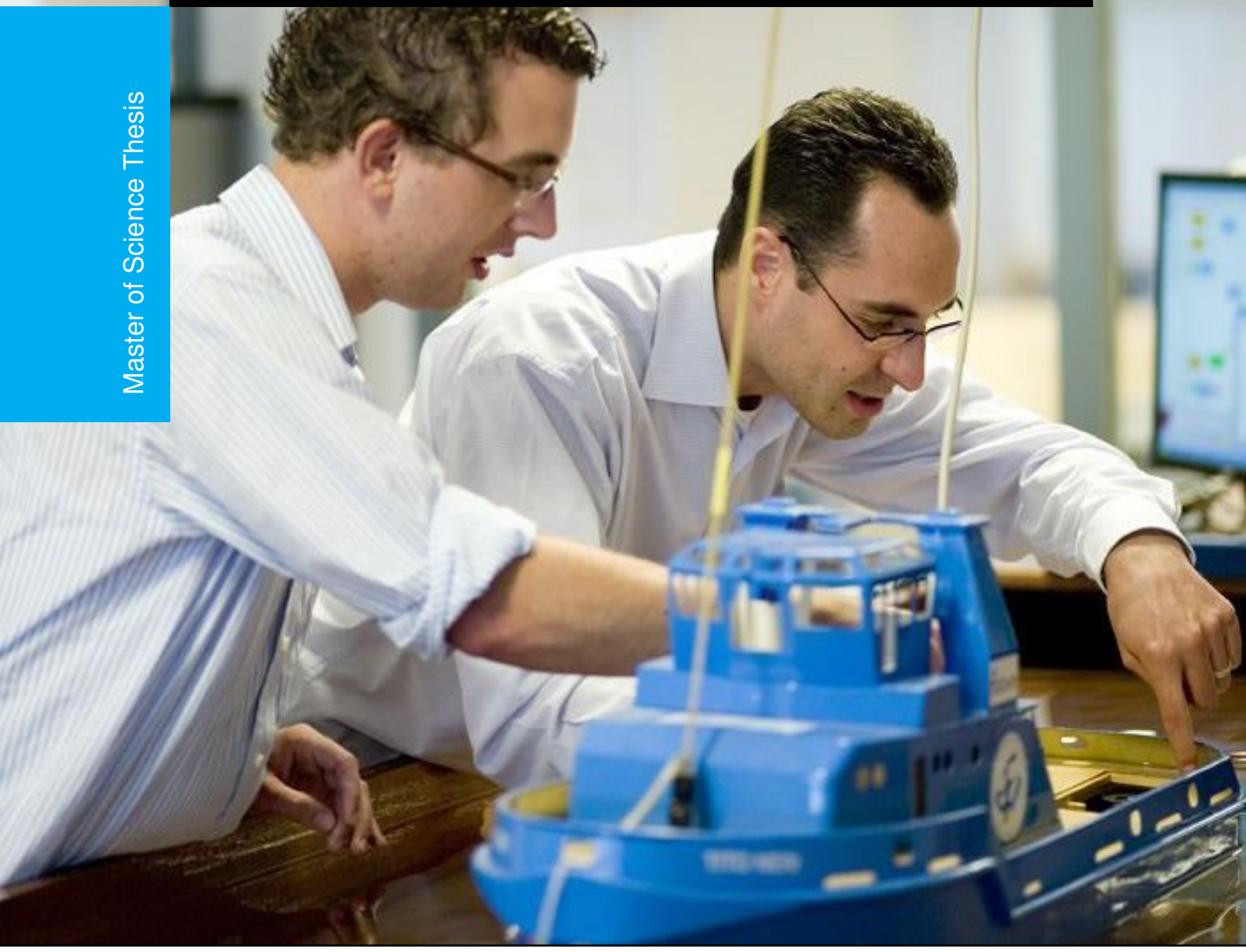
IMU-to-Segment Calibration with Stiff Joint Using Deep Learning Integrating Kinematic Constraints

Liuyi Zhu

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

IMU-to-Segment Calibration with Stiff Joint Using Deep Learning Integrating Kinematic Constraints

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft University of Technology

Liuyi Zhu

January 20, 2025

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Abstract

IMU-to-Segment (I2S) calibration is a critical step in using Inertial Measurement Unit (IMU) for human motion capture, as it determines the relative orientation and position between the IMU and the body segment it is attached to. Traditional constraint-based method rely on kinematic constraints to perform I2S calibration but fail in scenarios where no relative motion occurs between connected segments. To address this limitation, this thesis extends existing deep learning approaches for I2S calibration to the stiff case and investigates methods to further enhance calibration accuracy.

After completing I2S calibration using deep learning with a single IMU in stiff case, this thesis further explores joint training with dual-IMU model and integrates kinematic constraints into the model. Experiment results demonstrate that the joint training allows the models to leverage inter-IMU motion information, improving the model performance. Furthermore, integrating kinematic constraints with appropriate weights into the loss function of deep learning model improves calibration accuracy by guiding predictions to satisfy physical constraints. However, overly large constraint weights may result in larger calibration error.

This thesis provides insights into how deep learning can be adapted to address the challenges of I2S calibration in stiff joint scenarios. It also combines deep learning with kinematic information through joint training and the integration of kinematic constraints, achieving improved calibration accuracy. Future work will explore the application of this approach to real-world motion data and the integration of diverse kinematic constraints.

Table of Contents

Preface

The idea of the thesis topic came after discussions with my supervisors dr. Manon Kok and Ruiyuan Li of Delft Center for Systems and Control (DCSC). After reviewing a large amount of literature in human motion tracking, I have developed a keen interest in the application of kinematic constraints for I2S calibration in human motion capture. Utilizing the unique properties of human body's structure to address challenges in this field is particularly intriguing to me. Therefore, I decided to further explore this path.

After further reviewing the literature, I found that using kinematic constraints for I2S calibration is not suitable for all scenarios. For instance, when there is no relative motion between two connected segments, this method fails to achieve calibration. Meanwhile, other studies have demonstrated the feasibility of using deep learning for I2S calibration without relying on kinematic constraints. Motivated by these findings, I decided to explore deep learning as an alternative approach to achieve I2S calibration in stiff case.

I would like to thank my supervisors dr. Manon Kok and Ruiyuan Li for their guidance and feedback completing this thesis.

Delft, University of Technology — Liuyi Zhu
January 20, 2025

“In the future, airplanes will be flown by a dog and a pilot. And the dog’s job will be to make sure that if the pilot tries to touch any of the buttons, the dog bites him.”

— *Scott Adams*

Chapter 1

Introduction

1-1 Background

Human motion capture is crucial for understanding and analyzing complex body movements, which has applications in numerous fields including healthcare [6][31], sports performance [49], ergonomics [5], and entertainment [31]. It provides detailed insights into biomechanics, enabling the development of better rehabilitation strategies [17], injury prevention protocols [38], and improved athletic training techniques [28][2]. Human body motion capture is typically achieved using either vision-based technologies [27] or inertial sensors [36][28]. The primary principle of using vision-based technologies for human body motion capture is to use cameras and image processing algorithms to measure and track the human body movement. The main drawback of this method is that it can be constrained by moving space. If the motion space is extensive or if there are obstacles between cameras and human body, accurate motion capture becomes challenging. Compared with vision-based technologies, using Inertial Measurement Unit (IMU) is a method with fewer environmental constraints and simpler equipment requirements [37]. It offers greater flexibility in terms of motion capture [50].

To achieve motion capture, IMUs need to be equipped on different segments of human body, which are pairwise connected by joints with one, two or three rotational Degree of Freedom (DoF). An IMU typically comprises 3D gyroscopes, 3D accelerometers, and 3D magnetometers. In motion capture, the commonly used IMU measurements include angular velocity obtained from gyroscopes and acceleration measured by accelerometers. By integrating the angular velocity, the orientation change of the IMU can be determined. By double integrating the acceleration after subtracting gravity, the displacement of the IMU can be obtained [39][12].

IMU-to-Segment (I2S) calibration is an essential step in human motion tracking using IMU. I2S calibration involves estimating the calibration parameters, as well as the relative position and orientation between the IMU frame and body segment frame [24]. After establishing the spatial relationship between two frames, the IMU orientation estimated from measurements can be converted into the estimated pose of body segment, achieving human motion capture [45].

Some earlier I2S calibration methods, such as manual calibration or using defined poses or motions, may introduce significant human-induced errors during the calibration process [26]. Another commonly used approach is to utilize kinematic constraints for I2S calibration. The constraint-based method employs optimization techniques, such as Gauss-Newton method, to estimate the calibration parameters that best satisfies the kinematic constraints [23][29]. This method only requires the subject to perform arbitrary movements. Therefore, it is more flexible and more accurate compared to previous methods. There is also a study that utilizes deep learning for I2S calibration, employing end-to-end learning to directly derive calibration parameters from IMU measurements [55].

1-2 Research Gap

A standard solution for I2S calibration is the utilization of kinematic constraints, due to its high accuracy and the requirement of only arbitrary motion. Current studies using this approach predominantly assume that sufficient relative motion exists between segments for effective I2S calibration [30].

However, if there is little or no relative motion between segments, segments will move like a single rigid body and IMU measurements are no longer sufficient for calibration parameters identification [30]. When the calibration error is substantial, the accuracy of subsequent human posture estimations will also decrease [26].

During the application of kinematic constraints for I2S calibration, it is possible that the collected sensor data is obtained under conditions where there is limited or no relative motion between segments. When using IMUs to monitor movement or provide rehabilitation therapy for patients or elderly individuals, certain conditions—such as osteoarthritis [35], cerebral palsy [52], knee injuries [46][47], or Anterior cruciate ligament (ACL) injuries [13]—may result in limited joint movement. In these applications, manual calibration or static pose methods are typically used for I2S calibration, as patients or elderly individuals may be unable to perform sufficient arbitrary movements. However, manual calibration can introduce large errors in I2S calibration [55]. Therefore, it is necessary to find a more accurate I2S calibration method under stiff conditions to ensure the reliability and applicability of the system in these scenarios.

In the absence of relative motion between segments, kinematic constraints may no longer provide sufficient information for calibration. In this scenario, deep learning could offer a potential solution. In previous literature, deep learning has been employed to derive calibration parameters by utilizing neural networks to learn patterns of measurements during motion [55]. This approach adopts an end-to-end framework that focuses exclusively on measurements from a single IMU, without relying on kinematic constraints existing between different segments of human body.

