 3)^\intercal$ and $\boldsymbol{n} = (3, 1, 2)^\intercal$. The third and last option is given by node 3. In this case the parent array is unchanged but $\boldsymbol{n} = (3, 2, 1)^\intercal$.

A second example is given in Figure 3.

Definition C.2. The functions `randSegmentToSegment` and `randSensorToSegment` randomize the body-to-body- and sensor-to-body positions expressed in the local coordinate, respectively. Internally, they both draw these vectors randomly from uniform values ranges. These value ranges are chosen such that there exists a dominant longitudinal direction (x-component), and two equal transversal directions.

Definition C.3. The function `randBernoulli`(p) returns 1 (or true) with probability p and zero (or false) else.

Definition C.4. The functions `getRigidStif` and `getRigidDamp` return the stiffness and damping parameters (both are six-dimensional) of the free joint that connects body i to IMU body $i + N$. This mechanism is used to simulate nonrigid IMU attachment. However, these functions provide a fixed set of values that leads to highly stiff and critically damped spring-damper system such that there is effectively no motion between body i and IMU body $i + N$.

Definition C.5. The functions `randRigidStif` and `randRigidDamp` return the stiffness and damping parameters of the free joint that connects body i to IMU body $i + N$. Internally, both functions draw these six-dimensional vector randomly from log-uniform values ranges.

Definition C.6. The function `randFreeTraj`(`motionConfig`, T) returns a random trajectory of minimal (for free joint minimal=maximal) coordinates $\in \left(\mathbb{H} \otimes \mathbb{R}^3\right)^T$ with T timesteps for a free 6-DoF joint. The function `randHingeTraj`(`motionConfig`, T) returns a random trajectory of minimal coordinates $\in \mathbb{R}^T$ with T timesteps for a hinge joint. Internally, they both use the `motionConfig` (see Appendix C.1) to constraint the random motion to physically relevant motion, i.e., the, e.g., angular velocity is bounded from above. Exemplary trajectories are shown in Figure 8. For additional details, the reader is referred to the software implementation, see Appendix 4.5.

Definition C.7. The function `PDControl`(`sys`, $\boldsymbol{q}_r, \boldsymbol{q}$) computes the generalized force vector $\boldsymbol{\tau} \in \mathbb{R}^{N_{\dot{q}}}$ using a decentralized scheme using N independent PD controllers, and where $\boldsymbol{q_r} \in \mathbb{R}^{N_q}$ denotes the reference minimal coordinates and $\boldsymbol{q} \in \mathbb{R}^{N_q}$ denotes the observed minimal coordinates. Note that the provided system object `sys` has $2N$ bodies but only the first N bodies are actuated. The remaining bodies are passive free joints.

Definition C.8. The function `forDyn`(`sys`, $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}$) applies forward dynamics in the system `sys` and integrates the minimal coordinates position and velocity vector $\boldsymbol{q}, \dot{\boldsymbol{q}}$ by $\texttt{sys.T}_\texttt{s}$. The function `forKin`(`sys`, $\mathbf{Q}$) applies forward kinematics in the system `sys`, i.e., it provides a map from minimal $\mathbf{Q}$ to maximal coordinates $\mathbf{T}$. Here, it is additionally vectorized over the time dimension. A text-book reference for both well-known algorithms can be found in Featherstone (2008).

Definition C.9. The function `simIMU`($\mathbf{T}, T_s$) simulates a 6D IMU from a trajectory of maximal coordinates. First, the maximal coordinates are butter-worth low-pass-filtered (both the quaternion and position trajectory). Then, a second-order numerical differentiation for both gyroscope and accelerometer is used. The accelerometer is low-pass-filtered. Gravity and simulated noise and bias terms are added. The cutoff frequencies have been optimized such that experimental IMU data is recovered with the highest fidelity from the maximal coordinate trajectories from OMC. Additional details can be found in Bachhuber et al. (2022).

D RING's Time Complexity and Computational Requirements at Inference

For practical applications, the inference time and computational requirements of RING are critical to enable real-world online applications. Therefore, we conduct a theoretical and empirical time complexity analysis of RING at inference.

The theoretical time complexity of RING depends on the operations involved when advancing the prediction by one timestep. First, we assume a naive matrix multiplication complexity, i.e, let $\mathbf{A} \in \mathbb{R}^{C \times D}$ and $\mathbf{B} \in \mathbb{R}^{D \times E}$ then $\mathbf{AB}$ is $\mathcal{O}(C \times D \times E)$. Now, recall that N is the number of bodies, F the number of features per body (here $F = 9$), M is the message dimension, and H is the hidden state dimension, then

- eq. (4) is $\mathcal{O}(N \times H \times H + N \times 1 \times H^2 + N \times M \times H)$ (f_θ),
- eq. (5) is $\mathcal{O}(N \times M)$ (summation operation, a tree with N nodes has at most $N - 1$ edges),

Table 3: Non-exhaustive list of IMU hardware and their typical (typ.) noise and bias properties as provided in the manufacturers' technical specifications. Unfortunately, not all manufacturers provide this information.

Hardware	Gyr. N.D. $[^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}}]$	Gyr. Of. $[^\circ\,\mathrm{s}^{-1}]$	Acc. N.D. $[\mu\mathrm{g}/\sqrt{\mathrm{Hz}}]$	Acc. Of. [mg]
Bosch BMI160 (1)	0.007	±3	180	±40
A.D. ADXL355 (2)	Acc. only	Acc. only	22.5	±25
Xsens MTi 10 (3)	0.03	±0.2	60	±5
Xsens MTi 100 (3)	0.01	±0.2	60	±5
Xsens MTw (ours) (4)	0.01	N.P.	200	N.P.
Movella Dot (5)	0.007	N.P.	120	N.P.
K. KXTC9-2050 (6)	Acc. only	Acc. only	125	N.P.
max (worst-case)	0.03	±3	200	±40

Noise Density (N.D.), Offset (Of.), Not Provided (N.P.)
Accelerometer units are micro-gravity per square-root of Hertz and milli-gravity
Sources in supplementary materials: `bosch_bmi160.pdf` (1), `analog_devices_adxl355.pdf` (2), `xsens_mti.pdf` (3), `xsens_mtw.pdf` (4), `movella_dot.pdf` (5), `kionix_kxtc9 − 2050.pdf` (6)

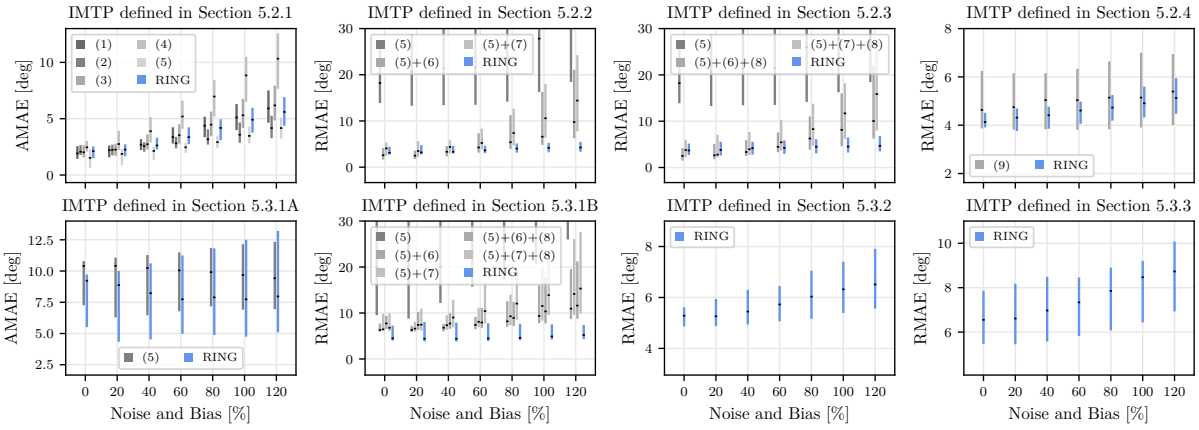


Figure 9: Comparison of the robustness of RING and SOTA methods as the level of noise and bias is increased. To simulate reasonable noise density and bias offset ranges, we constructed a worst-case IMU by combining the worst properties of various IMU manufacturers, as summarized in Table 3. Then, those worst-case noise and bias values are incrementally (in seven equidistant steps) increased from 0 to 120%, and a corresponding amount of simulated noise and bias is added to our real-world IMU data (see Section 5.1), which yields seven modified datasets. RING and all SOTA methods are validated on the modified dataset, and this procedure is repeated using ten different seeds. The AMAEs and RMAEs of all methods are plotted for all IMTPs as a function of the seven steps. The 25%/50%/75%-percentiles across all trials and seeds are shown, and they show that RING maintains accuracy comparable to SOTA methods in all IMTPs and, especially in two-segment KC tracking, substantially outperforms them. Methods are Weber et al. (2021)(1), Madgwick (2010)(2), Mahony et al. (2008)(3), Seel & Ruppin (2017)(4), Laidig & Seel (2023)(5), Laidig et al. (2017)(6), Lehmann et al. (2020)(7),Olsson et al. (2020)(8), Bachhuber et al. (2023)(9).

Table 4: Performance comparison across different hardware configurations.

Hardware	Computation Time [µs]				Latency [µs]			
	λ_1	λ_2	λ_3	λ_4	λ_1	λ_2	λ_3	λ_4
(1)	131 ± 8	757 ± 143	881 ± 128	916 ± 228	158 ± 7	789 ± 137	912 ± 84	975 ± 172
(2)	132 ± 5	172 ± 20	181 ± 17	192 ± 19	206 ± 5	241 ± 25	251 ± 31	267 ± 30
(3)	78 ± 5	127 ± 8	129 ± 8	127 ± 9	152 ± 14	199 ± 20	197 ± 11	201 ± 11
(4)	376 ± 90	681 ± 148	647 ± 96	879 ± 361	603 ± 167	834 ± 174	947 ± 211	1106 ± 377
(5)	214 ± 55	327 ± 69	325 ± 51	338 ± 60	494 ± 53	692 ± 133	684 ± 426	704 ± 153

Apple M2 Pro (1), W-1390P (2), W-1390P + RTX A5000 (3), Intel Xeon @ 2.0 Ghz, 1 Core, 2 Threads (Google Colab) (4), Intel Xeon @ 2.0 Ghz, 1 Core, 2 Threads + Tesla T4 GPU (Google Colab) (5)

- eq. (6) is $\mathcal{O}(N \times H \times (2M + F) + N \times H^2)$ (first GRU cell of g_θ), $\mathcal{O}(N \times H)$ (Layernorm), $\mathcal{O}(N \times H \times H + N \times H^2)$ (second GRU cell of g_θ),
- eq. (7) is $\mathcal{O}(N \times H)$ (Layernorm), $\mathcal{O}(N \times H \times H + N \times 1 \times H^2 + N \times 4 \times H)$ (h_θ),
- eq. (8) is $\mathcal{O}(N)$ (normalization).

This leads to an overall complexity of $\mathcal{O}\left(N \times H \times (H + M + F)\right)$. Note that the leading N term is implemented in a way such that it corresponds to an efficient batch operation and not a for loop. This is crucial for performance (especially on GPUs).

We conduct the empirical analysis for various types of hardware and report computation time and latency required for advancing the prediction by one timestep. Latency includes overheads such as conversion of the NumPy array to the deep-learning-framework-specific array type and potential to-and-from-device transfer overheads. Effectively, latency measures the time required from NumPy array input to NumPy array prediction, i.e.,

$$\underbrace{\underbrace{\underset{\texttt{np.ndarray}}{\mathbf{X}_t} \overset{\text{to device}}{\rightarrow} \underset{\texttt{jax.Array}}{\mathbf{X}_t} \underbrace{\rightarrow \texttt{ring} \rightarrow}_{\text{Computation Time}} \underset{\texttt{jax.Array}}{\hat{\mathbf{Y}}_t} \overset{\text{to host}}{\rightarrow} \underset{\texttt{np.ndarray}}{\hat{\mathbf{Y}}_t}}}_{\text{Latency}}.$$

Consequently, latency needs to be lower than the time delta due to the IMU sampling rate to enable lag-free real-time application (excluding a delay of one frame). Table 4 reports the timings for various hardware and the different IMTPs that include either one-, two-, three-, or four-segment KCs.

E Software Example

This example code uses the published software (see Section 4.5) and showcases how RING is applied in Section 5.3.2, i.e., it solves an IMTP that consists of a three-segment KC with sparse 6D IMU attachment and with unknown joint axes directions.

```python
import ring
import numpy as np

T  : int       = 30        # sequence length     [s]
Ts : float     = 0.01      # sampling interval   [s]
B  : int       = 1         # batch size
lam: list[int] = [-1, 0, 1] # parent array; because of Python's conventions body counting starts at 0, as a consequence the base body is indicated by -1 and not 0
N  : int       = len(lam)  # number of bodies
T_i: int       = int(T/Ts) # number of timesteps

X              = np.zeros((B, T_i, N, 9))
# where X is structured as follows:
```

```
# X[..., :3]   = acc
# X[..., 3:6]  = gyr
# X[..., 6:9]  = jointaxis

# let's assume we have an IMU on each outer segment of the
# three-segment kinematic chain
X[..., 0, :3]  = acc_segment1
X[..., 2, :3]  = acc_segment3
X[..., 0, 3:6] = gyr_segment1
X[..., 2, 3:6] = gyr_segment3

ringnet = ring.RING(lam, Ts)
yhat, _ = ringnet.apply(X)
# yhat: unit quaternions, shape = (B, T_i, N, 4)
```

Paper D

Dispelling Four Challenges in Inertial Motion Tracking with One Recurrent Inertial Graph-based Estimator (RING)

Authors: Bachhuber, Simon and Weygers, Ive and Seel, Thomas

Published in: IFAC Symposium on Biological and Medical Systems

Publication date: September 2024

DOI: https://doi.org/10.48550/arXiv.2409.02502

Dispelling Four Challenges in Inertial Motion Tracking with One Recurrent Inertial Graph-based Estimator (RING)

S. Bachhuber * **I. Weygers** * **T. Seel** **

* *Department Artificial Intelligence in Biomedical Engineering, FAU Erlangen-Nürnberg, 91052 Erlangen, Germany (e-mail: first.last@fau.de)*
** *Institute of Mechatronic Systems, Leibniz Universität Hannover, 30167 Hannover, Germany (e-mail: first.last@imes.uni-hannover.de)*

Abstract: In this paper, we extend the Recurrent Inertial Graph-based Estimator (RING), a novel neural-network-based solution for Inertial Motion Tracking (IMT), to generalize across a large range of sampling rates, and we demonstrate that it can overcome four real-world challenges: inhomogeneous magnetic fields, sensor-to-segment misalignment, sparse sensor setups, and nonrigid sensor attachment. RING can estimate the rotational state of a three-segment kinematic chain with double hinge joints from inertial data, and achieves an experimental mean-absolute-(tracking)-error of 8.10 ± 1.19 degrees if all four challenges are present simultaneously. The network is trained on simulated data yet evaluated on experimental data, highlighting its remarkable ability to zero-shot generalize from simulation to experiment. We conduct an ablation study to analyze the impact of each of the four challenges on RING's performance, we showcase its robustness to varying sampling rates, and we demonstrate that RING is capable of real-time operation. This research not only advances IMT technology by making it more accessible and versatile but also enhances its potential for new application domains including non-expert use of sparse IMT with nonrigid sensor attachments in unconstrained environments.