However, the deep learning method used in previous study [55] also has a limitation. Different body segments exhibit varying patterns of motion. Some segments have simpler movement patterns, making it easier for deep learning to capture. While others display more complex and variable motions which are harder for deep learning to learn. This variation leads to differences in average error for I2S calibration across segments, with some segments experiencing higher calibration errors [55]. In contrast, the accuracy of calibration results using constraint-based methods is not affected by the complexity of motion, since movements of varying complexity all adhere to these constraints. Therefore, even for the segment with complex motion, kinematic constraints can still provide equally valuable information for its calibration. For the deep learning method in [55], the neural networks for IMUs

placed on different parts of body are trained separately and independently, which means that the kinematic constraints between different IMUs are not considered during training. In other fields, some studies have combined deep learning with physical constraints to make the predictions of deep learning more consistent with physical models [43][20]. In the context of I2S calibration, this approach could potentially address aforementioned limitations to some extent, completing I2S calibration with higher accuracy, especially for the segments with complex motion.

1-3 Motivation and Research Questions

The motivation for this thesis is that the constraint-based methods previously used in the literature [42][44][29][23] are unable to achieve I2S calibration for two segments with stiff hinge joint, and no existing research addresses how to accomplish I2S calibration in this case. The objective of this thesis is to extend the deep learning method used for I2S calibration in [55] to the case of two segments connected by a stiff hinge joint. Additionally, it investigates if the integration of kinematic constraints can, to some extent, compensate the limitation of deep learning that do not account for kinematic information, thereby further improving calibration accuracy.

The research questions for this thesis are as follows:

1. Is it possible to extend the deep learning method in [55] to perform I2S calibration for two segments connected by a stiff hinge joint?

2. How will incorporating the kinematic constraint with varying weights into deep learning impact calibration accuracy compared to using deep learning method alone?

To address these research questions, the study in this thesis can be divided into the following four parts:

1. The first part will employ the constraint-based method from previous literature to show that in the absence of relative motion between segments, significant error may arise in I2S calibration.

2. The second part will modify the deep learning model based on prior research [55] to investigate if more accurate I2S calibration can be achieved in stiff case. It will also explore different data configurations for deep learning model, such as the ideal input sliding window length and input normalization. This part addresses the first research question.

3. Since the integration of kinematic constraints involves the relationship between two IMUs' measurements, the third part of the experiment tests the performance of different dual-IMU networks, which couples two single-IMU models. The goal is to find the appropriate coupling strategy.

4. In the fourth part, kinematic constraints will be integrated into the dual-IMU network to explore how the integration of kinematic constraint impact calibration results, including the effect of different constraint weights. This part answers the second research question.

1-4 Contribution

The main contributions of this thesis are as follows:

1. Extension of deep learning method for I2S calibration in stiff case: This thesis investigates the applicability of deep learning method for I2S calibration in scenarios where two IMUs are attached to segments connected by a hinge joint with constant joint angle during motion.
2. Integration of kinematic constraints: This research integrates kinematic constraints with varying weights into deep learning frameworks for I2S calibration. It evaluates how the constraint with different weight influences the I2S calibration results and explores its potential to improve accuracy beyond deep learning method.

1-5 Outline

After the introduction section, this thesis is primarily divided into five parts: related work, method, simulation setting, simulation results, and conclusion.

The related work section 2 summarizes various methods from previous literature used for I2S calibration. The method section 3 provides a detailed explanation of the specific physical model employed in this thesis, the algorithm used in constraint-based method and the deep learning models used in simulation. The simulation setting section 4 clarifies the establishment of the simulated model, the generation of the simulation dataset, and specific configurations applied in various parts of the simulation. The simulation results section 5 lists and analyzes the results of different parts of simulation. Finally, the conclusion section 6 draws the conclusion for this thesis and offers insights into future work.

Chapter 2

Related Work

2-1 Review of Existing I2S Calibration Methods

2-1-1 Aligning IMU Manually

In previous literature, multiple methods have been proposed to achieve I2S calibration. In [4], manual calibration has been validated as a I2S calibration method. The manual calibration method requires operators to position the sensor on a human segment rigorously by aligning the sensor edges with the anatomy. The manual calibration does not require any computational step. However, manual calibration has some drawbacks. Firstly, it is susceptible to individual subjective judgment and operational skills, which may lead to significant calibration errors. Secondly, manual calibration requires a considerable amount of time. If the calibration is performed before each test, it can cost substantial time and labor.

2-1-2 Predefined Calibration Pose of Movement

Using predefined poses for calibration is also a common I2S calibration method. In [34], the sensor measurements were recorded in known static poses, and subsequently measurements were associated with actual body posture to infer the transformation matrix from sensor coordinate to body segment coordinate. In addition, functional calibration is also a method of calibrating sensors by measuring specific functional movements. In [8], [15], [16] and [14], the relative orientation of IMU coordinate system was determined by analyzing movement patterns of body during predefined actions.

The method using predefined postures or movements can complete calibration by employing simple poses or actions to establish the connection between IMU measurements and the segment. However, this calibration approach also introduces significant uncertainty. Firstly, if the subjects fail to execute the required postures or movements accurately due to lack of experience or movement constraints, it may result in calibration errors. Additionally, this calibration method can also be time-consuming, requiring a amount of time for subjects to perform the specified poses or movements.

2-1-3 Constraint-based Method

Commonly Used Constraints

Compared to methods that require predefined procedures or manual adjustments, using kinematic constraints for I2S calibration is a more reliable approach with higher accuracy. Due to the natural motion constraints between different parts of the human body, these constraints can provide valuable information for I2S calibration. The commonly used kinematic constraints for the physical model with two segments and two IMUs in I2S calibration include the following:

First, there are constraints related to angular velocity derived from the limited DoF of the joints. When modeling human body parts, in certain situations, to simplify the model, specific joints of the human body may be simplified to have only one or two rotational DoF. In this case, there will be certain constraints between the angular velocities of the two segments, as their relative motion is restricted to rotation around one or two fixed joint axes.

In [42], [33], [32] and [22], the angular velocity constraint for hinge joint was deduced for I2S calibration. Since a hinge joint has only one fixed joint axis j_0, the angular velocity measured in two IMU frames (ω_1 and ω_2) vary solely in terms of the joint angle velocity and a rotation matrix. If ω_1 and ω_2 can be decomposed into components parallel to j_0 and components perpendicular to j_0, the perpendicular components of ω_1 and ω_2 should have same magnitude since two segments only undergo independent relative rotation in the direction parallel to the joint axis. In other words, their projections into the joint plane, i.e. the plane to which j_0 is the normal vector, have the same lengths for each instant in time, which can be represented as:

$$||\omega_{1,t} \times j_1||_2 = ||\omega_{2,t} \times j_2||_2 \ \forall t \tag{2-1}$$

where j_i is the joint axis of hinge joint represented in two IMU frames.

In [29] and [23], similar angular velocity constraint was extended to joint with 2-DoF. The angular velocities measured by two IMUs can be used to compute the relative angular velocity ω_{s_1,s_2} between two segments. Since there are two rotational DoF, ω_{s_1,s_2} can also be decomposed into motions around two fixed rotation axes ($j_{1,2}$ and $j_{2,2}$). The ω_{s_1,s_2} lies in the plane formed by these two rotation axes $j_{1,2}$ and $j_{2,2}$, which means that it is perpendicular to the normal vector of this plane. The normal vector can be expressed as the cross product of two rotation axes. The constraint can be expressed as:

$$\begin{aligned} &\omega^G_{s_1,s_2} = -\omega^G_{2,t} + \omega^G_{1,t} = \alpha_t j^G_{1,2} + \beta_t j^G_{2,2} \\ &\omega^G_{s_1,s_2} \cdot (j^G_{2,2} \times j^G_{1,2}) = 0 \end{aligned} \tag{2-2}$$

where α_t and β_t are time-varying scalars. All measurements and joint axes are expressed in the fixed global frame G to ensure coordinate system consistency during calculation.

Another commonly used kinematic constraint is derived from the acceleration measurements of the two IMUs. In [41],[42] and [40], the acceleration of each sensor was decomposed into the sum of the acceleration at joint center (a^{cen}_1 and a^{cen}_2) and acceleration resulting from rotation of that sensor around the joint center. The magnitudes of a^{cen}_1 and a^{cen}_2 should be equal. The measured acceleration a_i can be represented as linear combination of a^{cen}_i and the relative position between IMU frame and joint center r_i:

$$a_i = a^{\text{cen}}_i + K(\omega_i, \dot{\omega}_i) r_i$$

$$K(\omega_i, \dot{\omega}_i) = \begin{bmatrix} -\omega_i^{y^2} - \omega_i^{z^2} & \omega_i^x \omega_i^y - \dot{\omega}_i^z & \omega_i^x \omega_i^z + \dot{\omega}_i^y \\ \omega_i^x \omega_i^y + \dot{\omega}_i^z & -\omega_i^{x^2} - \omega_i^{z^2} & \omega_i^y \omega_i^z - \dot{\omega}_i^x \\ \omega_i^x \omega_i^z + \dot{\omega}_i^y & \omega_i^y \omega_i^z + \dot{\omega}_i^x & -\omega_i^{x^2} - \omega_i^{y^2} \end{bmatrix}$$

For convenience $K(\omega_i, \dot{\omega}_i)$ will be written as K_i.

The magnitudes of a_1^{cen} and a_2^{cen}, as measured by two IMUs, are identical. If the joint is a hinge joint, the lengths of these vectors projected onto the joint axis in their respective coordinate systems are also the same:

$$j_1^\intercal a_1^{\text{cen}} = j_2^\intercal a_2^{\text{cen}}$$

Constraints between accelerations measured by two IMUs can be derived:

$$j_1^\intercal a_1 - j_2^\intercal a_2 = j_1^\intercal a_1^{\text{cen}} - j_2^\intercal a_2^{\text{cen}} + j_1^\intercal K_1 r_1 - j_2^\intercal K_2 r_2 = j_1^\intercal K_1 r_1 - j_2^\intercal K_2 r_2$$

Under the condition that the acceleration due to rotation around the joint center is minimal($j_i^T K_i r_i \approx 0$), the acceleration constraint can be expressed as follows:

$$j_1^\intercal a_1 - j_2^\intercal a_2 \approx 0 \tag{2-3}$$

Algorithm

The kinematic constraints 2-1, 2-2 and 2-3 analyzed above incorporate known IMU measurements, including acceleration and angular velocity, as well as the unknown representations of the joint axes in the IMU frame, represented as j_1 and j_2. The basic idea of the constraint-based method is to estimate j_1 and j_2 that best satisfy these kinematic constraints through optimization techniques. Since the representation of the joint axes in the segment frame j_0 is known, the relative orientation between the segment frame and the IMU frame can be derived, thereby completing the I2S calibration.

In [42], [41] and [32], the angular velocity constraint 2-1 and the acceleration constraint 2-3 are utilized to perform I2S calibration for two segments connected by hinge joint.