Keywords: Recurrent Neural Networks, Inertial Measurement Units, Orientation Estimation, Sparse Sensing, Magnetometer-free, Sensor-to-Segment Alignment

1. INTRODUCTION

Numerous recent developments in biomedical engineering applications require precise estimation of the motion of articulated bodies in space. Some prominent examples include unobtrusive human motion tracking outside the lab (García-de Villa et al., 2023a), and realizing intelligent symbiosis between humans and robots that enter immersive environments (Dafarra et al., 2024). Inertial Measurement Units (IMUs) are used in all these systems because of their unique ability to track movements of articulating rigid bodies of Kinematic Chains (KCs), in a cheaper and more reliable way than State-Of-The-Art (SOTA) multi-camera systems that require continuous line of sight.

All IMU-based motion tracking use cases heavily rely on Inertial Motion Tracking (IMT) algorithms that fuse different measurement signals to estimate motion parameters. This, however, is inherently limited by the following four key challenges (García-de Villa et al., 2023b):
(1) Inhomogeneous magnetic fields indoors and near ferromagnetic materials or electric devices;
(2) Sensor-to-segment alignment that involves determining joint positions and axis directions in local sensor coordinates;
(3) Solving sparse problems where some segments of the KC are not equipped with a sensor;
(4) Addressing real-world disturbances due to nonrigid sensor attachment and caused by large acceleration signals from impacts and soft tissue artifacts.

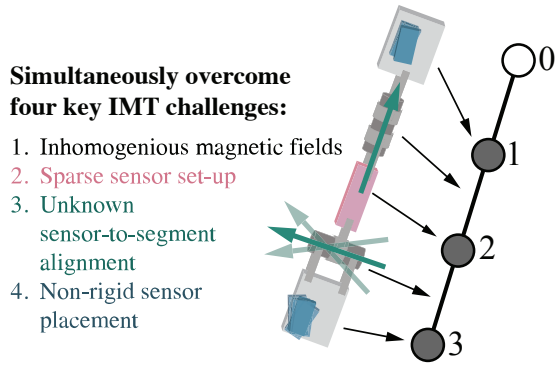


Fig. 1. A three-segment KC with two IMUs (blue boxes). The graph representation of the KC is given by the parent array $\boldsymbol{\lambda} = (0, 1, 2)^\intercal$. The neural network-based multiple-IMU sensor fusion algorithm RING receives the graph representation and IMU data as input and estimates the rotational state of the KC, overcoming all four key challenges of IMT simultaneously.

In recent years, many highly specialized methods have been proposed to address these challenges. We will provide

a brief overview of the latest and most notable developments accompanied by recent comprehensive methodological overviews. First, a multitude of different kinematic constraints are proposed to replace missing magnetometer heading information, as reviewed in Weygers et al. (2023). Furthermore, recent general-purpose (Caruso et al., 2021) magnetometer-free attitude estimators by Laidig and Seel (2023) achieved remarkable accuracy improvements in comparison with, e.g., the widely used filters from Madgwick (2010) and Mahony et al. (2008). Second, several algorithms have been developed to achieve automatic sensor-to-segment alignment, as outlined by Vitali and Perkins (2020) for specific joints with full sensor setups (Taetz et al., 2016; McGrath et al., 2018; Weygers et al., 2021). Third, a recent trend in sparse sensor setups is visible with methods that either use a limited number of sensors but include magnetometer measurements (Sy et al., 2020, 2021; Huang et al., 2018; von Marcard et al., 2017; Zheng et al., 2021) or are magnetometer-free (Grapentin et al., 2020; Bachhuber et al., 2023; Van Wouwe et al., 2023; Yi et al., 2021, 2022). Finally, the literature on IMT methods to overcome real-world disturbances is limited and focuses on late interception by outlier rejection techniques (Remmerswaal et al. (2021)) or further advances in connection constraints (García-de Villa et al., 2021).

Real-world IMT applications typically present multi-faceted challenges, requiring data-driven state observers like the Recurrent Neural Network-based Observer (RNNO) (Bachhuber et al. (2023)) that can effectively address the increasing complexity. To overcome a redundant implementation task in retraining RNNOs for every combination in a large grid of IMT challenges, we proposed the Recurrent Inertial Graph-based Estimator (RING) (Bachhuber et al. (2024)) as a pluripotent approach that solves IMT Problems (IMTPs) of tree-structured systems.

Despite RING's ability to provide a solution to a variety IMT challenges, its real-world applicability for a combination of all the aforementioned IMT challenges has not been investigated and their individual impact are unknown. Furthermore, while RING is aimed to be applicable in a plug-and-play fashion, it still requires a specific fixed sampling rate, which vastly limits its applicability in practice. Moreover, the real-time capability in inference has not been explored. In this work, we enhance and validate RING's usability with the following contributions:

(1) Extending RING's usability by enabling applicability to data from a broad range of sampling rates.
(2) Solve *for the first time* the four IMT challenges at once.
(3) Show zero-shot experimental generalizability in an extensive ablation study to gain insights on the performance of RING on individual IMT challenges.
(4) Analyze the real-time capability of RING.

2. PROBLEM FORMULATION

Consider a KC with three segments that are connected by hinge joints with arbitrary and unknown joint axes directions. Only the outer bodies are equipped with nonrigidly attached IMUs. A KC is a rigid-body system and it consists of multiple rigid objects (segments) that are rigidly attached to coordinate systems (bodies). In general, the topology of such a rigid-body system can be represented by a Connectivity Graph (CG) (Featherstone, 2008) where nodes represent bodies and edges represent degrees of freedom in the system. Here, for each segment there is one body with one segment attached to it, such that there is a one-to-one correspondence between segments and bodies. After the N bodies have been numbered, the CG can be encoded via a parent array $\boldsymbol{\lambda} \in \mathbb{N}^N$ where $\boldsymbol{\lambda}[i]$ is the body number of the parent of body i. For a three-segment KC, this graph representation and the one parent array utilized in this work is shown in Figure 1.

In this work, the goal is to estimate the complete rotational state of the KC up to a global heading offset (Weygers et al., 2023). We approach this through a filtering problem formulation, where an estimate of the complete rotational state $\boldsymbol{x}(t)$ is obtained at every time instant t from the current and all previous IMU measurements $\boldsymbol{y}(t' \leq t)$ that are combined into one measurement signal $\boldsymbol{y}(t) \in \mathbb{R}^{12}$ defined as

$$\boldsymbol{y}(t) = \left(\boldsymbol{\omega}_1(t)^\intercal, \boldsymbol{\rho}_1(t)^\intercal, \boldsymbol{\omega}_3(t)^\intercal, \boldsymbol{\rho}_3(t)^\intercal\right)^\intercal \quad \forall t \tag{1}$$

where $\boldsymbol{\omega}_i(t), \boldsymbol{\rho}_i(t)$ denote gyroscope and accelerometer measurements at time t of the IMU that is nonrigidly attached to body $\mathcal{S}_i \in \{1, 3\}$. The rotational state $\boldsymbol{x}(t) \in \mathbb{H}^3$ of the KC is straightforwardly defined by

$$\boldsymbol{x}(t) = \left({}_0^1\mathbf{q}(t)^\intercal, {}_1^2\mathbf{q}(t)^\intercal, {}_2^3\mathbf{q}(t)^\intercal\right)^\intercal \quad \forall t \tag{2}$$

where ${}_j^i\mathbf{q}(t)$ denotes the orientation from body $\mathcal{S}_i$ to body $\mathcal{S}_j$ at time t and where body 0 denotes the earth frame. Note that the ${}_0^1\mathbf{q}$ can only be estimated up to a heading offset from 6D measurements.

Real-world applicability requires solving all of the following challenges of the IMTP that is said to

- be **magnetometer-free** (or 6D in contrast to 9D) if the IMUs measure only three-dimensional angular rates and specific forces, and not provide magnetometer readings.
- require **sensor-to-segment alignment** when hinge joint axes directions are unknown.
- be **sparse** if not every segment that constitute the KC has an IMU attached. Here,the middle segment does not have an IMU attached. An IMTP that has an IMU attached to each body is said to have a full IMU setup.
- suffer from **motion artifacts** if the IMUs are not rigidly attached to the respective bodies, such that there can occur transnational and rotational motions between the segment and IMU. An IMTP without motion artifacts assumes that there cannot exist any relative motion between segment and IMU.

From this, it follows that the IMTP considered here is magnetometer-free, requires sensor-to-segment alignment, sparse, and suffers from motion artifacts.

3. METHODS

In this work, we extend the Neural Network-based (NN-based) multiple-IMU sensor fusion algorithm from Bachhuber et al. (2024). We address all four key IMT challenges outlined in Section 2, while enabling sampling rate robustness and showcase that RING is real-time capable. This is

achieved by both adapting the training procedure (Section: 3.1) and the NN-based multiple-IMU sensor fusion algorithm (Section: 3.2).

3.1 Simulated Training Data at Various Sampling Rates

RING (Bachhuber et al., 2024) is trained on large amounts of simulated input-output data at various sampling rates. The procedure that generates the data for the training of RING is called the Random Chain Motion Generator (RCMG) Bachhuber et al. (2024). It generates extensively augmented random motions of KCs with:

(1) different number of segments,
(2) randomized segment lengths,
(3) randomized IMU placement,
(4) randomized joint axes directions, and
(5) rigidly or nonrigidly attached IMUs (by simulating spring-damper-systems with randomized damping and stiffness parameters) (Bachhuber et al., 2024).

From these random KC motions we compute IMU and orientation measurements, but in this work at various sampling rates. These form the input-output pairs for training RING.

RCMG can be summarized as a function that only from PseudoRNG returns the training pair

- $\mathbf{X} \in \mathbb{R}^{T\times N\times 10}$, where $\mathbf{X}[:,i,:6]$ is the 6D IMU data for body i (if it is not dropped out), and where $\mathbf{X}[:,i,6{:}9]$ is the joint axis direction of the hinge joint between body i and its parent (if it is not dropped out, and if the parent is not the base), and where $\mathbf{X}[:,i,10]$ is the inverse sampling rate $\frac{1}{F}$, and
- $\mathbf{Y} \in \mathbb{H}^{T\times N}$ where $\mathbf{Y}[:,i]$ is the orientation from body i to its parent $\boldsymbol{\lambda}[i]$, and

where T is the number of timesteps, and N is the number of bodies in the KC (here $N = 3$).

To achieve a wide coverage, training data is generated for sampling rates drawn from

$$F \in \{40, 60, 80, 100, 120, 140, 160, 180, 200\}\text{Hz}$$

and to allow for efficient training data batching, the sequence duration is adjusted based on the sampling rate to achieve a common number of timesteps of $T = 6000$. A training batch is then built up by stacking 512 sequences, and additional details regarding the RCMG can be found in Bachhuber et al. (2024).

3.2 Neural Network Architecture: RING with Sampling Rate Input

We use a NN trained on data generated using the procedure described in Section 3.1. The network architecture is based on RING (Bachhuber et al., 2024), a powerful multiple-IMU sensor fusion algorithm that is composed of a decentralized network of message-passing Recurrent NNs (RNNs). Most notably, RING's parameters are defined on a per-node level and shared across all nodes in the graph. This design enables RING to be applied to broad range of IMTPs with a single set of parameters and enables its exceptional pluripotency.

In this work, the architecture of RING is extended to additionally accept a sampling rate input, such that the dimensionality of RING's per-timestep and per-node input increases by one. To summarize, RING can be viewed as the following step function that maps the previous state of RING $\boldsymbol{\xi}_{t\text{-}1} \in \mathbb{R}^{N\times 2H}$ and network input $\mathbf{X}_t \in \mathbb{R}^{N\times 10}$ at time t to the next RING state $\boldsymbol{\xi}_t$ and output $\hat{\mathbf{Y}}_t \in \mathbb{H}^N$, i.e.,

$$\boldsymbol{\xi}_t, \hat{\mathbf{Y}}_t = \texttt{ring}\left(\boldsymbol{\xi}_{t\text{-}1}, \mathbf{X}_t, \boldsymbol{\lambda}\right) \quad \forall t \tag{3}$$

where $H \in \mathbb{N}$ is the hidden state dimensionality, $\boldsymbol{\xi}_0 = \mathbf{0}$, and where the vector $\boldsymbol{\lambda}$ is the parent array, defined in Section 2. Internally, RING has the parameters of

- the Message-MLP-network $f_\theta : \mathbb{R}^H \to \mathbb{R}^M$, and
- the Stacked-GRUCell-network $g_\theta : \mathbb{R}^{2H} \times \mathbb{R}^{2M+10} \to \mathbb{R}^{2H}$ which consists of the sequence of Gated-Recurrent-Unit(GRU)Cell, LayerNorm, GRUCell (Cho et al., 2014), and
- the Quaternion-MLP that combines a Layernorm, and a MLP-network $h_\theta : \mathbb{R}^H \to \mathbb{R}^4$, and

where $M \in \mathbb{N}$ is the dimensionality of the messages that are passed along the edges of the graph. Note that the two GRUCells each have a hidden state dimensionality of H, thus the hidden state of RING is of dimensionality of $2H$. Then, equation (3) consists of several consecutive steps, for all N bodies:

(1) Messages $\mathbf{M}_t \in \mathbb{R}^{N\times M}$ are computed using f_θ.
(2) Messages are passed along the edges of the graph.
(3) The hidden state is updated using g_θ.
(4) The unnormalized output $\tilde{\mathbf{Y}} \in \mathbb{R}^{N\times 4}$ is computed using the Quaternion-MLP.
(5) The output is normalized to allow interpretation as unit quaternions. The final RING output is one unit quaternion per node $\hat{\mathbf{Y}}_t \in \mathbb{H}^N$.

RING is trained by comparing $\hat{\mathbf{Y}}_t$ to the ground truth $\mathbf{Y}_t$ which is provided by the RCMG, and by minimizing the mean-squared orientation error.

Additional details regarding the the RING architecture and its optimization strategy can be found and is exactly the same as in Bachhuber et al. (2024).

A software implementation of the RCMG and RING, and the source code for creating the results presented in Section 4.3 are hosted on GitHub [1] .

4. RESULTS AND DISCUSSION

In this section, we evaluate the accuracy of RING's orientation estimation, trained in simulation only, when applied to experimental data (see Section 4.1) of the problem specified in Section 2. The performance of RING is compared to (SOTA) methods (see Section 4.3).

It is remarkable, that RING can solve an experimental IMTP that combines all four challenges in IMT (magnetometer-free, unknown sensor-to-segment alignment, sparse sensor setup, and nonrigid sensor attachment) simultaneously, despite being trained on simulated data only.

[1] `https://github.com/SimiPixel/ring`

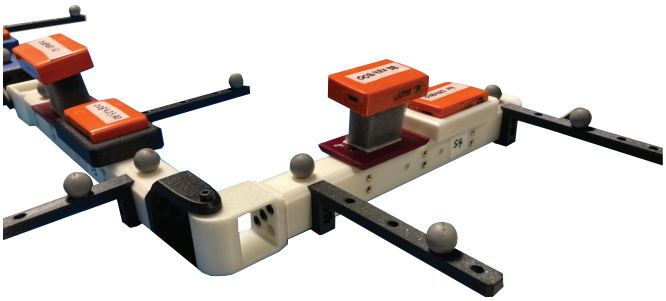


Fig. 2. Experimental 3D-printed KC used to validate the RING algorithm. To validate that RING overcomes the IMT challenge of non-rigid sensor placement, each segment of the KC has an IMU attached using foam padding. Additionally, a second IMU is rigidly attached to assess the impact of the foam padding on the accuracy of orientation estimates. Figure from Bachhuber et al. (2024).

4.1 Experimental Setup and Data Acquisition

We utilize a five-segment KC to record the experimental data but only the data for the given IMTP relevant parts of the KC are used for evaluation. The five-segment KC includes a singular spherical joint followed by three hinge joints, each oriented along the x, y, and z axes, respectively. Each segment of the KC was equipped with two IMUs: one rigidly attached to the segment and another nonrigidly attached using foam padding, as depicted in Figure 2.

Two distinct trials were conducted, involving random movements of the five-segment KC. Here, two three-segment KCs are thus extracted, one with joint axes direction x and y, and one with y and z. During evaluation, the first trial spans a duration of 66 s and features a diverse range of motions. Additionally, the second trial, with a length of 68 s, includes periods where the entire KC remained stationary. Overall, this results in four trials in total.

We refer to (Bachhuber et al., 2024) for additional details regarding the experimental setup and preprocessing.

Table 1. Experimental magnetometer-free orientation estimation accuracy (in degrees) of RING compared to two SOTA methods. The IMUs are nonrigidly attached and to counteract this influence both VQF and RNNO are used in combination with a low-pass filter with optimized cutoff-frequency. All methods are evaluated at 100 Hz.

Method	S. Misal.[1]	6D	Sparse	MAE [deg]
VQF[2]	✓	✗	✗	18.45 ± 9.10
RNNO[3]	✗	✓	✓	8.64 ± 4.13
RING	✓	✓	✓	8.10 ± 1.19

[1] Sensor-to-segment Misalignment
[2] Laidig and Seel (2023)
[3] Bachhuber et al. (2023)

4.2 Evaluation Metrics and Baselines

The ground truth orientations for the experimental trials (see Section 4.1) were recorded using optical motion capture. Orientation estimation accuracy is quantified using the Mean-Absolute-(tracking)-Error (MAE) in degrees. Here, the mean calculation reduces the dimensions of the different trials, time, and three orientations (including inclination and two relative orientations). In the time dimension, initial 5 s of each trial were deliberately excluded from the MAE calculations. This decision was made to ensure that the recorded errors accurately reflected the system's performance post-convergence.

To the best of the authors' knowledge, there exists no alternative method that can be applied to the IMTP as described in Section 2. However, two SOTA baseline methods can be identified after simplifying the IMTP so that it does not contain all four challenges simultaneously. The first baseline is obtained by using conventional IMT methods, that is, using a full 9D IMU setup and tracking each segment independently. The SOTA method for such single-IMU sensor fusion is VQF Laidig and Seel (2023). The second baseline is obtained after eliminating the challenge of anatomical calibration. Under the assumption of known joint axes direction, RNNO can be applied Bachhuber et al. (2023). Note that since two KCs with different directions of the joint axes are used for experimental validation (see Section 4.1), two trained RNNO networks are required. To compensate for the violation of the rigid-IMU-attachment assumption, both baselines additionally utilize a low-pass filter. The cutoff frequency was grid searched and we report only the best result for each baseline method.

4.3 Experimental Validation of RING

The trained RING is applied to experimental data from an IMTP that combines the four challenges of nonrigid IMU attachment, misaligned sensors and segments, magnetometer-free measurements, and a sparse sensor setup. The MAE in the orientation estimate for RING and the two SOTA baseline methods are reported in Table 1 and confirms that RING outperforms both alternative methods despite solving the more challenging IMTP. The first 15 seconds of one example sequence are shown in Figure 3 and demonstrate RING's prediction performance and quick convergence.

In Table 2, we conduct an ablation study to analyze the impact of nonrigid IMU attachment, sensor-to-segment alignment, and sparse IMU setup on RING's orientation estimation accuracy.

In Figure 4, the experimental data is resampled to a wide range of sampling rates to assess the robustness of RING w.r.t. the sampling rate. RING achieves a nearly constant orientation estimation accuracy which only, unsurprisingly, degrades slightly for low sampling rates.

4.4 Real-time Applicability of RING

By design, RING can be applied online as it is defined by a step function (see eq. (3)) that, based on the measurements at a certain timestep (see eq. (2)), returns an updated internal state and the rotational state estimate (see eq. 1) of the KC. Therefore, if the step function executes faster than the sampling rate requires, then it is said to be real-time capable. After compilation, the runtime of the step

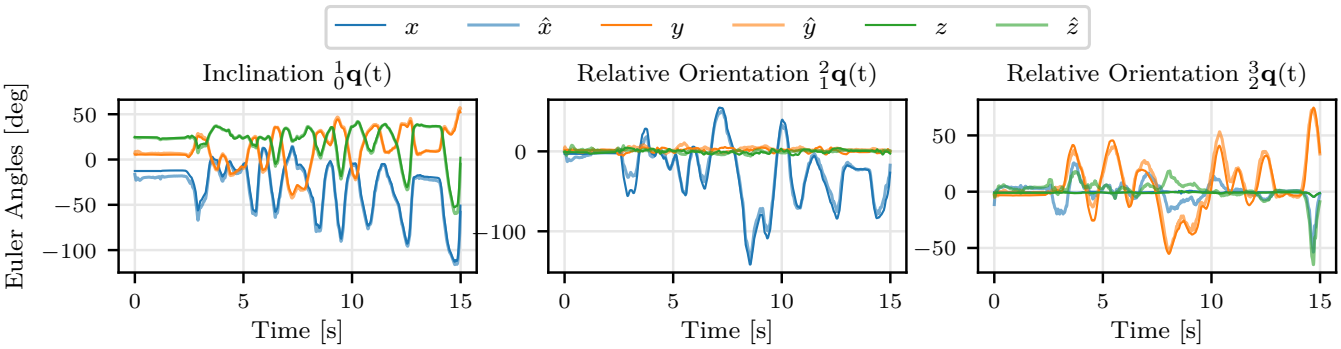


Fig. 3. Experimental example sequence that demonstrates RING's prediction performance from sparse, magnetometer-free, nonrigidly attached IMUs and without joint axes direction, and comparing to ground truth orientations for the first 15 s of one trial and for a double hinge joint KC with joint axes directions in $x-$ and $y-$direction.

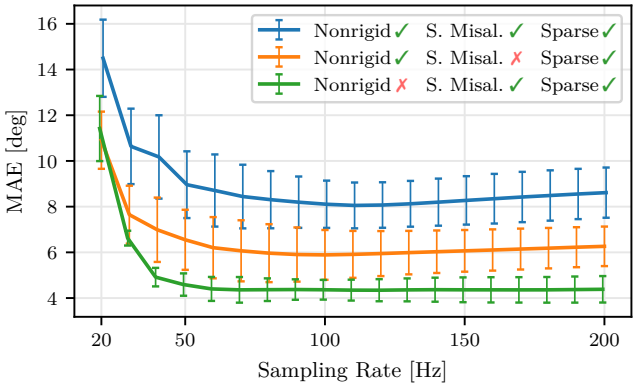


Fig. 4. Experimental magnetometer-free motion tracking accuracy (in degrees) of RING across various sampling rates. RING achieves a nearly constant estimation accuracy across sampling rates ranging from 50 to 200 Hz. Uncertainties are one standard deviation.

function of RING is (794 ± 16.7)µs on a M2 Macbook Pro. Thus, RING is real-time capable up to ≈ 1000 Hz.

5. CONCLUSION

In this work, we have extended RING, a powerful IMT method, to generalize across a wide range of sampling rates, and we have showcased that it can simultaneously overcome the four key challenges in IMT: inhomogeneous magnetic fields, sensor-to-segment misalignment, sparse sensor setups, and nonrigid sensor attachment. With an experimental tracking MAE of 8.10 ± 1.19 degrees if all four challenges are present simultaneously, RING accurately estimates the rotational state of a three-segment KC from IMU measurements. RING leverages a decentralized network of message-passing RNNs that is trained on simulated data but is capable of zero-shot generalization to real-world data. Our evaluations reveal RING's superiority over SOTA methods in terms of accuracy and applicability, additionally, we demonstrate RING's robustness across various sampling rates, and its real-time capability. By enabling plug-and-play usability and extending the applicability of inertial motion capture technology, RING not only advances the field but also opens new avenues for research and practical applications in environments previously deemed challenging.

Table 2. Ablation Study of the impact of individual IMT challenges on RING's experimental orientation estimation accuracy. In all scenarios, RING uses only magnetometer-free or 6D IMUs. Ablation study conducted at a sampling rate of 100 Hz.

Nonrigid	S. Misal.[1]	Sparse	MAE [deg]
✗	✗	✗	3.85 ± 0.25
✗	✗	✓	3.69 ± 0.22
✗	✓	✗	4.00 ± 0.17
✗	✓	✓	4.36 ± 0.49
✓	✗	✗	6.13 ± 0.57
✓	✗	✓	5.89 ± 1.25
✓	✓	✗	7.38 ± 0.50
✓	✓	✓	8.10 ± 1.19

[1] Sensor-to-segment Misalignment

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). The hardware is funded by the German Research Foundation (DFG).

REFERENCES

Bachhuber, S., Lehmann, D., Dorschky, E., Koelewijn, A.D., Seel, T., and Weygers, I. (2023). Plug-and-Play Sparse Inertial Motion Tracking With Sim-to-Real Transfer. *IEEE Sensors Letters*, 7(10), 1–4. doi:10.1109/LSENS.2023.3307122. Conference Name: IEEE Sensors Letters.

Bachhuber, S., Weygers, I., Lehmann, D., Dombrowski, M., and Seel, T. (2024). Recurrent Inertial Graph-based Estimator (RING): One Orientation Estimator to Track Them All. Under review for publication at Transactions on Machine Learning Research.

Caruso, M., Sabatini, A.M., Laidig, D., Seel, T., Knaflitz, M., Della Croce, U., and Cereatti, A. (2021). Analysis of the accuracy of ten algorithms for orientation estimation using inertial and magnetic sensing under optimal

conditions: One size does not fit all. *Sensors*, 21(7). doi:10.3390/s21072543.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. doi:10.48550/arXiv.1406.1078. URL http://arxiv.org/abs/1406.1078. ArXiv:1406.1078 [cs, stat].

Dafarra, S., Pattacini, U., Romualdi, G., Rapetti, L., Grieco, R., Darvish, K., Milani, G., Valli, E., Sorrentino, I., Viceconte, P.M., Scalzo, A., Traversaro, S., Sartore, C., Elobaid, M., Guedelha, N., Herron, C., Leonessa, A., Draicchio, F., Metta, G., Maggiali, M., and Pucci, D. (2024). icub3 avatar system: Enabling remote fully immersive embodiment of humanoid robots. *Science Robotics*, 9(86), eadh3834. doi:10.1126/scirobotics.adh3834.

Featherstone, R. (2008). *Rigid body dynamics algorithms*. Springer, New York. OCLC: ocn190774140.

García-de Villa, S., Casillas-Pérez, D., Jiménez-Martín, A., and García-Domínguez, J.J. (2023a). Inertial sensors for human motion analysis: A comprehensive review. *IEEE Transactions on Instrumentation and Measurement*, 72, 1–39. doi:10.1109/TIM.2023.3276528.

García-de Villa, S., Casillas-Pérez, D., Jiménez-Martín, A., and García-Domínguez, J.J. (2023b). Inertial sensors for human motion analysis: A comprehensive review. *IEEE Transactions on Instrumentation and Measurement*, 72, 1–39. doi:10.1109/TIM.2023.3276528.

García-de Villa, S., Jiménez-Martín, A., and García-Domínguez, J.J. (2021). Novel imu-based adaptive estimator of the center of rotation of joints for movement analysis. *IEEE Transactions on Instrumentation and Measurement*, 70, 1–11. doi:10.1109/TIM.2021.3073688.

Grapentin, A., Lehmann, D., Zhupa, A., and Seel, T. (2020). Sparse Magnetometer-Free Real-Time Inertial Hand Motion Tracking. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 94–100. doi:10.1109/MFI49285.2020.9235262.

Huang, Y., Kaufmann, M., Aksan, E., Black, M.J., Hilliges, O., and Pons-Moll, G. (2018). Deep inertial poser: learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics*, 37(6), 185:1–185:15. doi:10.1145/3272127.3275108.

Laidig, D. and Seel, T. (2023). VQF: Highly accurate IMU orientation estimation with bias estimation and magnetic disturbance rejection. *Information Fusion*, 91, 187–204. doi:10.1016/j.inffus.2022.10.014.

Madgwick, S.O.H. (2010). An efficient orientation filter for inertial and inertial/magnetic sensor arrays.

Mahony, R., Hamel, T., and Pflimlin, J.M. (2008). Nonlinear Complementary Filters on the Special Orthogonal Group. *IEEE Transactions on Automatic Control*, 53(5), 1203–1218. doi:10.1109/TAC.2008.923738. Conference Name: IEEE Transactions on Automatic Control.

McGrath, T., Fineman, R., and Stirling, L. (2018). An Auto-Calibrating Knee Flexion-Extension Axis Estimator Using Principal Component Analysis with Inertial Sensors. *Sensors*, 18(6), 1882. doi:10.3390/s18061882. URL https://www.mdpi.com/1424-8220/18/6/1882. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

Remmerswaal, E., Weygers, I., Smit, G., and Kok, M. (2021). Fast relative sensor orientation estimation in the presence of real-world disturbances. In *2021 European Control Conference (ECC)*, 411–416. doi:10.23919/ECC54610.2021.9654849.

Sy, L., Raitor, M., Rosario, M.D., Khamis, H., Kark, L., Lovell, N.H., and Redmond, S.J. (2021). Estimating Lower Limb Kinematics Using a Reduced Wearable Sensor Count. *IEEE transactions on bio-medical engineering*, 68(4), 1293–1304. doi:10.1109/TBME.2020.3026464.

Sy, L.W.F., Lovell, N.H., and Redmond, S.J. (2020). Estimating Lower Limb Kinematics Using a Lie Group Constrained Extended Kalman Filter with a Reduced Wearable IMU Count and Distance Measurements. *Sensors (Basel, Switzerland)*, 20(23), 6829. doi:10.3390/s20236829.

Taetz, B., Bleser, G., and Miezal, M. (2016). Towards Self-Calibrating Inertial Body Motion Capture. doi:10.48550/arXiv.1606.03754. ArXiv:1606.03754 [cs].

Van Wouwe, T., Lee, S., Falisse, A., Delp, S., and Liu, C.K. (2023). Diffusion Inertial Poser: Human Motion Reconstruction from Arbitrary Sparse IMU Configurations. ArXiv:2308.16682 [cs].