Assume a measurement sequence with length N, precisely $(\omega_{1,t}, \omega_{2,t}, a_{1,t}, a_{2,t})_{t=1}^N$ are provided, where $N >> 4$. Based on constraint 2-1 and 2-3, the residual vector is defined as:

$$e_\omega(t) = w_{\omega,t}[\,||\omega_{1,t} \times \hat{j}_1|| - ||\omega_{2,t} \times \hat{j}_2||\,] \tag{2-4}$$

$$e_a(t) = w_{a,t}[\hat{j}_1^\intercal a_{1,t} - \hat{j}_2^\intercal a_{2,t}] \tag{2-5}$$

where w_ω and w_a are weights for the angular velocity constraint 2-1 and the acceleration constraint 2-3. $\hat{j}_i$ is the estimated joint axis in IMU frame.

The sum of squares of residuals that needs to be minimized is defined as:

$$\Psi(\hat{j}_1, \hat{j}_2) = \sum_{k=1}^{N} \left([e_\omega(t)]^2 + [e_a(t)]^2 \right) \tag{2-6}$$

To reduce the number of unknown parameters, j_1 and j_2 are assumed to be unit vectors. Therefore, these two vectors can be represented by an inclination and an azimuth respectively, reducing the number of unknown parameters from six to four. The vector of unknown parameters can be represented as:

$$x := (\phi_1, \theta_1, \phi_2, \theta_2)^\intercal \tag{2-7}$$

where ϕ_i and θ_i are inclination and azimuth of $\hat{j}_i$ in the i^{th} IMU's coordinate system. The estimated unit joint axis vector can be represented using these two angles:

$$\begin{aligned}\hat{j}_1 &= (\cos(\phi_1)\cos(\theta_1), \cos(\phi_1)\sin(\theta_1), \sin(\phi_1))^\intercal \\ \hat{j}_2 &= (\cos(\phi_2)\cos(\theta_2), \cos(\phi_2)\sin(\theta_2), \sin(\phi_2))^\intercal\end{aligned} \tag{2-8}$$

The optimization problem for I2S calibration can be formulated as:

$$\hat{x} = \text{argmin}_x \Psi(x) \tag{2-9}$$

To minimize the sum of squared errors and achieve convergence, the Gauss-Newton algorithm is selected as the optimization problem solver for this nonlinear least squares problem. It is worth noting the directions of j_1 and j_2 need to be aligned. Using Gauss-Newton method may result in convergence to a local minimum. At this local minimum, the directions of the two joint axes j_1 and j_2 are opposite to each other. To avoid this issue, after obtaining the first local minimum $\hat{x}^{(1)} = [\hat{\phi}_1, \hat{\theta}_1, \hat{\phi}_2, \hat{\theta}_2]$ using the Gauss-Newton method, an additional initial point is selected by reversing the full direction of the joint axis j_2: $\hat{x}(0) = [\hat{\phi}_1, \hat{\theta}_1, -\hat{\phi}_2, \hat{\theta}_2 + \pi]$. At this new initial point, the Gauss-Newton method is applied again to find the corresponding local minimum $\hat{x}^{(2)}$. The cost function $\Psi(x)$ is then evaluated for both local minima, and the solution with smaller cost is considered the correct one, where the two joint axes are properly aligned. The complete constraint-based method for I2S calibration is described in Algorithm 1[32].

In [23] and [29], the similar algorithm is extended to achieve I2S calibration for two segments connected by a joint with two degrees of rotational freedom by using the angular velocity constraint for 2-DoF joint 2-2 and the acceleration constraint 2-3. Additionally, different weights were assigned to each constraint, allowing for greater adaptability across various movement conditions. In [25], the angular velocity constraint 2-1 was employed to identify j_1 and j_2 using Principal Component Analysis (PCA) method.

One drawback of the constraint-based method is that no verification is typically performed to ensure that the collected motion data contain adequate motion information before using it for calibration [32]. To slove this issue, [32] proposes a plug-and-play I2S calibration method that continuously evaluates the amount of informative motion within the measurement sequence. Non-informative motions are filtered out, and once sufficient informative motion data are collected, automatic I2S calibration is performed. This algorithm effectively mitigates the risk of calibration errors caused by non-informative motion data.

2-1-4 Deep Learning

There is also a study that utilizes deep learning methods for I2S calibration, using end-to-end learning to directly derive calibration parameters from IMU measurements collected in walking sequence [55]. This research focused on the lower body of the human anatomy, encompassing seven segments, namely the pelvis, two upper legs, two lower legs, and two feet. Each segment is equipped with an individual IMU. The architecture of the deep learning model in [55] comprises Convolutional Neural Network (CNN) layers, Recurrent Neural Network (RNN) layers and Fully Connected (FC) layers. Within RNN layers, Long Short-Term Memory (LSTM) layers are utilized to capture the long-term dependencies within the sensor data time series. In [55], the input training dataset consists of raw IMU data from various subjects walking at different speeds, and output of model is the relative orientation

Algorithm 1 Constraint-based method for I2S calibration

Required: IMU measurements $(\omega_{1,t}, \omega_{2,t}, a_{1,t}, a_{2,t})_{t=1}^{N}$, initial estimate $\hat{x}(0)$, constraint weight w_ω and w_a, tolerance Ψ_{tol}
Output: estimated $\hat{j}_1$, $\hat{j}_2$
for $i \in 1, 2$ **do**
    $k \leftarrow 0$ ▷ Begin Gauss-Newton
    $\Delta_\Psi \leftarrow \Psi_{tol}$
    Initialize $\leftarrow \hat{x}(0)$
    $\Psi(0) \leftarrow \Psi(\hat{x}(0)$ ▷ $\Psi(x)$ defined by 2-6
    while $\Delta_\Psi \geq \Psi_{tol}$ **do**
        Compute residuals $r(\hat{x}(k))$ and Jacobian $J(\hat{x}(k)) = \frac{\partial r(\hat{x}(k))}{\partial \hat{x}(k)}$
        Update step: $\Delta_x(k) = (J(\hat{x}(k))^\intercal J(\hat{x}(k)))^{-1} J(\hat{x}(k))^\intercal r(\hat{x}(k))$
        Update parameters: $\hat{x}(k+1) = \hat{x}(k) - \Delta_x(k)$
        $k \leftarrow k+1$
        $\Psi(k) \leftarrow \Psi(\hat{x}(k))$
        $\Delta_\Psi \leftarrow |\Psi(k) - \Psi(k-1)|$
    end while
    $\hat{x} \leftarrow \hat{x}(k)$ ▷ End Gauss-Newton
    $\hat{x}^{(i)} = [\hat{\phi}_1, \hat{\theta}_1, \hat{\phi}_2, \hat{\theta}_2]^\intercal \leftarrow \hat{x}$
    $\hat{x}(0) \leftarrow [\hat{\phi}_1, \hat{\theta}_1, -\hat{\phi}_2, \hat{\theta}_2 + \pi]^\intercal$ ▷ initialize at $-\hat{j}_2$
end for
$\hat{x} = [\hat{\phi}_1, \hat{\theta}_1, \hat{\phi}_2, \hat{\theta}_2]^\intercal \leftarrow \text{argmin}_{x \in \{\hat{x}^{(1)}, \hat{x}^{(2)}\}} \Psi(x)$
$\hat{j}_1 \leftarrow (\cos(\hat{\phi}_1)\cos(\hat{\theta}_1), \cos(\hat{\phi}_1)\sin(\hat{\theta}_1), \sin(\hat{\phi}_1))^\intercal$
$\hat{j}_2 \leftarrow (\cos(\hat{\phi}_2)\cos(\hat{\theta}_2), \cos(\hat{\phi}_2)\sin(\hat{\theta}_2), \sin(\hat{\phi}_2))^\intercal$
Return $\hat{j}_1$, $\hat{j}_2$

of IMU with respect to the segment. Walking is a relatively repetitive and patterned motion, so it is likely that the deep learning network in this study learned the regular walking pattern to complete I2S calibration effectively.

2-2 Research Gap and Possible Solution: Deep Learning

The primary approach for I2S calibration found in the literature is the constraint-based method, which is favored for its high accuracy and the ability to function with merely arbitrary motion. Existing research primarily assumes that adequate relative motion between segments is available. However, if sufficient informative motion cannot be obtained over an extended period, such as in cases where medical conditions limit joint rotation, this method will result in significant calibration errors [30].

In several studies using IMUs for motion monitoring or rehabilitation therapy in patients or elderly individuals, the movement between segments is restricted. In [35], sensors were integrated into pants to monitor rehabilitation training for patients with osteoarthritis. However, if there is significant relative movement between the clothing and the body, the resulting errors could affect the accuracy of the monitoring results. In [52], sensors were placed on the shank to monitor muscle changes in patients with cerebral palsy. The I2S calibration was performed manually. In [13], two IMUs were equipped on the thigh and calf to assess the severity of ACL injuries. The I2S calibration was achieved using a static pose performed by the participants. In [46] and [47], a similar IMU placement was used to monitor the movement of patients with knee injuries. For [46], IMU was integrated into clothing, and in [47] the I2S calibration was performed manually. These studies primarily rely on manual calibration or static pose for I2S calibration. However, as analyzed in sections 2-1-1 and 2-1-2, these calibration methods are prone to significant human-induced errors and time-consuming. Currently, there is no literature that has investigated how to perform reliable automatic I2S calibration in stiff case.

The deep learning method in [55] is achieved without depending on kinematic constraints between different segments of human body. Consequently, even in scenarios where relative motion is absent, the deep learning method may still successfully achieve I2S calibration.

However, the deep learning method in [55] was under the condition that the input IMU measurements had to be from subjects performing walking motions. This study did not account for random segment motions or scenario where there is no relative movement between segments. Additionally, the calibration networks for IMUs positioned on various body parts were trained separately and independently, meaning that the potential spatial relationships between different IMUs were not taken into account during training process. It may result in large differences in average calibration error across different segments. For example, in [55], the mean angle error around z-axis for I2S calibration using deep learning method is 14.35° for left upper leg. Whereas the error for left lower leg is 28.59°, which is twice that of the upper leg.

2-3 Integrating Kinematic Constraints in Deep Learning: Insights from Other Fields

This drawback of deep learning method could have been mitigated by incorporating kinematic constraints [7], as this integration allows the model to obtain more motion information during training, leading to predictions that better adhere to kinematic constraints. In the literature on I2S calibration,

there has been no attempt to combine deep learning with kinematic constraints for I2S calibration. However, in other fields, there are already numerous applications that integrate deep learning with relevant physical constraints.

In [43], a dual-driven fracturing effect evaluation model for coalbed methane reservoirs by integrating data-driven deep learning methods with physical constraints is developed. A neural network with physical constraints embedded in its loss function is proposed to enhance prediction accuracy by incorporating initial conditions and expert knowledge.

[20] proposes the Physics-Guided Neural Network (PGNN), a framework that integrates physics-based model simulations with observational data to enhance neural network predictions. This framework is demonstrated in lake temperature modeling, where physical relationships between temperature, density, and water depth are incorporated into the model. This paper employs two methods to integrate physical constraints. The first method incorporates the constraints into the loss function, while the second method uses approximate predictions obtained through physical relationships as additional inputs to the neural network.