Vitali, R.V. and Perkins, N.C. (2020). Determining anatomical frames via inertial motion capture: A survey of methods. *Journal of Biomechanics*, 106, 109832. doi:https://doi.org/10.1016/j.jbiomech.2020.109832.

von Marcard, T., Rosenhahn, B., Black, M.J., and Pons-Moll, G. (2017). Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs. doi:10.48550/arXiv.1703.08014. ArXiv:1703.08014 [cs].

Weygers, I., Kok, M., Seel, T., Shah, D., Taylan, O., Scheys, L., Hallez, H., and Claeys, K. (2021). In-vitro validation of inertial-sensor-to-bone alignment. *Journal of Biomechanics*, 128, 110781. doi:https://doi.org/10.1016/j.jbiomech.2021.110781.

Weygers, I., Lehmann, D., Bachhuber, S., Laidig, D., and Seel, T. (2023). Do we still need magnetometers for inertial motion tracking? *Proceedings on Automation in Medical Engineering*, 2(1), 763–763. doi:10.18416/AUTOMED.2023.

Yi, X., Zhou, Y., Habermann, M., Shimada, S., Golyanik, V., Theobalt, C., and Xu, F. (2022). Physical Inertial Poser (PIP): Physics-aware Real-time Human Motion Tracking from Sparse Inertial Sensors. doi:10.48550/arXiv.2203.08528. ArXiv:2203.08528 [cs].

Yi, X., Zhou, Y., and Xu, F. (2021). TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors. ArXiv:2105.04605 [cs].

Zheng, Z., Ma, H., Yan, W., Liu, H., and Yang, Z. (2021). Training Data Selection and Optimal Sensor Placement for Deep-Learning-Based Sparse Inertial Sensor Human Posture Reconstruction. *Entropy*, 23(5), 588. doi:10.3390/e23050588. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

Paper E

Neural ODEs for Data-Driven Automatic Self-Design of Finite-Time Output Feedback Control for Unknown Nonlinear Dynamics

Authors: Bachhuber, Simon and Weygers, Ive and Seel, Thomas

Published in: IEEE Control System Letters (Volume 7)

Publication date: 07 July 2023

DOI: https://doi.org/10.1109/LCSYS.2023.3293277

Neural ODEs for Data-Driven Automatic Self-Design of Finite-Time Output Feedback Control for Unknown Nonlinear Dynamics

Simon Bachhuber, Ive Weygers, and Thomas Seel

***Abstract*—Many application fields, e.g., robotic surgery, autonomous piloting, and wearable robotics greatly benefit from advances in robotics and automation. A common task is to control an unknown nonlinear system such that its output tracks a desired reference signal for a finite duration of time. A learning control method that automatically and efficiently designs output feedback controllers for this task would greatly boost practicality over time-consuming and labour-intensive manual system identification and controller design methods. In this contribution we propose Automatic Neural Ordinary Differential Equation Control (ANODEC), a data-efficient automatic design of output feedback controllers for finite-time reference tracking in systems with unknown nonlinear dynamics. In a two-step approach, ANODEC first identifies a neural ODE model of the system dynamics from input-output data of the system dynamics and then exploits this data-driven model to learn a neural ODE feedback controller, while requiring no knowledge of the actual system state or its dimensionality. In-silico validation shows that ANODEC is able to —automatically— design competitive controllers that outperform two controller baselines, and achieves an on average ≈ 30 % / 17 % lower median RMSE. This is demonstrated in four different nonlinear systems using multiple, qualitatively different and even out-of-training-distribution reference signals.**

***Index Terms*—Autonomous systems, data-driven modeling, learning systems, motion control, neural networks.**

I. Introduction

ADVANCES in robotics and automation continue to have a significant impact on a large range of application fields: Cars that autonomously perform certain maneuvers, assembly lines with robotic manipulators that perform repetitive tasks, and patients that regain the ability to perform functional motions through combinations of functional electrical stimulation and wearable robotics.

A common task in all of these application fields is to control the output of a system with initially unknown nonlinear dynamics such that it follows a desired reference signal for a finite duration of time. Finding a feedback controller that solves this task is typically time-consuming and labour-intensive, since it requires modeling and/or system identification as well as controller design and adaptation. This greatly contrasts practical needs for methods and algorithms that exploit small amounts of input-output data from unknown nonlinear dynamics and autonomously design feedback controllers that enable accurate tracking of agile finite-time references (cf. Fig. 1).

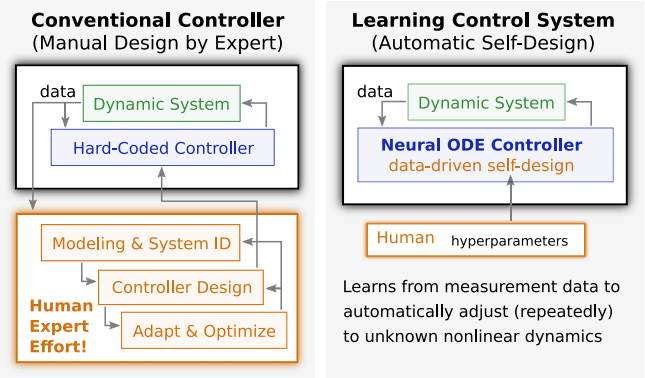


Fig. 1. Core concept of automatic self-design in learning control systems. Conventional controller design requires manual effort by human experts, which is expensive, time-consuming and does not scale. In contrast, the proposed ANODEC approach automatically designs competitive controllers from input-output data of unknown nonlinear system dynamics.

In the present contribution we provide an approach that solves this task and neither assumes knowledge of the system dynamics nor of the system state or its dimensionality. It automatically designs an output feedback controller that tracks *real-time* reference signals and does not require these references to be repetitive or known ahead of time (AOT).

Similar problems have been addressed by several previous approaches, most notably in the field of reinforcement learning (RL) [1], [2]. These methods, however, typically require full state knowledge, which severly limits applicability [3]. Within the field of RL, this limitation can be overcome using Partially Observable Markov Decision Processes (POMDP). RL methods tailored for POMDPs have been developed [4] and include, e.g., a recurrent policy function [5], stacking multiple observations, or, as a patch, a recurrent filter prior to a MDP-RL

Manuscript received 17 March 2023; revised 27 May 2023; accepted 21 June 2023. Date of publication 7 July 2023; date of current version 27 July 2023. Recommended by Senior Editor T. Oomen. *(Corresponding author: Simon Bachhuber.)*

The authors are with the Department Artificial Intelligence in Biomedical Engineering, FAU Erlangen-Nürnberg, 91052 Erlangen, Germany (e-mail: simon.bachhuber@fau.de).

Digital Object Identifier 10.1109/LCSYS.2023.3293277

algorithm [6]. Unfortunately, the applicability of these methods is limited by large data requirements [1], [7]. Moreover, while some RL methods have been used to effectively learn feedback control, they always assumed a single-objective trial-invariant state-dependent reward function which encodes information about a reference signal or target state that is usually known AOT [8].

More system-and-control-rooted approaches to similar tasks include optimal control (OC) and iterative learning control (ILC) methods and almost always require a known system model. OC approaches, including model-predictive control, typically optimize a state-dependent quadratic loss and thus assume full state knowledge [9]. Even recent data-driven methods still require knowledge of a nominal model of the dynamics and only learn the model mismatch from data [10]. ILC approaches, see, e.g., [11], [12], may not assume full state knowledge, but their design requires a system model or experimental tuning, and they assume a reference signal that is known AOT and typically trial-invariant.

In the present work we propose a solution based on *neural ordinary differential equations (ODEs)*, in which the right-hand-side of an ODE is parameterized by a neural network. This concept is enabled by numerical integrators being differentiable operations in the context of automatic differentiation and has recently gained traction [13]. Neural ODEs are not only used for modeling [13], [14] but also for representing controllers [15], [16]. In contrast to previous work on the latter, we assume to have no prior knowledge about the state or the dynamics of the system to be controlled.

Precisely, we propose Automatic Neural ODE Control (ANODEC) (cf. Fig. 1), a data-efficient automatic design of neural ODE feedback controllers for finite-time reference tracking in systems with unknown nonlinear dynamics. ANODEC consistently outperforms two common control approaches, and achieves an on average $\approx$ 30 % / 17 % lower median RMSE. This is demonstrated using multiple, qualitatively different and out-of-training-distribution reference signals in several double-pendulum dynamics and vehicle steering dynamics.

II. Problem Formulation

Assume there exists some *unknown* system dynamics Ψ that maps a time-varying, finite-time input signal $u(t) \in \mathbb{R}^p$ to a possibly noisy output vector $y(t) \in \mathbb{R}^q$, that is

$$y(t) = \Psi\big[u(t' < t)\big] \quad \forall t \in [0, T], \tag{1}$$

where $T \in \mathbb{R}$ is the finite trial duration. Here, Ψ can be thought of as any causal, time-invariant, potentially nonlinear, deterministic dynamical system, which includes, e.g., linear state-space models, nonlinear higher-order differential equations, and lifted-system dynamics. However, at least in the present work, the systems should be well-behaved in the sense that they exhibit none of the following very challenging phenomena: Finite escape times [17], ill-conditioned linearizations [18], or non-minimum phase characteristics [19].

We want to design a feedback controller that manipulates $u(t)$ to let $y(t)$ follow a given time-varying reference signal $r(t) \in \mathbb{R}^q$. Thus, we seek to find a controller dynamics Φ that maps $r(t)$ and $y(t)$ to the input vector $u(t)$, i.e.,

$$u(t) = \Phi\big[r(t' < t), y(t' < t), t\big] \quad \forall t \in [0, T], \tag{2}$$

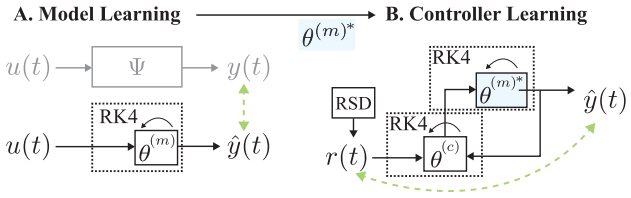


Fig. 2. Internal two-step approach underlying Automatic Neural ODE Control (ANODEC): (A) First, a neural ODE that models the unknown nonlinear dynamics of system Ψ is learned from input-output data (green arrow left side). (B) Then, a performant neural ODE controller is learned from thousands of forward simulations of the closed-loop system of the frozen model (blue highlighting) and the neural ODE controller. The optimization objective of the neural ODE controller is accurate reference tracking in the closed loop system (green arrow right side) subject to randomly drawn reference signals $r(t)$ from the Reference Signal Distribution (RSD).

such that it minimizes the tracking error between the output and the reference signal, i.e.,

$$\Phi^* = \arg\min_{\Phi} \int_0^T \|r(t) - y(t)\|_2 dt, \tag{3}$$

over the finite trial duration. Here, Φ can be used to express any causal, potentially nonlinear and time-variant, deterministic dynamical system, which includes, e.g., transfer functions, dynamic nonlinearities, and recurrent neural networks.

Synthesizing a feedback controller that only measures the output of the system dynamics Ψ, and not the state, is a particularly challenging problem, especially when the reference signal is only provided in real time and not ahead of time. Note that this problem formulation assumes no knowledge of the inner or physical state of the system dynamics Ψ or of its dimensionality. We do, however, assume that the system captured by Ψ is repeatedly reset to some trial-invariant initial state before every trial.

III. Automatic Neural ODE Control (ANODEC)

Here, we present the mathematical formulation of the proposed method ANODEC. Generally speaking, ANODEC designs controllers in a two-step approach of model learning and controller learning (cf. Fig. 2). However, the two steps are *internal* to the algorithm and don't require any intermediate human attention, which contrasts the typical manual approach of system identification and controller design. The content of this section is split up according to these two steps in Sections III-A and III-B, respectively.

A. Model Learning

The requirement of learning a feedback controller using the model, necessarily forces us to model causally. One-shot models that map a sequence to a sequence are not possible. While ordinary differential equations (ODEs) have a long history for modeling of physical (causal) phenomena, neural networks are a first choice for function approximation from data. We thus choose the combination of both, i.e., neural ODEs, for automatic data-driven model learning. Additionally, neural ODEs are differentiable, which will facilitate fast controller optimization in Section III-B.

We approximate Ψ in (1) by a neural ODE that is learned from experimental input-output data (u_i, y_i) (more on data requirements in Section IV-A). The state of the dynamical

system Ψ is not observed. The neural ODE that approximates Ψ is given by

$$\frac{d\xi^{(m)}(t)}{dt} = f_\theta^{(m)}\left(\xi^{(m)}(t), u(t)\right),$$
$$\hat{y}(t) = g_\theta^{(m)}\left(\xi^{(m)}(t)\right), \tag{4}$$

where $f_\theta^{(m)}$ and $g_\theta^{(m)}$ are feedforward neural networks, $u(t) \in \mathbb{R}^p$ the network input, $\hat{y}(t) \in \mathbb{R}^q$ the network output, and $\xi^{(m)}$ is the latent state vector of arbitrary dimension in which the neural ODE evolves. We denote the vector of all parameters of $f^{(m)}, g^{(m)}$ by $\theta^{(m)}$ and use supervised learning and training data of input-output pairs (u_i, y_i) to estimate and then optimize for these parameters

$$\theta^{(m)*} = \arg\min_{\theta^{(m)}} \mathbb{E}_{(u,y)}\left[\int_0^T \left\|y(t) - \hat{y}(t)\right\|_2 dt\right]. \tag{5}$$

B. Controller Learning

As a second step, we design a controller using the trained model (cf. Section III-A) with frozen parameters $\theta^{(m)*}$. In contrast to previous approaches [15], [16], we assume to have no measurements of the state of the dynamical system, i.e., we consider output feedback control. A performant controller, in the sense of Section II, is learned from a very large number of forward simulations of the trained model with different reference signals.

We represent the controller Φ as a neural ODE, given by

$$\frac{d\xi^{(c)}(t)}{dt} = f_\theta^{(c)}\left(\xi^{(c)}(t), r(t), y(t), t\right),$$
$$u(t) = g_\theta^{(c)}\left(\xi^{(c)}(t)\right), \tag{6}$$

where $f_\theta^{(c)}$ and $g_\theta^{(c)}$ are feedforward neural networks and $\xi^{(c)}$ is the latent state vector of arbitrary dimension in which the neural ODE evolves. We then close the loop between the neural ODEs that approximate the system dynamics Ψ and the controller Φ. The resulting closed-loop ODE takes the reference $r(t)$ as an input and outputs $\hat{y}(t)$. Ideally, this mapping should be identity, and thus we optimize the controller parameters $\theta^{(c)}$ via

$$\theta^{(c)*} = \arg\min_{\theta^{(c)}} \left(\lambda^{(c)}\left\|\theta^{(c)}\right\|_2 + \mathbb{E}_{r(t)\sim\mathcal{R}}\left[\int_0^T \left\|r(t) - \hat{y}(t)\right\|_2 dt\right]\right) \tag{7}$$

where $\lambda^{(c)}$ is a small regularization weight, and $\mathcal{R}$ is the reference signal distribution (RSD) (cf. Section IV-D). Note that ANODEC can easily be trained for a certain noise type and level by disturbing the model output, i.e., $\hat{y} \to \hat{y} + \epsilon$ in (7) where ϵ can be drawn from any distribution. Finally, note that deliberate choices in the neural network architecture of the model (cf. Section IV-B) allow to omit additional cost terms in (7) that are typically used to prevent model inversion. Such terms are not required here, as a similar effect is achieved by limiting the rate of change of the model output (cf. Section IV-B).

Algorithm 1 Functions for Drawing $u(t)$ to Probe the System

Precondition: ts is array of timesteps, seed is integer, scale is float
function DRAWGP(ts, seed, scale= 0.25)
  kernel ← 0.15 `squaredExpKernel`(length= 0.75)
  us ← `gaussianProcess`(kernel, ts, seed)
  us ← $\left(\frac{\text{us}-\texttt{mean(us)}}{\texttt{std(us)}}\right)\times$ scale
  return us
function DRAWCOS(ts, seed)
  $\omega \leftarrow 2\pi \times$ seed
  us ← $\cos(\omega \times \text{ts})\frac{\sqrt{\omega}}{2}$
  return us

IV. Algorithm

In this section we layout the required details to implement ANODEC. This includes the specific choices of feedforward neural networks f_θ, g_θ that parameterize (4) and (6) and how integration, expectation, and minimization operators of (5) and (7) are resolved.

A. Data Requirements

To apply ANODEC, a (training) dataset $\mathcal{D}$ that comprises N pairs of input-output data (each corresponding to one trial) must first be gathered from the system dynamics Ψ. The model learning (cf. Section III-A) then uses the dataset

$$\mathcal{D} = \{(u_i(t), y_i(t)) | i \in 1 \dots N\}. \tag{8}$$

Input trajectories $u(t)$ for gathering the data thus probing the system are drawn from a cosine frequency generator and a smooth Gaussian process with an overall dataset that is split with a ratio of 3:1. These functions are provided as pseudo-code in Algorithm 1. Between each trial the system resets to a trial-invariant initial state. Without loss of generality, we here make the choice of trials that last $T = 10$ s and are recorded at a sampling rate of 100 Hz. This leads to, e.g., an array representation of $u_i \in \mathbb{R}^{1000\times p}\ \forall i$.

Finally, for the remainder of this document we assume that in addition to the training dataset there exists a separate validation dataset that comprises three trials.

B. Neural Network Architecture

To parameterize the right-hand-side in (4), we use a fully-connected neural network $f_\theta^{(m)} : \mathbb{R}^{N^{(m)}\times p} \to \mathbb{R}^{N^{(m)}}$

$$\left(\xi^{(m)}(t)^\intercal, u(t)^\intercal\right)^\intercal \to \texttt{Linear} \to \texttt{Relu} \to \texttt{Linear} \to \texttt{Tanh} \to \frac{d\xi^{(m)}(t)}{dt} \quad \forall t \in [0, T], \tag{9}$$

with a width of $N^{(m)}$, a latent state vector $\xi^{(m)} \in \mathbb{R}^{N^{(m)}}$ and a layer size of $N^{(m)}$. Note the usage of `Tanh` in (9) to limit the rate of change of the latent state. Bounding the absolute rate of change from above in combination with a finite trial duration ensures that the model state and output will also stay within bounds. Similarly, in (4) $g_\theta^{(m)} : \mathbb{R}^{N^{(m)}} \to \mathbb{R}^q$ is parameterized as a single `Linear` layer, i.e., as a linear affine transformation. Note that the `Tanh` is used for system dynamics $f_\theta^{(m)}$ and not for system output $g_\theta^{(m)}$, and since the latent state $\xi^{(m)}$ can be arbitrarily scaled by $g_\theta^{(m)}$, limiting the rate of change of $\xi^{(m)}$ does not impose restrictions on the model output range.

To parameterize the controller, i.e., the right-hand-side in (6), we let $f_\theta^{(c)} : \mathbb{R}^{N^{(c)}\times q\times q} \to \mathbb{R}^{N^{(c)}}$ be a `Linear` layer and $g_\theta^{(c)} : \mathbb{R}^{N^{(c)}} \to \mathbb{R}^p$ with $\xi^{(c)}(t) \to$ `Linear` $\to$ `Tanh` $\to u(t) \quad \forall t \in [0, T]$. Together the two latent state dimensionalities of model $N^{(m)}$ and controller $N^{(c)}$ are the hyperparameters of ANODEC.

C. Time Integration

We use Runge-Kutta of fourth order (RK4) to obtain a numerical solution of the integration in (5) and (7) together with the initial condition $\xi^{(m)}(t=0) = 0$ and $\xi^{(m)}(t=0) = \xi^{(c)}(t=0) = 0$, respectively.

D. Expectation and Minimization Operators

For model training the training dataset $\mathcal{D}$ is split into four minibatches. In (5) the expectation operator $\mathop{\mathbb{E}}_{(u,y)}$ is estimated using a single minibatch. Gradients are then computed and an optimizer step is performed. After an epoch is finished (equaling four optimizer steps) the dataset is shuffled. For model training the minimization operation $\arg\min_{\theta^{(m)}}$ is performed using the Adam optimizer, a cosine-decaying learning rate schedule and norm-based and absolute value gradient clipping. The model is trained for (at most) 700 epochs and the validation dataset is used for early stopping.

The expectation operator $\mathop{\mathbb{E}}_{r(t)\sim\mathcal{R}}$ in (7) is estimated using a minibatch of six references where at every epoch thirty reference signals (containing five minibatches) are drawn randomly from the training RSD $\mathcal{R} = \texttt{steps}(0, 3)$ defined by

$$\texttt{steps}(a,b) = \left\{ r(t) = s r_0 \ \forall t \in [0,T] \;\middle|\; \begin{array}{l} r_0 \in \mathcal{U}([a,b]) \\ s \in \mathcal{U}(\{-1,1\}) \end{array} \right\} \tag{10}$$

where $a, b \in \mathbb{R}$ and $\mathcal{U}(\cdot)$ denotes a uniform distribution over a set. For controller training the minimization operation $\arg\min_{\theta^{(c)}}$ is performed using the same optimization strategy as it is used for model training with the addition of a small regularization weight $\lambda^{(c)} = 0.1$ in (7). The controller is trained for 1200 epochs.

E. Software Implementation

A software implementation of ANODEC and of the entire research project, including the routines to simulate the example systems used for validation (cf. Sections V-A, V-B), is available https://github.com/SimiPixel/chain_control.

The Python-based software uses JAX [20] and MuJoCo [21]. The ANODEC pipeline requires less than two minutes of compute time on a i7-1165G7. The trained neural ODE controller can be applied in real time at well above 100 Hz.

V. In-Silico Validation

In this section we validate ANODEC (cf. Sections III, IV) on four simulated complex nonlinear dynamical systems: (A) three different spring-damper double pendulum systems (DPs) (cf. Section V-A) and (B) a vehicle with Ackermann steering (AST) (cf. Section V-B). To showcase generalization capabilities of ANODEC in Section V-E, we propose several unseen reference signal distributions that, in Section V-F, will be used to validate ANODEC's performance compared to a system-specific optimal baseline (cf. Section V-D).

A. Double Pendulum (DP) System

A first dynamical system Ψ (cf. Section II) that is used for validation is a DP attached to a cart that moves horizontally on a slider using a single motor input. A connected DP with two consecutive hinge joint segment pairs is attached to the cart. Both hinge joints are damped with a factor $\gamma \in \mathbb{R}$, and are being pulled into a straight position by springs with stiffness $k \in \mathbb{R}$. Gravity would lead to less challenging dynamics, so it is disabled. The single input of the system is the cart's motor input moving the cart left-to-right on the slider. The single output is the left-to-right Cartesian coordinate of the end effector of the double pendulum, i.e., the end position of the second segment.

We investigate three parametrizations of this system corresponding to DP1/DP2/DP3, $\gamma = 6.0\times10^{-2}/\ 3.0\times10^{-2}/\ 1.5\times10^{-2}$, and $k = 6.0/3.0/1.5$, respectively. This results in notably different pendulum swinging characteristics. All three systems are simulated using MuJoCo [21], the training dataset (8) contains $N = 12$ trials, and the neural network widths of model and controller are chosen as $N^{(m)} = 50$ and $N^{(c)} = 15$.

B. Ackermann Steering (AST) System

To showcase applicability across different application systems, we also validate ANODEC for the system dynamics of a vehicle with Ackermann Steering. The vehicle with width $w_r = 1.5$ m and length $l_r = 4.0$ m drives forward with a constant velocity of $v_r = 3$ m s^{-1}. The single input of the system is the steering wheel angle ϕ_r in radians, clipped to a range $[-30.0°, 30.0°]$. The single output of the system is the Cartesian y-position of the vehicle $y_r \in \mathbb{R}$. The dynamics are given by $\frac{d\theta_r(t)}{dt} = \frac{v_r}{\frac{l_r}{\tan(\phi_r(t)+\epsilon)} - \frac{w_r}{2}}$, and $\frac{d}{dt}\begin{bmatrix} x_r(t) \\ y_r(t) \end{bmatrix} = v_r \begin{bmatrix} \cos(\theta_r(t)) \\ \sin(\theta_r(t)) \end{bmatrix}$, where $\theta_r \in \mathbb{R}$ is the angle between the cart's forward / length axis and the global x-axis, and $x_r \in \mathbb{R}$ is the Cartesian x-position of the vehicle. Initially, $x_r(t=0) = y_r(t=0) = \theta_r(t=0) = 0$, i.e., the vehicle is aligned with the global x-axis.

For this system the training dataset (8) contains $N = 24$ trials, and the neural network widths of model and controller are chosen as $N^{(m)} = 25$ and $N^{(c)} = 25$.

C. Noisy System Output

In all simulations we consider noisy output measurements by adding Gaussian white noise with a standard deviation of 0.02 m to the system output $y(t)$ in (1). Note that in Fig. 4 we merely plot the noise free system output, yet the controller still observes the noisy system output.

D. Baseline Controllers

We compare the performance of ANODEC to two baseline controllers: One controller that assumes the ability to perfectly tune a PI or PD controller, one controller that assumes that perfect model knowledge is available at least locally in the form of a linearization of the plant around its initial state.

1) Optimal-Tuning Baseline Controller: As a first baseline controller we determine, for each of the four systems, an optimal PI or PD controller (whichever performs better) by performing two exhaustive grid searches for both gains on the true system dynamics. At every grid point we evaluate

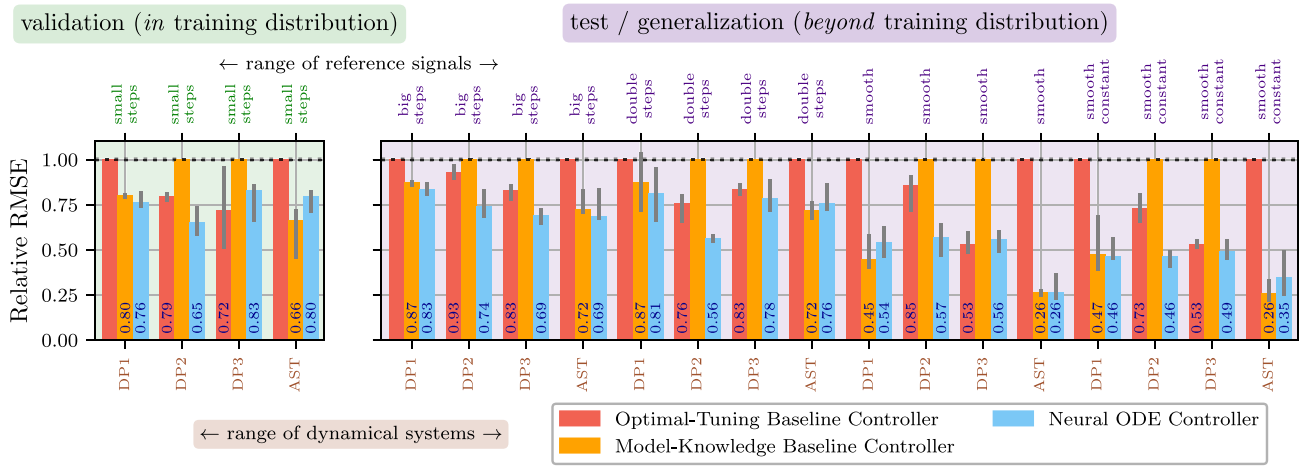


Fig. 3. Trained neural ODE controller's performance relative to the optimal-tuning controller baseline and model-knowledge controller baseline on all combinations of four systems and five reference signal distributions (RSDs). For each combination of system and RSD, the 25%/50%/75%−percentiles are obtained from 100 randomly drawn references from the RSD. The neural ODE controller achieves an on average ≈ 30 % / 17 % / 7 % / 40 % lower median RMSE across all validation and test combinations compared to the optimal-tuning controller baseline / model-knowledge controller baseline / the best out of the two baselines / the worst out of the two baselines.

the PI (or PD) controller's performance for 15 reference signals drawn from the training RSD (cf. Section IV-D) that are fixed across grid points. Note that this yields the best possible PI/PD controller for the given task and that performing such an exhaustive parameter optimization would require enormous amounts of interaction time in real-world systems.

2) Model-Knowledge Baseline Controller: As a second baseline controller we determine, for each of the four systems, a 5th-order transfer function controller by linearization of the nonlinear system around the initial state and pole placement of the closed loop. We find a suitable set of poles by grid search.

Note that these controllers constitute well-performing baselines. Of course better performing controllers could be obtained at the cost of increased tuning efforts or more restrictive assumptions such as perfect nonlinear model knowledge, but this is beyond the scope of this contribution.

E. Generalization to Tracking of Unseen Reference Signals

A common criticism of data-driven/machine-learning-based solutions is that they might only perform well in cases that are covered by the training+validation dataset. To verify that ANODEC generalizes beyond reference signals from the training+validation RSD, we test the performance on various additional, qualitatively different RSDs. We denote the training RSD $\texttt{steps}(0, 3)$ (cf. Section IV-D) by 'small steps', and we additionally define:

- 'big steps': $\texttt{steps}(4, 8)$, (cf. Equation (10))
- 'double steps': like 'small steps', but re-draws step amplitude mid trial at $t = 5$ s,
- 'smooth':

$$\left\{ r(t) = \Psi[\texttt{drawGP}(\text{seed})] \; \forall t \in [0, T] \; | \text{seed} \in \mathcal{U}(\mathbb{N}) \right\},$$

- 'smooth constant': like 'smooth', but the reference becomes a constant mid trial at $t = 5$ s.

Note that only references from 'smooth' RSD are feasible.

F. ANODEC Reference Tracking Performance

We validate the proposed method (cf. Sections III, IV) against the optimal-tuning and model-knowledge baseline controllers (cf. Section V-D) for each of the four systems (cf. Sections V-A, V-B) and five choices of RSD (cf. Section V-E). To this end we draw, for every combination of system and RSD, 100 signals from the corresponding RSD. After forward simulation of the 100 trials, the tracking RMSE (mean across time) of the neural ODE controller is divided by the tracking RMSE of the controller with the worst performance out of the three. Finally, the 25%/50%/75%−percentiles of this relative RMSE are determined across the 100 trials. We choose to visualize the relative RMSE since the absolute RMSE varies orders of magnitude across systems and RSDs.