[11] introduces a physics-constrained deep learning method for developing control-oriented models of building thermal dynamics. Using the Perron-Frobenius theorem, the method bounds eigenvalues to maintain system dissipativeness. Compared with traditional method, the physics-constrained deep learning method performs better in prediction accuracy and robustness.

From [43][20][11], it can be concluded that combining deep learning with physical constraints is particularly suitable for scenarios where well-established physical constraints exist, along with abundant raw data (e.g., sensor data) for training deep learning models. In these scenarios, relying solely on physical relationships for prediction may face challenges such as the inability to construct precise mathematical models due to the complexity of the system, insufficient information provided by the physical constraints (e.g., low-order constraints that fail to fully capture system dynamics), or low robustness in predictions caused by noisy sensor data. Conversely, using deep learning alone for prediction may produce results that significantly violate the physical constraints, as physical knowledge is not embedded in the training process. Combining both approaches allows for more accurate and robust predictions.

In I2S calibration case, there are certain established kinematic constraints. Additionally, in stiff case, these kinematic constraints can no longer provide sufficient information for I2S calibration. In such scenarios, combining deep learning methods with kinematic constraints could be a suitable approach to achieve more precise calibration.

Chapter 3

Method

This chapter provides a detailed description of the physical model, inertial measurement models and parameters used in this thesis, as well as the algorithm and deep learning models employed in simulation.

3-1 Physical Model and Parameters

The I2S calibration involves determining the relative position and orientation between coordinate frame of IMU and that of segment it is attached to. Comparative studies in sensor fusion for inertial body motion tracking reveal that the sensor's positional accuracy relative to a segment is generally less critical than its relative orientation accuracy [26]. Therefore, this thesis focuses solely on calibrating the relative orientation between IMUs and segments, while assuming that the relative position is known and fixed.

The physical model and parameters in this model are shown in Figure 3-1.

The model presented in this thesis consists of two rigid segments (s_1 and s_2) with length l_1 and l_2, connected by a hinge joint, which allows for rotation around a single axis. The segment frames are denoted as S_1 and S_2, while the global frame is denoted as G. The origin of segment frame coincides with the center of segment mass. The y-axis is aligned with hinge joint's rotation axis, x-axis is aligned with the line connecting two endpoints of segment, and z-axis is parallel to the sagittal plane of the segment, pointing outward from the segment's anterior surface. The joint axis is represented as j_0 in the segment frame. In this thesis j_0 is $[0, 1, 0]^\intercal$

Each segment is equipped with an IMU mounted in an arbitrary orientation on the respective segment. The IMU frames are denoted as I_1 and I_2 respectively. The origin of IMU frame coincides with the center point on the anterior surface of the segment. The anterior surface is defined as the side of the segment that faces forward during motion. In this thesis, to simplify the problem, the IMU frame is assumed to rotate only around its z-axis, and its z-axis is aligned with the z-axis of the corresponding segment frame. As a result, the relative orientation between S_i and I_i can be represented by a single

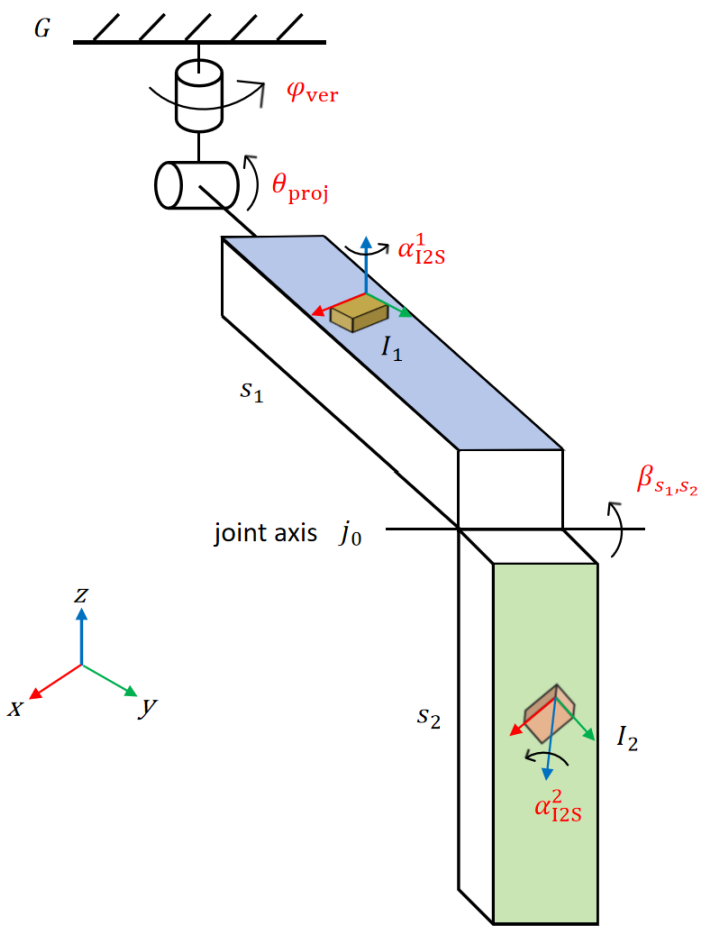


Figure 3-1: Physical Model and Parameters

angle, referred to as the I2S angle in the following content, denoted as α^i_{I2S}, $i = 1, 2$. This I2S angle corresponds to the rotation angle around the z-axis of I_i that aligns it with S_i.

There are also parameters that describe the motion of two segments: $\beta_{s_1,s_2}, \theta_{\text{proj}}, \phi_{\text{ver}}$. The range of variation for parameters is primarily based on the movements of the human thigh and shank during basic activities such as walking and running. For example, β_{s_1,s_2} corresponds to the flexion and extension angle of knee joint, which typically ranges from 40 to 180 degrees in a normal knee. Additionally, lengths of two segments are also referenced from the typical length ranges of adult thighs and shanks. This reference to real human models enhances the biological validity of the model and establishes a foundation for potential future work that may be applied to actual human I2S calibration [48].

The definitions, ranges, and symbolic representations of parameters in this model are listed in Table 3-1.

Parameters	Definition	Range
$\alpha_{\text{I2S}}^{1,2}$	rotation angle of I_1/I_2 frame around its z-axis related to S_1/S_2	-180-180(°)
β_{s_1,s_2}	the relative angle between longitudinal axes of s_1 and s_2, rotate around joint axis	40-180(°)
θ_{proj}	angle between s_1 and its projection onto the horizontal plane, present the inclination of s_1 relative to the horizontal ground.	0-90(°)
ϕ_{ver}	angle by which s_1 rotates around the vertical axis	0-360(°)
$l_{1,2}$	length of s_1/s_2	35-50(cm)

Table 3-1: Variable Parameters in Model

3-2 Inertial Measurement Models

IMU measurements include 3D angular velocity measured by the gyroscope, 3D acceleration measured by the accelerometer and magnetic field strength measured by the magnetometer. In previous I2S calibration studies, magnetometer data was typically omitted, largely because indoor environments often cause substantial magnetic disturbances, leading to unreliable measurements [25][32]. In this thesis, only angular velocity and linear acceleration are considered for performing I2S calibration.

The angular velocity and linear acceleration measured by the two IMUs at time t, expressed in their respective IMU frames, are given as: $\omega_{i,t}, a_{i,t},\ i = 1,2$.

The gyroscope measurements are modeled as:

$$\omega_{i,t} = \omega_{i,t}^{\text{true}} + b^{\omega} + e_{i,t}^{\omega},\ \text{for } i = 1,2 \tag{3-1}$$

where $\omega_{i,t}^{\text{true}}$ is the true angular velocity in I_i. The measurements are affected by a constant additive bias b^{ω} and noise $e_{i,t}^{\omega} \in \mathbb{R}^3$.

The accelerometer measurements are modeled as:

$$a_{i,t} = a_{i,t}^{\text{true}} + R^{I_i G} g^G + b^a + e_{i,t}^a,\ \text{for } i = 1,2 \tag{3-2}$$

where $a_{i,t}^{\text{true}}$ is the true acceleration in I_i. g^G is the gravitational acceleration in global frame, which is assumed to be constant, $R^{I_i G}$ describes the rotation from the global frame to IMU frame. Similarly, the measurements from accelerometer are affected by a constant additive bias b^a and noise $e_{i,t}^a \in \mathbb{R}^3$.

3-3 Constraint-Based Method

In first part of simulation, which demonstrates the inapplicability of constraint-based method in stiff case, the algorithm 1 from [32] is chosen. In [32], this algorithm was used to estimate the joint axis in the IMU frame (j_1 and j_2) for a model consisting of two segments connected by a hinge joint, thereby deriving the relative orientation between the IMU frame and the segment frame. The kinematic constraints utilized in [32] include the angular velocity constraint 2-1 and the acceleration constraint 2-3.

In [32], sufficient motion exists between the two segments. However, the simulations conducted in this thesis are performed under conditions with no relative motion. The following analysis in this section will examine whether the kinematic constraints used in [32] still contain useful information in stiff case. If the kinematic constraints do retain useful information, they could potentially be integrated into deep learning to provide valuable guidance for training process.

In case where two segments do not have relative motion, two segments move as a single rigid body, with the relative orientation between two IMU frames I_1 and I_2 can be represented by a rotation matrix R that remains constant over time [30].

The representation of the joint axis j_0 in the two IMU frames, denoted as j_1 and j_2, can also be converted between each other using this constant rotation matrix R:

$$j_2 = Rj_1 \tag{3-3}$$

Similarly, the accelerations and angular velocities measured by the two IMUs can also be transformed between each other using R:

$$a_2 = Ra_1 \tag{3-4}$$

$$\omega_2 = R\omega_1 \tag{3-5}$$

Based on 3-3 and 3-5, we have:

$$\begin{aligned} &\omega_2 \times j_2 = (R\omega_1) \times (Rj_1) = R(\omega_1 \times j_1) \\ &||\omega_1 \times j_1||_2 = ||R(\omega_1 \times j_1)||_2 = ||\omega_2 \times j_2||_2 \end{aligned} \tag{3-6}$$

Based on 3-3 and 3-4, we have:

$$j_2^\intercal a_2 = (Rj_1)^\intercal (Ra_1) = j_1 R^\intercal R a_1 = j_1^\intercal a_1 \tag{3-7}$$

The two derived constraints 3-6 and 3-7 are identical to the angular velocity constraint 2-1 and acceleration constraint 2-3 used in [32], indicating that these constraints still hold in the stiff case. However, if j_0 is replaced with any arbitrary axis in space, the two constraints remain valid. This suggests that even if the constraints hold, they do not necessarily provide useful information for joint axis identification.

However, this does not mean kinematic constraints are entirely devoid of useful information. If the joint axis in one of IMU frames (j_1 or j_2) is known—meaning that joint axis in one frame is accurately calibrated—it is still possible to use the constraint to infer joint axis in other IMU frame. This implies that the calibration results of two IMUs can compensate for each other. For example, if calibration of IMU1 is more accurate while calibration result of IMU2 has a higher error, it may be possible to use result from IMU1 as a reference to reduce error in IMU2's calibration. Essentially, the constraint may allow one IMU's more reliable result to help improve the calibration accuracy of other IMU.