Results are shown in Fig. 3 and we observe: Even on previously unseen reference signals, ANODEC outperforms both baseline controllers in 12 out of the 16 cases and achieves an on average ≈ 30 % / 17 % / 7 % / 40 % lower median RMSE across all validation and test combinations compared to the optimal-tuning controller baseline / model-knowledge controller baseline / (for each combination of system and RSD) best out of the two baselines / worst out of the two baselines.

Fig. 4 showcases the performance of the trained neural ODE controller compared to the optimal-tuning baseline controller (the better out of the two baselines for this system and RSD combination) for the example of the DP2 system with two beyond-training-distribution reference signals (drawn from two RSDs). The neural ODE controller not only significantly outperforms the optimal-tuning baseline controller in a sense of error norm. It also successfully dampens occurring oscillations, unlike the optimal-tuning baseline controller, and follows reference steps not only faster but in a more controlled way.

VI. Conclusion

In this contribution we have proposed Automatic Neural ODE Control (ANODEC), a fully automatic design of feedback control for finite-time output tracking in systems

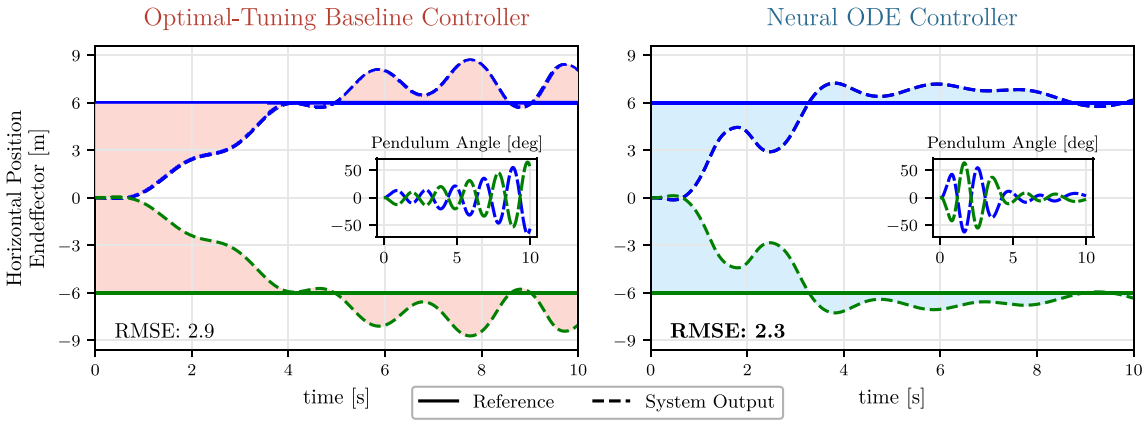


Fig. 4. Tracking performance of the optimal-tuning baseline controller (left) compared to the trained neural ODE controller (right) on two exemplary test reference signals that go *beyond* training distribution (drawn from 'big steps') in the double pendulum system (DP2). The system when controlled by the neural ODE controller reaches the reference point faster and with less oscillations. Video (https://youtu.be/tttkFFD81Qw) showcasing ANODEC on both reference signals in consecutive order (blue then green).

with unknown nonlinear dynamics. ANODEC is able to —automatically— design controllers that outperform two baselines, and achieves an on average $\approx$ 30 % / 17 % lower median RMSE compared to two common approaches that either obtain a baseline controller by optimal tuning or by assuming perfect linear model knowledge around the initial state. This has been demonstrated in four different nonlinear systems using multiple, qualitatively different and even out-of-training-distribution reference signals. ANODEC achieves this whilst requiring only two (or four) minutes of interaction data. ANODEC marks an important step towards data-efficient yet performant learning control solutions that generalize across different application systems and are easy-to-use.

Future work should aim to overcome the assumption of a trial-invariant initial state and include broad experimental evaluation of the method, also on high-dimensional multiple-input multiple-output systems with potentially strong nonlinearities. Furthermore, we aim to apply ANODEC in an iterative fashion allowing for automation of a minimal (yet sufficient) data retrieval, and provide more comparison of ANODEC to state-of-the-art data-driven methods. Finally, future work should involve a theoretical analysis of ANODEC to assess stability, guarantee safety, and quantify control performance trade-offs.

REFERENCES

[1] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. ICML*, Jan. 2011, pp. 465–472. [Online]. Available: https://dl.acm.org/doi/10.5555/3104482.3104541

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arxiv.abs/1707.06347*.

[3] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari. "Learning dynamical systems from partial observations." Feb. 2019. [Online]. Available: https://arxiv.org/abs/1902.11136

[4] M. Hausknecht and P. Stone. "Deep recurrent *Q*-learning for partially observable MDPs." 2015. [Online]. Available: https://arxiv.org/abs/1507.06527

[5] D. Wierstra, A. Forster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Logic J. IGPL*, vol. 18, no. 5, pp. 620–634, Oct. 2010.

[6] A. M. Schäfer, "Reinforcement learning with recurrent neural networks," Ph.D. dissertation, Inst. Inform., Univ. Osnabrück, Osnabrück, Germany, 2008.

[7] E. Schuitema, "Reinforcement learning on autonomous humanoid robots," Ph.D. dissertation, BioMechanical Eng., TU Delft, Delft, The Netherlands, 2012.

[8] E. Friedman and F. Fontaine. "Generalizing across multi-objective reward functions in deep reinforcement learning." Sep. 2018. [Online]. Available: https://arxiv.org/abs/1809.06364

[9] P. Bevanda, M. Beier, S. Heshmati-Alamdari, S. Sosnowski, and S. Hirche, "Towards data-driven LQR with koopmanizing flows," *IFAC PapersOnLine*, vol. 55, no. 15, pp. 13–18, Jan. 2022.

[10] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven MPC for quadrotors," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3769–3776, Apr. 2021.

[11] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 96–114, Jun. 2006.

[12] M. Meindl, D. Lehmann, and T. Seel, "Bridging reinforcement learning and iterative learning control," *Front. Robot. AI*, vol. 9, Jul. 2022, Art. no. 793512.

[13] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. NeurIPS*, vol. 31, 2018, pp. 6572–6583. [Online]. Available: https://dl.acm.org/doi/10.5555/3327757.3327764

[14] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *Proc. NeurIPS*, vol. 33, 2020, pp. 6696–6707.

[15] T. Asikis, L. Böttcher, and N. Antulov-Fantulin, "Neural ordinary differential equation control of dynamics on graphs," *Phys. Rev. Res.*, vol. 4, no. 1, Mar. 2022, Art. no. 13221.

[16] I. O. Sandoval, P. Petsagkourakis, and E. A. del Rio-Chanona. "Neural ODEs as feedback policies for nonlinear optimal control." 2022. [Online]. Available: https://arxiv.org/abs/2210.11245

[17] K.-K. Kim, "Stability analysis of Riccati differential equations related to affine diffusion processes," *J. Math. Anal. Appl.*, vol. 364, pp. 18–31, Apr. 2010.

[18] S. Skogestad, M. Morari, and J. Doyle, "Robust control of ill-conditioned plants: High-purity distillation," *IEEE Trans. Autom. Control*, vol. 33, no. 12, pp. 1092–1105, Dec. 1988.

[19] D. Ho and J. Karl Hedrick, "Control of nonlinear non-minimum phase systems with input–output linearization," in *Proc. ACC*, 2015, pp. 4016–4023.

[20] J. Bradbury et al. "JAX: Composable transformations of python+NumPy programs." 2018. [Online]. Available: https://github.com/google/jax/blob/main/CITATION.bib

[21] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE IROS*, 2012, pp. 5026–5033.

Paper F

A Soft Robotic System Automatically Learns Precise Agile Motions Without Model Information

Authors: John Doe and Homer "Doh" Simpson

Accpted at: 2024 IEEE/RSJ international conference on intelligent robots and systems (IROS)

Publication date: August 2024

DOI: https://doi.org/10.48550/arXiv.2408.03754

A Soft Robotic System Automatically Learns Precise Agile Motions Without Model Information

Simon Bachhuber[1†], Alexander Pawluchin[2†], Arka Pal[3], Ivo Boblan[2], Thomas Seel[4]

***Abstract*— Many application domains, e.g., in medicine and manufacturing, can greatly benefit from pneumatic Soft Robots (SRs). However, the accurate control of SRs has remained a significant challenge to date, mainly due to their nonlinear dynamics and viscoelastic material properties. Conventional control design methods often rely on either complex system modeling or time-intensive manual tuning, both of which require significant amounts of human expertise and thus limit their practicality. In recent works, the data-driven method, Automatic Neural ODE Control (ANODEC) has been successfully used to – fully automatically and utilizing only input-output data – design controllers for various nonlinear systems *in silico*, and without requiring prior model knowledge or extensive manual tuning. In this work, we successfully apply ANODEC to automatically learn to perform agile, non-repetitive reference tracking motion tasks in a real-world SR and within a finite time horizon. To the best of the authors' knowledge, ANODEC achieves, for the first time, performant control of a SR with hysteresis effects from only $30\,\mathrm{s}$ of input-output data and without any prior model knowledge. We show that for multiple, qualitatively different and even out-of-training-distribution reference signals, a single feedback controller designed by ANODEC outperforms a manually tuned PID baseline consistently. Overall, this contribution not only further strengthens the validity of ANODEC, but it marks an important step towards more practical, easy-to-use SRs that can automatically learn to perform agile motions from minimal experimental interaction time.**

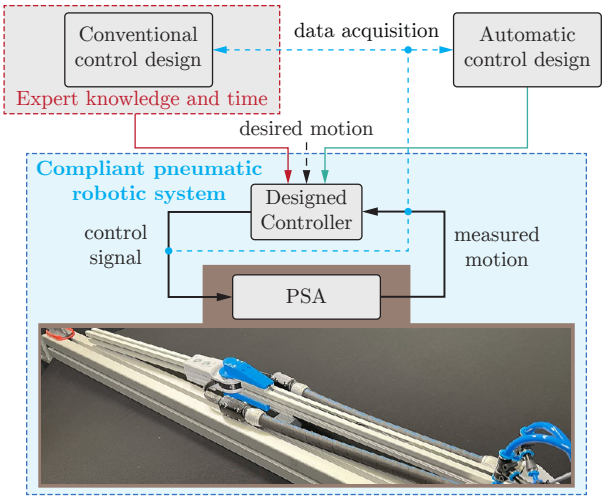


Fig. 1. Comparison between conventional controller design, reliant on human expertise, and an automatic, data-driven control structure for Pneumatic Soft Actuators (PSAs). Both controller design paradigms observe the PSA's control signal and resulting measured motion, and subsequently design position feedback controllers that enable the measured motion of the PSA to track the desired motion. In this work we show that the data-driven method, Automatic Neural ODE Control (ANODEC), enables automatic control design for a PSA.

I. Introduction

Soft Robots (SRs), including Pneumatic Soft Actuators (PSAs) and Pneumatic Artificial Muscles (PAMs), are gaining significant interest in diverse (bio)medical and industry application domains [1], [2]. Their rising popularity can be attributed to their inherent soft characteristics, derived from the use of flexible and deformable materials, which enables SRs to offer unique capabilities and adaptability in uncertain environments [3]. These attributes prove particularly advantageous in scenarios such as rehabilitation support [4] and human-robot interaction [5], where providing a safe interaction environment without external sensors and safety mechanisms is crucial.

Unfortunately, the very features that give PSAs their unique advantages also introduce several challenges for their accurate control, a dilemma that is yet to be fully addressed [3]. These challenges are for example: a) strong nonlinearities as the dynamics of PSAs are tightly coupled to the actuator's velocity which leads to lower control accuracy and overshoot, and b) creeping and hysteresis (cf. Figure 3, bottom subplot) due to the viscoelasticity and time-dependent properties of the soft material.

Despite these challenges, several methods have been previously applied for the control of PSAs. Model-based methods have been applied for PSAs such as, e.g., [6]–[9] but they typically require a detailed system model for accurate control and therefore involve extensive system modeling combined with human expertise.

Data-driven methods can alleviate the need for extensive system modeling and have been applied for PSAs. In [10], [11], Koopman operators are used to achieve reference tracking of a SR, however, they make restrictive assumptions such as either weak hysteresis characteristics or non-agile reference trajectory targets. In [12]–[14], a DNN-based MPC was employed to control SRs but they typically focus on non-agile motions and within a computationally intensive framework. In [15], an ODE-based kinematic model is learned and combined with MPC in order to learn to perform non-agile, quasistatic motions. In [16]–[18], Iterative Learning Control

†Both S.B. and Al.P. contributed equally to the manuscript.
[1]S.B. is part of the Department Artificial Intelligence in Biomedical Engineering, FAU Erlangen-Nürnberg, 91052 Erlangen, Germany
[2]Al.P. and I.B. are part of the CoRoLab Berlin, University of Applied Sciences and Technology, 13353 Berlin, Germany
[3]Ar.P. is part of the School of Mechanical Sciences, Indian Institute of Technology Bhubaneswar, 752050 Odisha, India
[4]T.S. is part of the Institute of Mechatronic Systems, Leibniz Universität Hannover, 30167 Hannover, Germany
Corresponding author: S.B. (simon.bachhuber@fau.de)

(ILC) in combination with different feedback controllers are used for the control of PSAs. However, ILC is tailored to repetitive motions and requires a series of iterations. Only in [18], deep learning is used to interpolate from previously learned trajectories and to achieve a broader range of motions, but they require large amounts of system interaction time to learn a single motion.

Recently, the data-driven method, Automatic Neural ODE Control (ANODEC) has been shown to – fully automatically – design feedback controllers for various nonlinear systems *in silico*, and without requiring prior model knowledge or extensive manual tuning. Unlike various other data-driven methods, e.g., [19]–[24], ANODEC requires neither the full state to be observed nor does it require an approximate system model.

In this contribution, we successfully apply ANODEC to automatically learn to perform agile, non-repetitive reference tracking motion tasks in a real-world PSA within a finite time horizon. To the best of the authors' knowledge, ANODEC achieves, for the first time, performant control of a PSA with hysteresis effects (cf. Figure 3) from only 30 s of input-output data and without any prior model knowledge. Additionally, the single feedback controller designed by ANODEC enables tracking of arbitrary reference signals and is validated on multiple, qualitatively different, and even out-of-training-distribution reference signals. Moreover, it is shown that ANODEC outperforms a manually tuned PID baseline, which similar to ANODEC, requires no prior model knowledge and does not require the system state to be observed.

Overall, it is shown that ANODEC allows for automatic control design (cf. Figure 1) and can enable PSAs to learn to perform agile motions from only a parsimonious amount of experimental interaction time and without requiring any prior model knowledge.

II. Pneumatic Soft Actuator

We consider a PSA, as shown in Fig. 2, that typically consists of a rigid or continuous kinematic structure and is driven by an antagonistic pair of PAMs. The muscles are attached via a belt and form a 1-Degree-of-Freedom (DOF) arm. The hinge joint angle represents the system output $\varphi \in \mathbb{R}$ and it is limited by the geometrical configuration of the muscle contraction and diameter of the pulley such that the range of motion $\varphi \in [\varphi_{\min}, \varphi_{\max}]$ where $\varphi_{\min/\max} \in \mathbb{R}$.