To validate this theory, Algorithm 1 will be used, with calibration parameter of one of the IMUs set to ground truth, while only the calibration parameter of the other IMU will be estimated. Both angular velocity constraint 2-1 and acceleration constraint 2-3 will be tested separately to assess their performance in stiff case. Prediction errors are presented in the Table 3-2.

Mean Error($\angle(j_i, \hat{j}_i)$), $i = 1,2$	**Angular Velocity Constraint**	**Acceleration Constraint**
j_1 known, estimate j_2	3.55°	72.46°
j_2 known, estimate j_1	0.36°	11.95°

Table 3-2: Prediction error in case one of IMU has been calibrated

For angular velocity constraint, estimation error is less than 4 degree, demonstrating that calibration results of two IMUs can indeed compensate for each other. However, for acceleration constraint, the estimation error remains relatively large. This is likely because the acceleration constraint 2-3 is a conditional approximation, and when the acceleration caused by rotation around the joint center becomes significant, the constraint breaks down. As a result, using acceleration constraint alone cannot achieve stable calibration.

Based on the prediction error results in Table 3-2, the angular velocity constraint 2-1 is shown to potentially provide useful information for I2S calibration in the stiff case. In the subsequent simulations, the integration of the angular velocity constraint with deep learning will be explored.

To quantify the extent to which the predicted joint axis satisfy kinematic constraints, a constraint loss metric for angular velocity constraint 2-1 is defined as the average loss computed over l time frames.

$$\text{Loss}_{\text{con}} = \frac{1}{l}\sum_{t=1}^{l}(\left|\left|\omega_{1,t} \times \hat{j}_1\right|\right|_2 - \left|\left|\omega_{2,t} \times \hat{j}_2\right|\right|_2) \tag{3-8}$$

where $\hat{j}_1$ and $\hat{j}_2$ can be calculated by predicted I2S relative angle $\hat{\alpha}^i_{I2S}$ and j_0 with equation 3-9.

$$j_i = \begin{bmatrix} cos(\alpha^i_{\text{I2S}}) & 0 & sin(\alpha^i_{\text{I2S}}) \\ 0 & 1 & 0 \\ -sin(\alpha^i_{\text{I2S}}) & 0 & cos(\alpha^i_{\text{I2S}}) \end{bmatrix}^{\mathsf{T}} j_0, \text{ for } i = 1,2 \tag{3-9}$$

Theoretically, if we substitute the ground truth of I2S angles α^i_{I2S} and IMU measurements ω_i, a_i into the loss function 3-8, the resulting value should be 0. However, due to sensor noise and other disturbances, Loss_{con} will still have a certain magnitude. A sample of 1000 random sequences was generated, then its ground truth I2S angle and measurements were substituted into the loss function. The distribution of Loss_{con} are shown in Figure 3-2. In case of perfectly accurate I2S calibration, the value of Loss_{con} fluctuates within a small range around 1.1e-3.

Figure 3-3 illustrates the relationship between error of α^2_{I2S} and Loss_{con} when α^1_{I2S} is set to ground truth. As shown in the figure, the further α^2_{I2S} deviates from its ground truth, the larger the loss becomes, reaching its maximum when error is 90 degrees. This result suggests that it is possible to reduce the prediction error of α_{I2S} by minimizing Loss_{con} in deep learning training.

This constraint analysis part confirms the validity of integrating the angular velocity constraint 2-1 into the deep learning method, and it verifies the possibility of achieving more accurate I2S calibration by minimizing its constraint loss Loss_{con}.

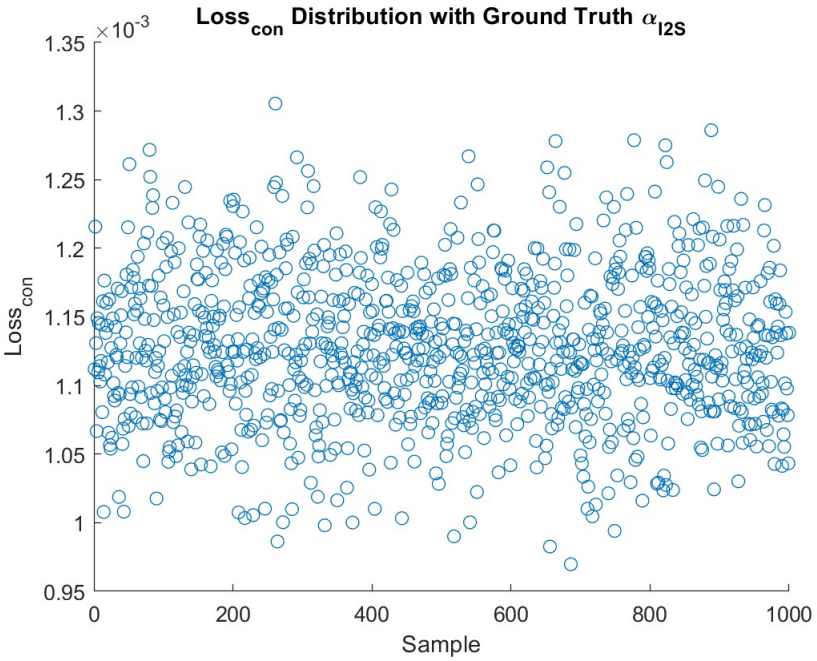


Figure 3-2: Constraint loss Loss_{con} distribution with ground truth α^{i}_{I2S}

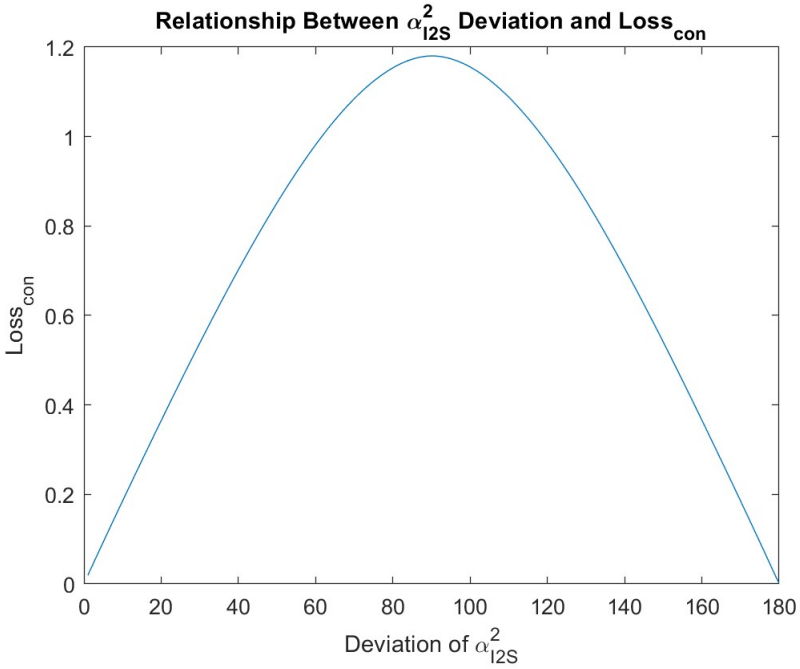


Figure 3-3: Relationship between α^{2}_{I2S} and Loss_{con}

3-4 Deep Learning Method

3-4-1 Single IMU

The deep learning network structure used in this thesis will be based on the network architecture in [55]. Building on that foundation, modifications will be made to better suit the specific scenarios addressed in this thesis.

The input to the deep learning model is a sliding window with length l, containing gyroscope and accelerometer measurements for l frames: $\{\omega_{i,t}, a_{i,t}\}_{t=1}^{l}, i = 1,2$. Each input has a shape of $(l,6)$, where each frame consists of 3D gyroscope and 3D accelerometer data. The main building blocks of the deep learning network include three Convolutional Neural Network (CNN) layers, followed by two Recurrent Neural Network (RNN) layers, and ending with three Fully Connected (FC) layers. The output of the network is the I2S calibration result, as well as the I2S angle: $\alpha_{I2S}^{i}, i = 1,2$. The structure of deep learning network is shown in Figure 3-4.

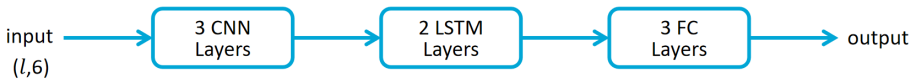


Figure 3-4: Single IMU deep learning network structure

In the deep learning network architecture, three parts play distinct roles in processing and learning from IMU data for I2S calibration:

1. CNN Layers: The convolutional layers are responsible for capturing local spatial patterns in the sensor data [1]. They apply convolution along sensor channel dimension, and filters will learn correlations between those 6 channels (3 for gyroscope and 3 for acceleromter).

2. RNN Layers: Two LSTM layers are used to capture temporal dependencies and long-term relationships in motion data across time. It can process time-series data sequentially, maintaining information from previous frames to understand how the motion evolves over time [54]. This is crucial for deep learning network to understand the dynamic nature of segment's motion.

3. FC Layers: The fully connected layers are the final part of the network, integrating the learned features and temporal relationships to produce the final output. It combine all the learned information to perform the final regression task, transforming the learned features into the target I2S calibration output.

Since the input to the deep learning network is a sliding window, the sliding window technique is applied to extract sequences with length l from motion sequence, with a stride of $\frac{l}{2}$. This ensures that there is a certain correlation between data within consecutive windows, allowing the network to reinforce its learning iteratively. In [55], l was set to 128, as a window of 128 frames roughly encompasses a complete walking cycle. However, in this thesis, we are not specifically learning walking data, so alternative values for l can be considered. Additionally, selecting a smaller l can reduce the size of individual input, which means that I2S calibration can be achieved using less motion

data. In this part of the simulation, the deep learning model will be trained using different window lengths to determine an appropriate choice.

In many deep learning training scenarios, input data needs to be normalized, which helps the model generalize better and adapt to different situations [3]. However, whether normalization is necessary in this thesis is a consideration. First, the purpose of normalization is to allow the model to generalize better across various situations. In this thesis, the training set consists of motion data generated in a simulated environment, which already includes data on movements of human at different speeds. Thus, further generalization in terms of magnitude of sensor data is unnecessary. Secondly, the input data consists of sensor measurements, and their magnitude reflects important information, such as the distribution of angular velocity and acceleration across the three coordinate axes, which relates to the relative orientation between IMU and segment. Normalizing the data would lead to a loss of this information. Therefore, in the simulation, training and validation will be conducted for both normalized and unnormalized training sets to determine whether normalization is necessary.

In [55], Mean Squared Error (MSE) was chosen as the loss function for training the model. In deep learning, MSE is a commonly used loss function. It measures the prediction error of the model by calculating the squared difference between the predicted values and the actual values, and then averaging the results [51]. Its formula is given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3-10}$$

where y_i is the ground truth, $\hat{y}_i$ is the predicted value and n is the sample number.

In this thesis, I2S angles α_{I2S}^i are chosen as output of deep learning model, and angles are periodic parameters, where 0 degrees and 360 degrees represent the same position. Therefore, the original loss function may fail to correctly handle this periodicity, especially when the angles are close to the boundary. To address this issue, the output labels are transformed from angles to sine and cosine values. This approach converts a periodic scalar into two values defined in a 2D plane, allowing the model to naturally capture the periodic relationship between angles. The sine and cosine values can be converted back into angles afterwards. The loss function is given by:

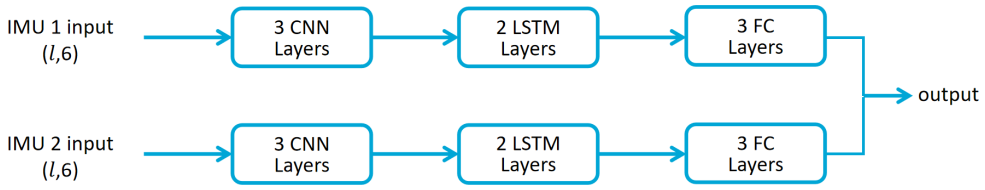
$$\text{Loss}_{\text{MSE}} = (\hat{y}_{cos}^i - y_{cos}^i)^2 + (\hat{y}_{sin}^i - y_{sin}^i)^2 \ \text{ for } i = 1,2 \tag{3-11}$$

where y_{cos}^i and y_{sin}^i are the cosine and sine values for I2S angles α_{I2S}^i and parameters with hat are the predicted output.

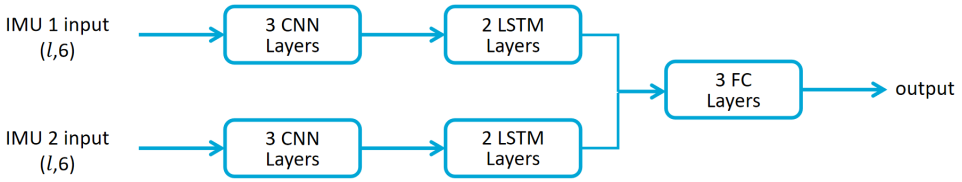
3-4-2 Extension to Dual IMU

After the implementation of deep learning approach for I2S calibration with a single IMU, an intuitive idea for deep learning network structure involving two IMUs is to combine the individual deep learning networks for each single IMU into one unified architecture. The key question to explore is where to merge the networks for two IMUs. In this part of the simulation, four different network structure will be studied, each representing a distinct merging strategy. The results will be analyzed and compared to select the appropriate network architecture.

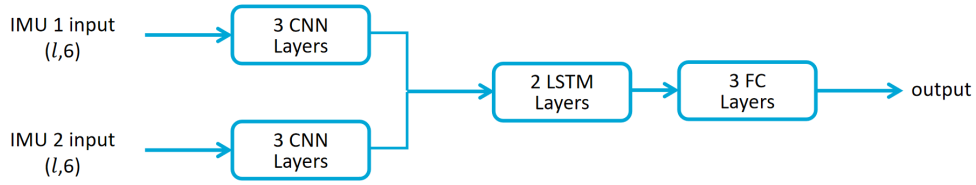
The four network structures are illustrated in Figure 3-5.



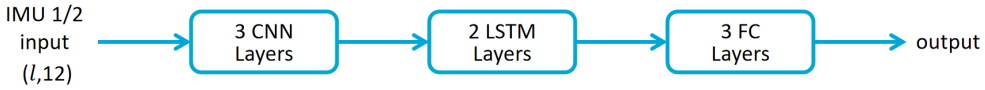
((a)) Model 1



((b)) Model 2



((c)) Model 3



((d)) Model 4

Figure 3-5: Dual IMU deep learning network structure

In the first structure, the networks for two individual IMUs remain completely separate until after predicted values are generated, at which point they merge. In the second structure, two networks merge before the FC layers. As for the third structure, networks merge before the RNN layers. In the fourth structure, two networks begin to merge from the very start of the architecture.

3-4-3 Integration of Constraint into Deep Learning Model

To integrate kinematic constraints into the deep learning model, the chosen approach is to incorporate the constraint as a penalty term in the loss function. By incorporating physical constraints into the loss function, the constraints can directly influence the gradient computation and parameter updates of deep learning model. This allows the model's optimization objective to not only account for data fitting but also adhere to physical laws, enabling the model to naturally learn outputs that are more physically consistent during the training process [7]. Based on the previous analysis in section 3-3, this thesis utilizes the angular velocity constraint 2-1 to integrate with deep learning. The loss associated with this constraint, which quantifies the degree to which the predicted outputs conform to the constraint, has been defined in 3-8.

For model with dual-IMU, the MSE loss is defined as:

$$\text{Loss}_{\text{MSE}}^{\text{dual}} = \frac{1}{4} \sum_{i=1,2} \left[(\hat{y}_{cos}^{i} - y_{cos}^{i})^2 + (\hat{y}_{sin}^{i} - y_{sin}^{i})^2 \right] \tag{3-12}$$

Adding two parts of loss 3-8 and 3-12, the loss function of deep learning model can be expressed as:

$$\text{Loss} = \text{Loss}_{\text{MSE}}^{\text{dual}} + w_{\text{con}} * \text{Loss}_{\text{con}} \tag{3-13}$$

where w_{con} is the weight of constraint loss.

The deep learning model will be trained with different w_{con} to find the optimal balance between $\text{Loss}_{\text{MSE}}^{\text{dual}}$ and Loss_{con}. In dual-IMU experiment, the average $\text{Loss}_{\text{MSE}}^{\text{dual}}$ at the convergence of the model was 0.012 in average and Loss_{con} was 0.118. Therefore, the baseline for w_{con} in the experiment combining deep learning with constraints is set as $0.012/0.118 \approx 0.1$. When w_{con} is set to 0.1, the theoretical losses from both components are equal, suggesting that they contribute similarly to model's convergence. In simulation, a range of other w_{con} values around this baseline will be tested to explore the balance between two loss components.

Chapter 4

Simulation Setup

This chapter presents the construction of the simulated model, the dataset generation, and the detailed setup for each part of the simulation.

4-1 Simulated Model

Due to the challenges associated with collecting real-world data involving two human body segments without relative motion, this thesis relies solely on simulated data for both model training and evaluation. This approach allows for consistent and repeatable data generation, facilitating the development and testing of the calibration method under idealized conditions.

4-1-1 Model Overview

The simulated model is built using blocks in *Matlab-Simulink-Simscape-Multibody*, mainly because this module in *Simulink* provides an intuitive and accurate approach to physical modeling for systems composed of multiple rigid segments and joints. It can precisely simulate the interactions and kinematics of each component, and it also allows for the input of custom defined joint motion data to simulate various types of movement. Additionally, it can provide dynamic visualization, which can display the motion of different segments in real time. These capabilities align perfectly with the requirements of this thesis for building simulated model and generating a large amount of motion dataset.

The simulated model built in *Simulink* includes modules for two segments (s_1 and s_2) and two IMUs. The segment s_1 has its endpoint, which is farthest from the joint, set to remain relatively stationary with respect to the global frame. The connection point is a joint with two degrees of rotational freedom, allowing rotation around a vertical axis and an axis parallel to the joint axis. The reason for this setting is that fixing one end in the world coordinate system reduces the number of variables that need to be considered in the model. This fixed point provides a stable reference, allowing the rest of the model to effectively perform motion modeling and analysis around it. While s_1 and s_2 modules are

connected by a hinge joint module. Two IMU modules are each attached to their respective segment modules. The overall *Simulink* architecture is illustrated in Figure 4-1.

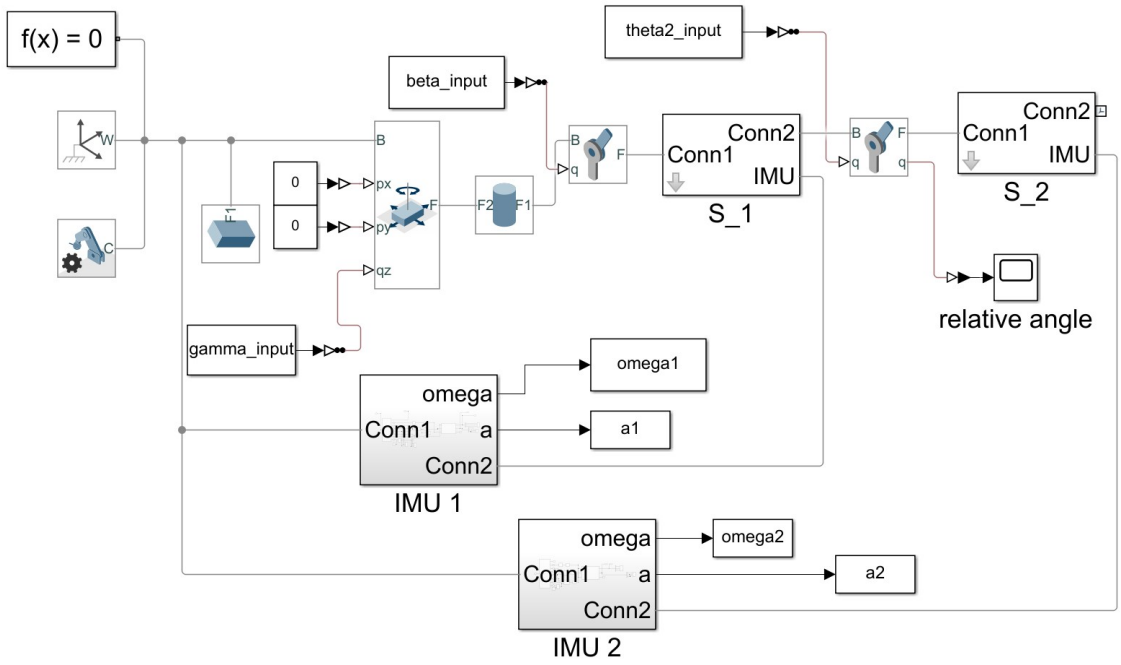


Figure 4-1: Model overview in Simulink

The visualization of simulated model is shown in figure 4-2.