Each of the PAMs exhibits undesired parasitic hysteresis and creep properties that strongly influence the nonlinear characteristic force map $F_{1,2} \in \mathbb{R}$. Given the unknown forces F_1 and F_2, we instead control the pressure in the bellows. For this purpose, we use two model-based pressure controllers, which are designed to follow the given desired pressure $p_{\mathrm{d},1} \in \mathbb{R}$ and $p_{\mathrm{d},2} \in \mathbb{R}$.

Based on this system description, the system output is given by

$$\varphi(t) = \tilde{\Psi}\left(p_{\mathrm{d},1}(t' < t), p_{\mathrm{d},2}(t' < t)\right) \quad \forall t \in [0, T], \tag{1}$$

where $\tilde{\Psi}$ captures the system's dynamics (including the two pressure controllers), $T \in \mathbb{R}$ is the finite trial duration and

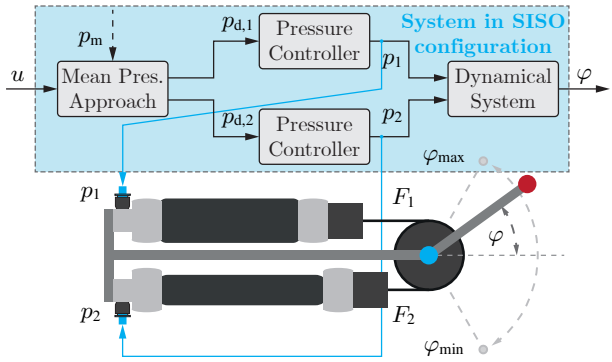


Fig. 2. The PSA including two PAMs is controlled using two pressure controllers, which implement a SISO control strategy through the mean and difference pressure approach. As a result, the single control input u controls both desired difference pressures $p_{\mathrm{d},1/2}$, and the single system output is the hinge joint angle φ.

where $p_{\mathrm{d},1/2}(t' < t)$ is used to denote that $\varphi(t)$ depends on all desired pressures up to time t, i.e., $p_{\mathrm{d},1/2}(t')\ \forall t' < t$.

Since the PSA has two pressure inputs, we couple them using the mean pressure approach in (2) to reduce the input dimension which then yields a Single-Input Single-Output (SISO) system. This is done by defining a mean pressure $p_{\mathrm{m}} = \text{const.} \in \mathbb{R}$ and a difference pressure $\Delta p_{\mathrm{d}} = p_{\mathrm{d},1} - p_{\mathrm{d},1}$, i.e.,

$$p_{\mathrm{d},1} = p_{\mathrm{m}} + 0.5u \quad \text{and} \quad p_{\mathrm{d},2} = p_{\mathrm{m}} - 0.5u. \tag{2}$$

The difference pressure becomes the system's single control input $u \in \mathbb{R}$, i.e., $u := \Delta p_{\mathrm{d}}$, which is limited by the maximum physical permissible pressure range $u_{\min} \in \mathbb{R}$ and $u_{\max} \in \mathbb{R}$. Finally, using (1) and (2), the system in SISO configuration is given by

$$\varphi(t) = \Psi\left(u(t' < t)\right) \quad \forall t \in [0, T]. \tag{3}$$

where the functional Ψ is called the system input-output map.

In this work, we aim to design a feedback controller that manipulates $u(t)$ to let $\varphi(t)$ follow a given time-varying reference signal $\varphi_{\mathrm{d}}(t) \in \mathbb{R}$. Thus, we seek to find a controller Φ that maps the desired angle $\varphi_{\mathrm{d}}(t)$ and the measured output $\varphi(t)$ to the input vector $u(t)$, that is,

$$u(t) = \Phi\left(\varphi_{\mathrm{d}}(t' < t), \varphi(t' < t)\right) \quad \forall t \in [0, T], \tag{4}$$

such that it minimizes the tracking error between the output and the reference signal, i.e.,

$$\Phi^* = \arg\min_{\Phi} \int_0^T \left\|\varphi_{\mathrm{d}}(t) - \varphi(t)\right\|_2 \mathrm{d}t \tag{5}$$

over the finite trial duration T. Note that the desired feedback controller measures the system output and not the state and that an arbitrary reference signal can be provided in real time and must not be provided ahead of time. However, we assume that the system captured by Ψ is repeatedly reset to some trial-invariant initial state before each trial, and that the open-loop system can be safely probed for input-output data. Note that the latter assumption is not only true for open-loop stable systems but can also be true for unstable system where

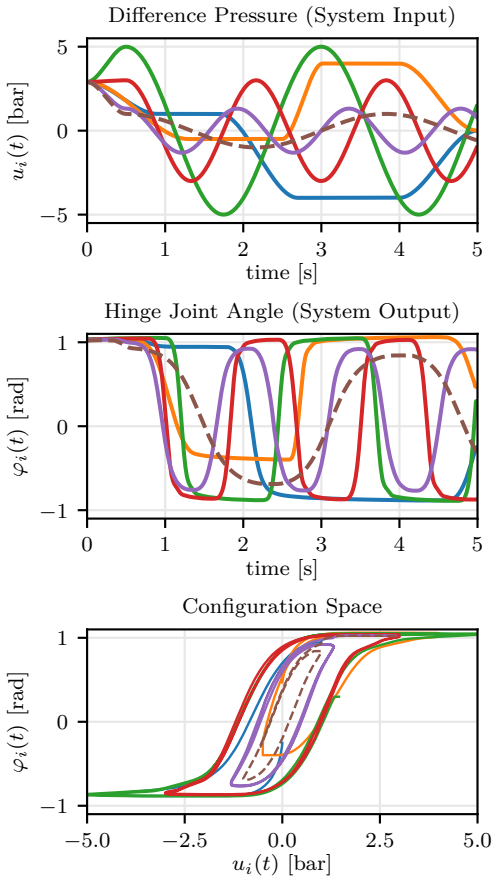


Fig. 3. Training data for ANODEC consists of five input-output pairs, each of a five seconds length, that are gathered from the experimental PSA. One additional input-output pair (dashed line) is collected and used as validation data and to prevent model overfitting. The feasible input and output intervals of the experimental PSA are $u(t) \in [-6, 6]$ bar and $\varphi(t) \in [-1, 1]$ rad, respectively.

the instability acts on large timescales relative to the finite trial duration, i.e., it is sufficiently slow.

III. Automatic Neural ODE Control (ANODEC)

Internally, the ANODEC algorithm [25] designs controllers in a two-step approach of model learning from input-output data pairs, and controller learning that utilizes the learned model.

A. Data Requirements

To apply ANODEC, a dataset $\mathcal{D}$ that comprises of six pairs of input-output data over time, each corresponding to one trial of length $T = 5\,\mathrm{s}$, is gathered from the system input-output map Ψ. The data is recorded at 100 Hz. Fig. 3 shows the six input-output data pairs.

The model learning (cf. Section III-B) then uses the dataset

$$\mathcal{D} = \{(u_i(t), \varphi_i(t)) | i \in 1 \dots 6\}. \tag{6}$$

The sixth pair of input-output data is split from the dataset $\mathcal{D}$ and used as validation data. The validation data is used to ensure that the learned model is accurate and that the amount of collected data is sufficient for an accurate model. Thus, in total ANODEC requires 30 s of interaction time with the PSA. Input trajectories $u(t)$ for gathering the data and thus probing the system are drawn from a sinusoidal and a spline function generator. These functions are provided as pseudocode in Algorithm 1. Before each trial starts, the system enters a saturation state such that $\varphi(t = 0) \approx \varphi_{\max}$. This procedure is required as otherwise there might be state ambiguity due to the hysteresis of the PAMs. Note that the system probing and model learning step (cf. Section III-B) can easily be extended to an iterative process that switches between system probing and model learning until a sufficiently accurate model is obtained. This way ANODEC can be extended to a fully autonomous control design algorithm that iteratively improves the performance of the designed feedback controller.

B. Model Learning

ANODEC approximates Ψ in (3) by a neural ODE that is learned from pairs of experimental input-output trajectories $u_i(t), \varphi_i(t)$ (cf. Section III-A). The neural ODE that approximates Ψ is given by

$$\begin{aligned} \frac{\mathrm{d}\xi^{(m)}(t)}{\mathrm{d}t} &= \tanh\left(A_1^{(m)}\left(\xi^{(m)\intercal}(t),\, u(t)\right)^\intercal + b_1^{(m)}\right), \\ \hat{\varphi}(t) &= A_2^{(m)}\xi^{(m)}(t) + b_2^{(m)}, \end{aligned} \tag{7}$$

where the latent state vector $\xi^{(m)} \in \mathbb{R}^9$, and with model parameters $A_1^{(m)} \in \mathbb{R}^{9\times 10}$, $b_1^{(m)} \in \mathbb{R}^9$, $A_2^{(m)} \in \mathbb{R}^{1\times 9}$, $b_2^{(m)} \in \mathbb{R}$. We denote the vector of all model parameters by $\theta^{(m)} \in \mathbb{R}^{109}$ and use supervised learning to estimate and then optimize for these parameters

$$\theta^{(m)^*} = \arg\min_{\theta^{(m)}} \sum_{i=1}^{5} \int_0^T \|\varphi_i(t) - \hat{\varphi}_i(t)\|_2 \,\mathrm{d}t. \tag{8}$$

C. Controller Learning

As a second step, ANODEC designs a feedback controller using the trained model with frozen parameters $\theta^{(m)^*}$. Internally, it performs a very large number of forward simulations of the closed-loop dynamics (of trained model and feedback controller) with randomly drawn step reference signals.

ANODEC represents the controller Φ as a neural ODE, given by

$$\begin{aligned} \frac{\mathrm{d}\xi^{(c)}(t)}{\mathrm{d}t} &= A_1^{(c)}\left(\xi^{(c)\intercal}(t),\, \varphi(t),\, \varphi_{\mathrm{d}}(t)\right)^\intercal + b_1^{(c)}, \\ \bar{u}(t) &= \tanh\left(A_2^{(c)}\xi^{(c)}(t) + b_2^{(c)}\right), \\ u(t) &= (u_{\max} - u_{\min})\,(\bar{u}(t)\,0.5 + 0.5) + u_{\min}, \end{aligned} \tag{9}$$

where the latent state vector $\xi^{(c)} \in \mathbb{R}^5$, and with controller parameters $A_1^{(c)} \in \mathbb{R}^{5\times 7}$, $b_1^{(c)} \in \mathbb{R}^5$, $A_2^{(c)} \in \mathbb{R}^{1\times 5}$, $b_2^{(c)} \in \mathbb{R}$. We denote the vector of controller parameters by $\theta^{(c)} \in \mathbb{R}^{46}$.

ANODEC then closes the loop between the neural ODEs that approximate the system input-output map Ψ (7) and the controller Φ (9). The resulting closed-loop ODE takes

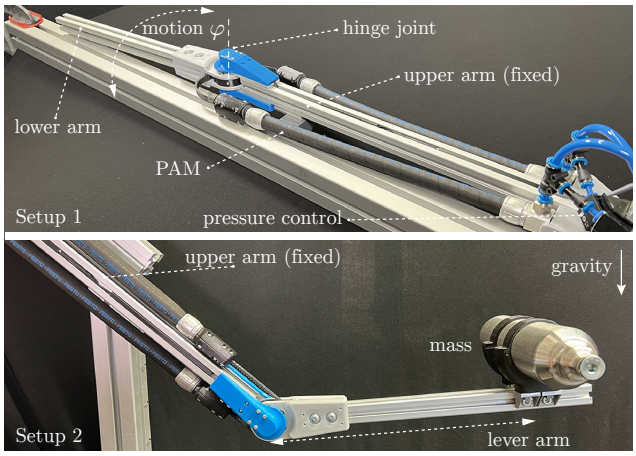


Fig. 4. Two experimental setups of a pneumatic arm with a single DOF. In both setups, two PAMs (pneumatic artificial muscles, black tubes) are used as an antagonistic pair to control the arm's forces and position. The upper setup shows the simplest configuration without the influence of gravity and external load. The lower setup is loaded with an external weight of $0.6\,\mathrm{kg}$, with a lever arm of $0.25\,\mathrm{m}$ oriented against gravity. Both arms are fixed to the ground to prevent undesired movements.

the reference $\varphi_\mathrm{d}(t)$ as an input and outputs $\hat{\varphi}(t)$, and we optimize the controller parameters $\theta^{(c)}$ via

$$\theta^{(c)^*} = \arg\min_{\theta^{(c)}} \left(\lambda^{(c)} \left\| \theta^{(c)} \right\|_2 + \mathop{\mathbb{E}}_{\varphi_\mathrm{d} \sim \mathcal{U}(\varphi_\mathrm{min}, \varphi_\mathrm{max})} \left[\int_0^T \left\| \varphi_\mathrm{d} - \hat{\varphi}(t) \right\|_2 \mathrm{d}t \right] \right) \quad (10)$$

where $\lambda^{(c)} = 4 \cdot 10^{-4}$ is a small regularization weight, and the reference φ_d is a constant value drawn uniformly from $[\varphi_\mathrm{min}, \varphi_\mathrm{max}]$.

D. Time Integration & Optimization

We use Runge-Kutta of fourth order (RK4) to obtain a numerical solution of the time integration in (8) and (10), together with the initial condition $\xi^{(m)}(t=0) = 0$ and $\xi^{(m)}(t=0) = \xi^{(c)}(t=0) = 0$, respectively.

For model training, the five input-output pairs from the training dataset $\mathcal{D}$ are combined into a single model training batch. For controller training, the expectation operator $\mathop{\mathbb{E}}_{\varphi_\mathrm{d} \sim \mathcal{U}(\varphi_\mathrm{min}, \varphi_\mathrm{max})}$ is estimated by randomly drawing 50 constant reference values at every optimizer step.

Both (8) and (10) are optimized using the Adam optimizer with a learning rate of 1×10^{-3} and global gradient L2 norm clipping of 1.0. Gradients are computed using backpropagation. The model is trained for $50\,000$ optimizer steps and the single validation input-output pair is used for early stopping and thus prevent overfitting during the model learning. The controller is trained for $12\,000$ optimizer steps.

IV. Experimental Validation

In this section, we validate ANODEC (cf. Section III) on an experimental PSA (cf. Section II). To showcase the generalization capabilities of ANODEC in Section IV-C, we propose unseen reference signal distributions that, in Section IV-D, will be used to validate ANODEC's performance compared to a PID controller baseline (cf. Section IV-B). The corresponding demonstration video to the presented results in Fig. 5 can be found here.

A. Experimental setup

We evaluate ANODEC's performance on a pneumatic compliant robotic arm which should solve a pick and place motion task on a given reference trajectory. The arm, as depicted in Fig. 4, consists of both an upper and a lower arm. The upper arm carries the PAMs responsible for controlling the motion of the lower arm. The hinge joint has a range of motion within $\varphi \in [-1, 1]\,\mathrm{rad}$ and connects both PAMs (DMSP-10-250N) via a constant pulley. The angle is measured using a MLX90316 rotary position sensor. The upper setup (Setup 1) illustrates the configuration without external influences such as gravity and load. The lower setup (Setup 2) shows the arm loaded with a $0.6\,\mathrm{kg}$ weight on a $0.25\,\mathrm{m}$ lever arm against gravity. Due to the non-negligible mass of the lower arm and the highly agile target motions, the upper arm is securely fixed to the ground to prevent undesired movements.