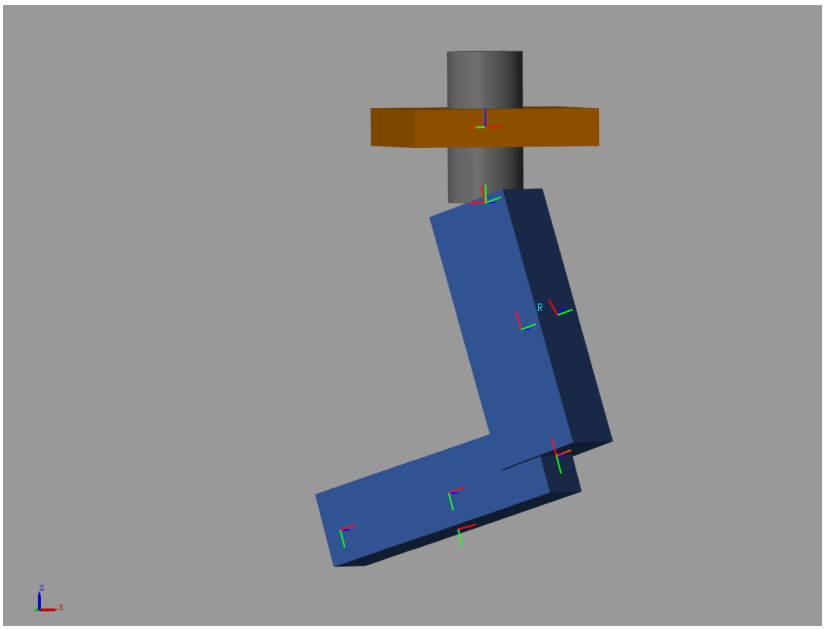


Figure 4-2: Model visualization

4-1-2 Segment Model

The segment model use the *Brick Solid* block in *Simulink-Simscape-Multibody-Body Elements*.

As described in section 3-1, there are several variable parameters within segment module, aimed at creating diverse motion sequences (including different segment lengths and varying angles between the two segments and so on). This diversity in movement is intended to provide the deep learning method with a varied dataset for training, enabling its application across different individuals or various scenarios.

The variable parameters related to the simulated segment model includes β_{s_1,s_2}, θ_{proj}, ϕ_{ver}, l_1 and l_2, as described in Table 3-1. In *Simulink* model, multiple input ports have been established to facilitate the reception of these external parameters. The values of these parameters are defined within *MATLAB* workspace. Table 4-1 shows how these parameters connect with modules in *Simulink*.

Input Parameter	Connected Module in Simulink
β_{s_1,s_2}	*Simulink-Simscape-Multibody-Joints-Revolute Joint-Actuation Motion-Provided by input*
θ_{proj}	*Simulink-Simscape-Multibody-Joints-Revolute Joint-Actuation Motion-Provided by input*
ϕ_{ver}	*Simulink-Simscape-Multibody-Joints-Planar Joint-Actuation Motion-Provided by input(x-axis)*
α^1_{I2S}, α^2_{I2S}	*Simulink-Simscape-Multibody-Frames and Transforms-Rigid Transform-Rotation*
l_1, l_2	*Simulink-Simscape-Multibody-Body Elements-Brick Solid-Properties-Geometry*

Table 4-1: Input parameters' connected module in Simulink

Other invariable *Properties* in two segment blocks are set as: Width: 12 *cm*, Height: 10 *cm*, Density: 1060 kg/m^3, which also reference the average values of thigh and shank in human body [19].

4-1-3 IMU Model

Two IMU modules are each attached to their respective segment modules, with origin of IMU frame fixed to the center point of forward surface of segment block, as defined in section 3-1. The structure of IMU module is shown in Figure 4-3.

In IMU module, during movement, the *Transform Sensor* block(*Simulink–Simscape–Multibody–Frames and Transforms–Transform Sensor*) is first used to obtain the relative orientation and relative acceleration between I_i and G. This data is then discretized using the *Zero-Order Hold* block(*Simulink–Discrete–Zero Order Hold*) before being fed into the *IMU* block(*Simulink–Sensor Fusion and Tracking Toolbox–Multisensor Positioning–Sensor Models–IMU*), which outputs the corresponding IMU measurements of angular velocity and acceleration. The output frequency of the IMU is set to 50Hz, which ensures accuracy while maintaining computational efficiency [18].

To make simulated IMU measurements more realistic, noise values are added to the *IMU* block based on the noise parameters of the Xsens MTi-100 IMU, as shown in Table 4-2 [56].

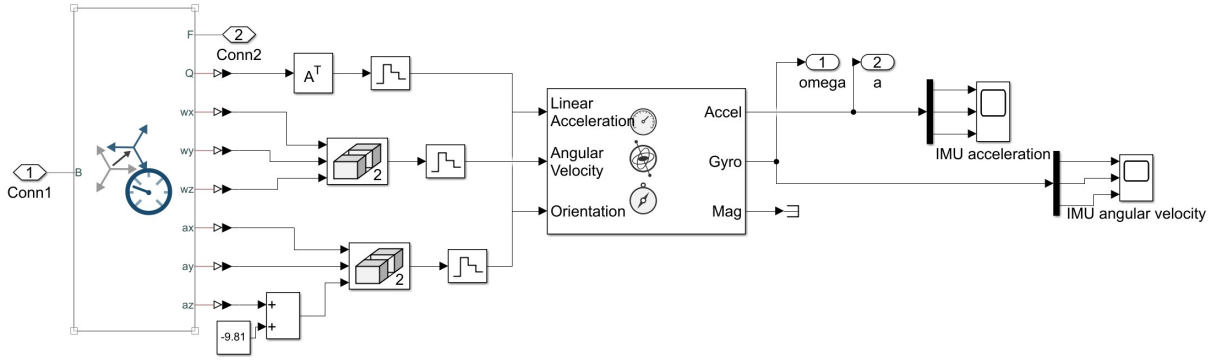


Figure 4-3: IMU model in Simulink

Noise Setting		Noise Parameters
Accelerometer	**Velocity Random Walk** $((m/s^2)/\sqrt{Hz})$	$[6\cdot10^{-4}, 6\cdot10^{-4}, 6\cdot10^{-4}]$
	Bias Instability (m/s^2)	$[2\cdot10^{-4}, 2\cdot10^{-4}, 2\cdot10^{-4}]$
Gyroscope	**Angle Random Walk** $((rad/s)/\sqrt{Hz})$	$[2\cdot10^{-4}, 2\cdot10^{-4}, 2\cdot10^{-4}]$
	Bias Instability (rad/s)	$[5\cdot10^{-5}, 5\cdot10^{-5}, 5\cdot10^{-5}]$

Table 4-2: Noise setting in IMU module

For every simulation, the randomization seed of *IMU* block is set to a random integer between $[0, 10^6]$, which ensures the noise added in each simulation is different. Other parameters in *IMU* block, such as environmental temperature and magnetic field conditions, are kept at their default values.

4-2 Dataset Creation

Deep learning methods require large datasets for model training and validation. After building the simulated model in *Simulink*, different motion data can be obtained by inputting various parameter values and running simulations.

To generate a sufficient dataset for training and validating the deep learning model, this study created numerous motion sequences, each with different parameter settings to simulate various motion scenarios. In this study, a single motion sequence can be seen as an independent scenario, where different parameters are configured uniquely, leading to distinct motion patterns and data characteristics. Each motion sequence lasts for 5 seconds, with a sampling frequency of 50Hz, resulting in a total of 250 frames.

In each motion sequence, there are parameters varying over time, defining the motion of the segment. There are also parameters that remain constant throughout the movement, such as relative I2S angle

and the segment length. Since this thesis aims to explore how to perform I2S calibration in a scenario where there is no relative rotation between the two segments, the relative angle between two segments(β_{s_1,s_2}) is also set to be constant. For each generated motion sequence, the main parameters are listed in Table 4-3.

Duration	5(s)	
Sample Rate	50(Hz)	
Parameters	**Variation Type**	**Parameters**
	constant	$\alpha_{\text{I2S}}^{1,2}$
		β_{s_1,s_2}
		$l_{1,2}$
	time-varing	θ_{proj}
		ϕ_{ver}

Table 4-3: Parameters for single motion sequence

By adjusting these parameters across multiple motion sequences, diverse motion data can be generated for deep learning model training. When generating the dataset, for parameters that remain constant (α_{I2S}^1, α_{I2S}^2, β_{s_1,s_2}, l_1 and l_2) within a single motion sequence, a random value is chosen within the defined range of each parameter for each generated sequence.

For θ_{proj} and ϕ_{ver} that vary over time within a motion sequence, their input frequency is also set to 50Hz. To make segment movements more realistic, this thesis introduces a random sample smooth function for these two angles. Specifically, an initial point is randomly selected within the defined range of θ_{proj} and ϕ_{ver}. At each step, a value for angle change is randomly generated based on a normal distribution with a predefined standard deviation (denoted as σ_θ and σ_ϕ). The value for the next sample is then obtained by adding this angle change to the previous sample value.

The values of two standard deviations(σ_θ and σ_ϕ) remain constant within a single motion sequence and are randomly generated before every simulation. When σ is small, it results in motion sequences that contain slow movements, conversely, a larger deviation value produces motion sequences that include fast movements. During human motion, the acceleration of thigh and shank is generally within the range of 30 m/s^2, while the angular velocity is below 10 rad/s [48]. The range of values for these two σ parameters is also informed by the acceleration and angular velocity range observed in human leg movements. σ_θ is randomly selected within $[0, 15]$ and σ_θ is chosen within $[0, 10]$ for every motion sequence.

The resulting sequence is further smoothed using a moving average method to produce smoother movement trajectories. The use of a moving average method helps eliminate abrupt changes, making the movement appear more natural and consistent with the typical smoothness of human motion. Figure 4-4 shows five example of randomly generated θ_{proj} sequences.

These motion sequences cover various potential movement conditions, aiming to enable deep learning method to learn a broad range of motion characteristics, thereby improving its generalizability across different individuals and application scenarios. It also makes the simulated data more similar to real-world experimental data. This is beneficial for extending this method to real data in future work.

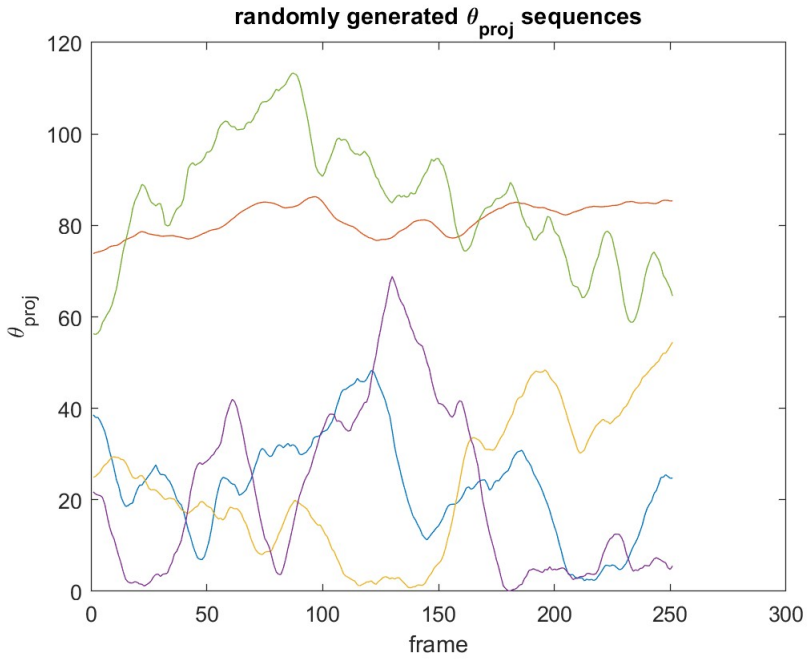


Figure 4-4: Five examples of randomly generated θ_{proj} sequences

4-3 Setup Details

As described in section 1-3, the process of the simulation is summarized in Table 4-4.