To generate forces, the PAMs are driven using a model-based pressure controllers and two proportional directional control valves MPYE-5-M5-010-B for controlling the mass flow. The controlled pressure is measured using two SPTE-P10R-S4-B-2.5K pressure transmitters. The supply pressure was set to $8\,\mathrm{bar}$. This results in a feasible system input range $u \in [-6, 6]\,\mathrm{bar}$.

The pressure controllers runs on an embedded computer with a sampling rate of $200\,\mathrm{Hz}$. This system is connected through Ethernet and ROS to a computer (Intel i7 7700) that controls both ANODEC and PID approaches in Python with a control rate of $100\,\mathrm{Hz}$.

B. Baseline Controller

As a baseline, we compare the performance of ANODEC to a manually tuned PID controller, which similar to ANODEC, requires no prior model knowledge and does not require the system state to be observed. Eleven trials were required (equaling $55\,\mathrm{s}$ of interaction time) for manual tuning and to achieve a sufficiently performant PID controller. Since we only control the pressure and have no assumptions regarding the force-pressure relation, an integral part is needed to compensate for the PAMs force nonlinearity and hysteresis.

The tuned gain parameters are $k_\mathrm{p} = 2, k_\mathrm{i} = 30$ and since the derivative term of the PID controller is highly sensitive to the sensor noise from the angle sensor, leading rapidly to undesired instabilities, the D term is set to zero. The PID controller's output u was clipped to stay within the feasible system input range of $[u_\mathrm{min}, u_\mathrm{max}]$. This is not required for ANODEC since its output remains within the feasible input range by design, see (9).

C. Generalization to Tracking of Unseen Reference Signals

A common criticism of data-driven or machine-learning-based solutions is that they might only perform well in

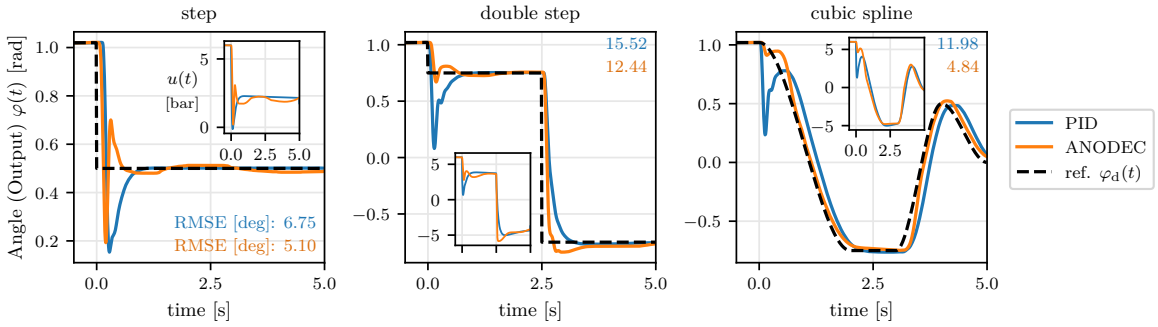


Fig. 5. Performance comparison of the automatic ANODEC and a manually tuned PID controller baseline in Setup 1 for one exemplary reference signals drawn from the three reference signal distributions: Step, double step, and smooth. Even on reference signals that were not present during training (double step, smooth), ANODEC is able to consistently outperform the PID baseline and achieves a lower RMSE tracking error while requiring less experimental interaction time. Video (`https://youtu.be/7HkXKy0WuRw`) that showcases these trials.

TABLE I

REFERENCE TRACKING RMSE (IN DEGREES) OF PID AND ANODEC FOR THREE REFERENCE SIGNAL DISTRIBUTIONS AND TWO SYSTEMS: STEPS, DOUBLE STEPS, AND CUBIC SPLINES. FOR EACH DISTRIBUTION, N DISTINCT REFERENCE SIGNALS ARE DRAWN AND USED TO ESTIMATE THE MEAN-RMSE AND ONE STANDARD DEVIATION.

REFERENCES	PID	ANODEC
SETUP 1		
STEPS ($N = 2$)	12.89 ± 6.14	10.03 ± 4.93
DOUBLE STEPS ($N = 2$)	12.86 ± 2.66	11.08 ± 1.37
CUBIC SPLINES ($N = 12$)	10.01 ± 1.44	4.48 ± 1.11
SETUP 2		
STEPS ($N = 2$)	4.00 ± 0.45	5.54 ± 0.53
DOUBLE STEPS ($N = 2$)	9.50 ± 4.03	6.81 ± 0.75
CUBIC SPLINES ($N = 4$)	4.56 ± 0.81	5.01 ± 0.79

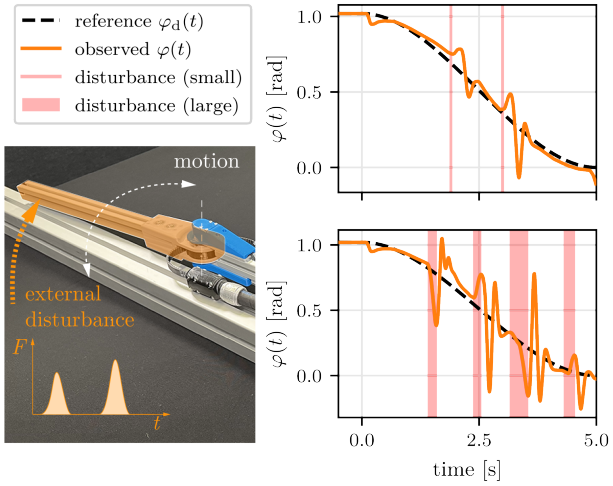


Fig. 6. Tracking performance of ANODEC in two disturbed trials. In the first trial (top) there are two small disturbances (poking the endeffector with a stick) whereas in the second trial (bottom) there are four large disturbances (grabbing and briefly holding the endeffector). ANODEC remains stable and behaves as expected in both trials.

the cases covered by the training dataset. To verify that ANODEC generalizes beyond the step reference signals (used internally by ANODEC, see (10)), we test the performance on two additional, qualitatively different reference signal distributions: Double-step and cubic-spline reference trajectories. In total, 16 different reference signals are used to validate ANODEC. One exemplary reference signal for each of the three reference signal distributions is illustrated in Fig. 5 (dashed black line).

D. ANODEC Reference Tracking Performance

We validate the proposed method (cf. Section III) and compare to the baseline controller (cf. Section IV-B) for 16 different reference signals (cf. Section IV-C) on the experimental setup of the PSA (cf. Section IV-A).

ANODEC is able to outperform the PID controller baseline for all three reference signal distributions, as can be seen in Table I. Overall, ANODEC consistently outperforms the PID controller baseline in all 16 reference signals individually, and achieves a, on average, $\approx 45.9\%$ lower RMSE tracking error across all reference signals while requiring less experimental interaction time (30 s versus 55 s). For each reference signal distribution, one exemplary reference signal is drawn. The performance of ANODEC compared to the PID controller baseline is shown in Figure 5.

Additionally, in Figure 6 we show that ANODEC is stable even if the trials are heavily disturbed.

These results show that ANODEC enables automatic yet competitive controller design for a PSA which, in contrast to conventional methods, requires no human expertise.

E. System Variation

To demonstrate the applicability of ANODEC to a second experimental system (Setup 2), we modify the experimental setup as it is described in Section IV-A. This variation significantly alters the system dynamics because the arm now rotates vertically, and the attached mass introduces a strong nonlinear term due to gravity.

The trial duration is extended from 5 s to 8 s and, as before, we record six input-output pairs (five for training, one for validation), now resulting in 48 seconds of data. For this second use case, ANODEC successfully designs a

competitive controller. The achieved RMSEs are summarized in Table I.

V. Conclusion

In this contribution, we have demonstrated that the data-driven method ANODEC can enable PSAs to – fully automatically – learn to perform agile, non-repetitive motions, from only $30\,\mathrm{s}$ of experimental interaction time. Utilizing only input-output data and without any prior model knowledge, ANODEC designs a single feedback controller that enables the tracking of arbitrary reference signals, which is then validated on multiple, qualitatively different, and even out-of-training-distribution reference signals. Specifically, it is shown that ANODEC can outperform a manually tuned PID baseline and achieves an up to 45.9% lower RMSE tracking error. Overall, this contribution not only reinforces the validity of ANODEC but also represents a significant advancement towards developing practical, user-friendly PSAs capable of learning to perform agile movements with only a minimal amount of experimental interaction time.

Future work will involve validation on different experimental PSAs, including PSAs with multiple DOFs, and increasing the autonomy of ANODEC by, e.g., self-tuning of involved hyperparameters (such as $\lambda^{(c)}$) and autonomous data acquisition.

References

[1] J. Walker, T. Zidek, C. Harbel, *et al.*, "Soft robotics: A review of recent developments of pneumatic soft actuators," *Actuators*, vol. 9, no. 1, 2020. DOI: 10.3390/act9010003.

[2] T.-L. Habich, J. Haack, M. Belhadj, D. Lehmann, T. Seel, and M. Schappler, "Sponge: Open-source designs of modular articulated soft robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, 2024. DOI: 10.1109/LRA.2024.3388855.

[3] P. Polygerinos, N. Correll, S. A. Morin, *et al.*, "Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing, control, and applications in human-robot interaction," *Advanced Engineering Materials*, vol. 19, no. 12, 2017. DOI: https://doi.org/10.1002/adem.201700016.

[4] M. Martens, J. Zawatzki, T. Seel, and I. Boblan, "A pneumatic-muscle-actuator-driven knee rehabilitation device for cam therapy," in *EMBC 2019*, 2019. DOI: 10.1109/EMBC.2019.8856526.

[5] D. Müller, A. Raisch, A. Hildebrandt, and O. Sawodny, "Nonlinear model based dynamic control of pneumatic driven quasi continuum manipulators," in *2020 IEEE/SICE SII*, 2020. DOI: 10.1109/SII46433.2020.9026206.

[6] C. Della Santina, C. Duriez, and D. Rus, "Model-based control of soft robots: A survey of the state of the art and open challenges," *IEEE Control Systems Magazine*, vol. 43, no. 3, 2023. DOI: 10.1109/MCS.2023.3253419.

[7] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft Robotics*, vol. 5, 2018. DOI: 10.1089/soro.2017.0007.

[8] V. Falkenhahn, *Modellierung und modellbasierte Regelung von Kontinuum-Manipulatoren* (Berichte aus dem Institut für Systemdynamik Universität Stuttgart). Aachen: Shaker Verlag, 2017, vol. 32.

[9] M. Martens, T. Seel, J. Zawatzki, and I. Boblan, "A novel framework for a systematic integration of pneumatic-muscle-actuator-driven joints into robotic systems via a torque control interface," *Actuators*, vol. 7, no. 4, 2018. DOI: 10.3390/act7040082.

[10] D. A. Haggerty, M. J. Banks, E. Kamenar, *et al.*, "Control of soft robots with inertial dynamics," *Science Robotics*, vol. 8, no. 81, 2023. DOI: 10.1126/scirobotics.add6864.

[11] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, 2021. DOI: 10.1109/TRO.2020.3038693.

[12] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *Robosoft 2018*, 2018. DOI: 10.1109/ROBOSOFT.2018.8404894.

[13] C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. Killpack, "Using first principles for deep learning and model-based control of soft robots," *Frontiers in Robotics and AI*, vol. 8, 2021. DOI: 10.3389/frobt.2021.654398.

[14] P. Hyatt, D. Wingate, and M. Killpack, "Model-based control of soft actuators using learned non-linear discrete-time models," *Frontiers in Robotics and AI*, vol. 6, 2019. DOI: 10.3389/frobt.2019.00022.

[15] M. Kasaei, K. K. Babarahmati, Z. Li, and M. Khadem, "A data-efficient neural ODE framework for optimal control of soft manipulators," in *7th Annual Conference on Robot Learning*, 2023.

[16] M. Hofer, L. Spannagl, and R. D'Andrea, "Iterative learning control for fast and accurate position tracking with an articulated soft robotic arm," in *IROS 2019*, 2019. DOI: 10.1109/IROS40897.2019.8967636.

[17] D. Schindele and H. Aschemann, "P-type ilc with phase lead compensation for a pneumatically driven parallel robot," in *2012 American Control Conference (ACC)*, 2012. DOI: 10.1109/ACC.2012.6314641.

[18] H. Ma, D. Büchler, B. Schölkopf, and M. Muehlebach, "A learning-based iterative control framework for controlling a robot arm with pneumatic artificial muscles," in *Proceedings of Robotics: Science and Systems XVIII (R:SS 2022)*, 2022. DOI: 10.15607/RSS.2022.XVIII.029.

[19] M. Deisenroth and C. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search.," in *ICML 2011*, 2011.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017. DOI: 10.48550/arXiv.1707.06347.

[21] P. Bevanda, M. Beier, S. Heshmati-Alamdari, S. Sosnowski, and S. Hirche, "Towards Data-driven LQR with Koopmanizing Flows," *IFAC-PapersOnLine*, 6th IFAC Conference, vol. 55, no. 15, 2022. DOI: 10.1016/j.ifacol.2022.07.601.

[22] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-Driven MPC for Quadrotors," *CoRR*, 2021. DOI: 10.48550/arXiv.2102.05773.

[23] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, 2006. DOI: 10.1109/MCS.2006.1636313.

[24] M. Meindl, D. Lehmann, and T. Seel, "Bridging reinforcement learning and iterative learning control," *Frontiers in Robotics and AI*, vol. 9, 2022. DOI: 10.3389/frobt.2022.793512.

[25] S. Bachhuber, I. Weygers, and T. Seel, "Neural ODEs for Data-Driven Automatic Self-Design of Finite-Time Output Feedback Control for Unknown Nonlinear Dynamics," *IEEE Control Systems Letters*, vol. 7, 2023. DOI: 10.1109/LCSYS.2023.3293277.

Appendix

Algorithm 1 Input Trajectory Generator

Require: $T \in \mathbb{R}_{>0}$, $f \in \mathbb{N}_{>0}$
  function GENERATESINUSOIDAL
    $\omega \leftarrow 2\pi f$
    us $\leftarrow \sin(\omega\ \mathtt{range}(0, T, 0.01))\frac{\sqrt{\omega}}{2}$
    return us

Require: $T \in \mathbb{R}_{>0}$, $\Delta t_{\min} \in \mathbb{R}_{>0}$, $\Delta t_{\max} > \Delta t_{\min}$, $u_{\min/\max} \in \mathbb{R}$
  function DRAWSPLINE
    ts, us $\leftarrow [0.0], [0.0]$
    while $\mathtt{last}(\mathrm{ts}) < T$ **do**
      $\mathrm{t} \leftarrow \mathtt{last}(\mathrm{ts}) + \mathtt{uniform}(\Delta t_{\min}, \Delta t_{\max})$
      $\mathrm{u} \leftarrow \mathtt{coinflip}(\mathtt{last}(\mathrm{us}), \mathtt{uniform}(u_{\min}, u_{\max}))$
      ts, us $\leftarrow \mathtt{append}(\mathrm{ts}, \mathrm{t}), \mathtt{append}(\mathrm{us}, \mathrm{u})$
    us $\leftarrow \mathtt{cubicInterpolate}(\mathtt{range}(0, T, 0.01), \mathrm{ts}, \mathrm{us})$
    return us