Simulation		Object
Constraint-based		prove inapplicability of constraint-based method
Deep learning	**Single IMU**	demonstrate that deep learning method can be effectively applied in stiff case
	Dual IMU	determine the appropriate dual IMU network structure
	Integrated with constraint	explore whether incorporating kinematic constraints can improve the accuracy of I2S calibration

Table 4-4: Experiment Summary

4-3-1 Constraint-based Method

For experiment which proves inapplicability of constraint-based method, to avoid any results being coincidental and to ensure more generalizable outcomes, 100 different motion sequences will be randomly selected according to Table 4-3. Each sequence will be processed using Algorithm 1, with a random initial point selected. Then the predicted joint axes $\hat{j}_1$ and $\hat{j}_2$ of each sequence will be obtained.

According to [32], w_ω and w_a were chosen as 10 and 1, respectively, and the tolerance Ψ_{tol} was set to 0.0001.

4-3-2 Deep Learning Method

For all deep learning training in this thesis, 3,000 motion sequences defined in Table 4-3 are randomly generated through simulation as the training set and validation set. After preprocessing the dataset, all data samples will be shuffled, and 30% of the dataset will be taken as validation set during deep learning training process, while remaining data will be used as the training set.

As evaluation set, 1000 random motion sequences, as defined in Table 4-3, are generated. After applying the same preprocessing as the training set, these data are input into the model to obtain the predicted sine and cosine values of angles. These values are then converted into the predicted I2S angle $\hat{\alpha}^i_{I2S}$, which are finally compared with true I2S angle α^i_{I2S}.

In single IMU part, the deep learning model will be trained using different window lengths. l will be set to 16, 32, 64, and 128. Also training will be done separately using normalized and unnormalized sets to determine if normalization is needed. In dual IMU part, training will be conducted using four different dual-IMU network architectures. In constraint integration part, the deep learning model will be trained with different w_{con}. The selected values for w_{con} are: 1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2, 0.1, 0.5, 1, 5, 10, 50, 100.

Each deep learning simulation will be repeated three times to reduce the impact of randomness in results. For each iteration, different training and test sets will be generated, and the final results will be averaged. The final average results will be presented in the Simulation Results section.

4-4 Deep Learning Configuration

In this thesis, TensorFlow, an open-source machine learning framework, is utilized to implement the deep learning model in section 3-4. TensorFlow is one of the most widely used deep learning frameworks in academia and industry. We can construct the neural network components (CNN, RNN, FC layers) with intuitive, modular blocks by using Keras Application Programming Interface (API) of TensorFlow [10].

Hyper Parameters

The hyper parameters of deep learning model are shown in Table 4-5.

Optimizer

The *Adam* optimizer is chosen for training due to its effectiveness in handling sparse gradients and its adaptive learning rate capabilities [21]. The learning rate is set to 0.001, a commonly used value that provides a balance between training speed and stability, ensuring that the model can learn effectively without overshooting the optimal parameters [55].

Optimizer	Batch size	512
	Learning rate	0.001
	Optimizer	Adam
CNN Layer	Layer	3
	Kernel size	4×1
	Kernel	16
RNN Layer	Layer	2
	Cell	LSTM
	Units	32
FC Layer	Layer	3
	Nodes	1:64, 2:32, 3:32
Loss Function	-	MSE

Table 4-5: Hyper Parameters

Early Stopping

To prevent overfitting and save computational resources, the early stopping regularization method is also used, with the patience parameter set to 50 [53]. Setting the patience to 50 allows the model to tolerate some fluctuations in validation performance before stopping training.

Computational Resource

The training was performed on Nvidia GPUs of DelftBlue High Performance Computer (HPC) [9], with an average training time of approximately 2 hours per model.

4-5 Evaluation Metric

Constraint-based Method

The evaluation metrics for this part of the experiment will be mean and variance of the angular difference between predicted joint axis ($\hat{j}_1$ and $\hat{j}_2$) and the ground truth joint axis (j_1 and j_2). These two metrics are chosen for the following reasons:

1. Mean Absolute Error (MAE): This metric represents the average error in the alignment of the predicted joint axis compared to the true joint axis. It provides a clear indication of how well the calibration method performs overall in aligning the two axes.

2. Variance of the angular difference: This metric captures the consistency of the algorithm's performance across different motion sequences. A lower variance suggests that the algorithm produces reliable and consistent results, while a higher variance indicates that the performance fluctuates significantly between different motion sequences.

The ground truth joint axis can be calculated as in equation 3-9. After obtaining predicted joint axis and ground truth joint axis, two evaluation metric can be calculated as follows:

$$\text{MAE}_i = \frac{1}{N}\sum_{k=1}^{N} \angle(j_i, \hat{j}_i), \text{ for } i = 1,2 \tag{4-1}$$

$$\text{Var}_i = \frac{1}{N}\sum_{k=1}^{N} \left(\angle(j_i, \hat{j}_i) - \text{MAE}_i\right)^2, \text{ for } i = 1,2 \tag{4-2}$$

Deep Learning Method

MAE is also used as the evaluation metric in deep learning method. It is important to note that since the output of deep learning model, I2S angle, is a periodic parameter, there are two possible ways to calculate the absolute error: the absolute value of the difference, or the value of 360 degree minus the absolute difference. The smaller of the two is taken as the absolute error.

$$\text{AE}_{i,k} = \min(|\hat{\alpha}^i_{I2S} - \alpha^i_{I2S}|, 360 - |\hat{\alpha}^i_{I2S} - \alpha^i_{I2S}|), \text{ for } i = 1,2 \text{ and } k = 1...N$$

$$\text{MAE}_i = \frac{1}{N}\sum_{k=1}^{N} \text{AE}_{i,k}, \text{ for } i = 1,2 \tag{4-3}$$

The median value of $\text{AE}_{i,k}$ is also used as one of the metrics, as it is less sensitive to outliers compared to the mean error. By combining mean and median errors, a more comprehensive evaluation of the model's performance can be achieved, including insights into the model's stability and error distribution.

For deep learning method with two IMUs, Loss_{con} defined in 3-8 is also introduced. This metric quantifies the extent to which the predicted I2S angles satisfy kinematic constraints over l frames. By analyzing this metric alongside other metrics, it becomes possible to figure out whether incorporating constraints into deep learning can enhance the model performance by adjusting the constraint loss. Another metric is the overall average error of the two IMUs calculated by $\frac{1}{2}(\text{MAE}_1 + \text{MAE}_2)$, which provides a comprehensive evaluation of the model's calibration accuracy by considering its performance for both IMUs collectively.

Chapter 5

Simulation Results

This chapter presents results of simulation, followed by a comparison and analysis of simulation results.

5-1 Constraint-based Method

The results of constraint-based method simulation are presented in Table 5-1.

Evaluation metric	IMU1	IMU2
MAE($^\circ$)	73.96	72.70
Error Variance($^{\circ 2}$)	400.1	381.6

Table 5-1: Constraint-based Method

Table 5-1 shows that the mean error and error variance of I2S calibration for both IMUs are relatively large. In the non-stiff case, the prediction error obtained using the same algorithm 1 can be within 2° [32]. This suggests that in cases there is no relative motion between two segments, the constraint-based method struggles to achieve reliable I2S calibration.

Figures 5-1 and 5-2 illustrate the distribution of $\Psi(\alpha^1_{\text{I2S}}, \alpha^2_{\text{I2S}})$ under different estimated values of α^1_{I2S} and α^2_{I2S} in the non-stiff and stiff cases, respectively. The values of $\Psi(\alpha^1_{\text{I2S}}, \alpha^2_{\text{I2S}})$ in the two figures are calculated from two random motion sequences: one allowing significant relative movement of the joint, and the other keeping the joint in a stiff state.

From Figure 5-1, it can be observed that when relative motion exists, $\Psi(\alpha_{\text{I2S}})$ exhibits four distinct minima, all falling below 0.6. These minima correspond to the true joint axis orientations. Since the axis has two possible directions, their combination results in a total of four minima in the mesh plot. However, as shown in Figure 5-2, in the stiff case, there are no distinct troughs in the mesh plot. Many combinations of α^1_{I2S} and α^2_{I2S} result in relatively small $\Psi(\alpha_{\text{I2S}})$. As analyzed in section 3-3, even if the joint axis j_0 is replaced with any arbitrary axis in space, the angular velocity constraint

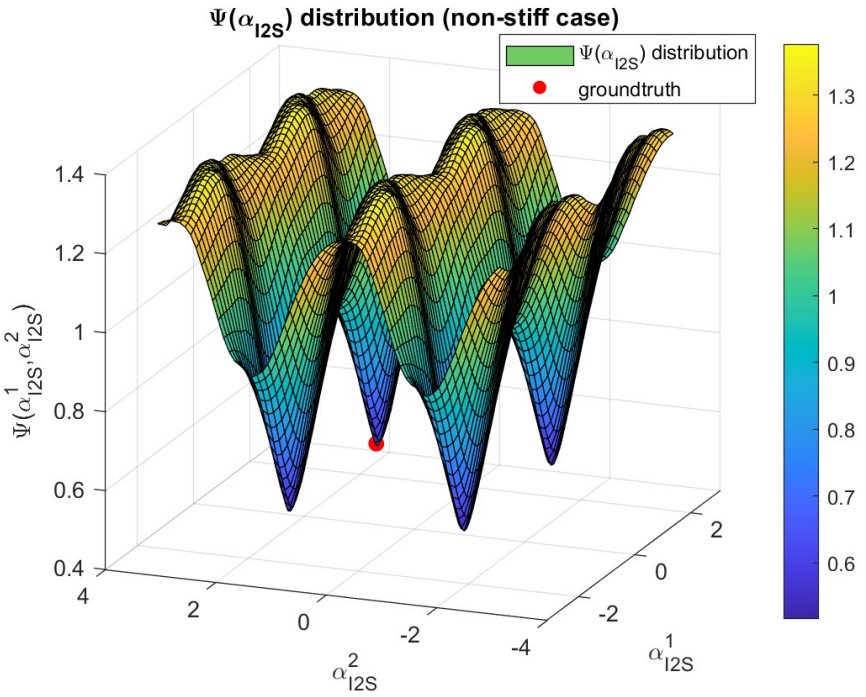


Figure 5-1: $\Psi(\alpha^1_{I2S}, \alpha^2_{I2S})$ distribution in non-stiff case

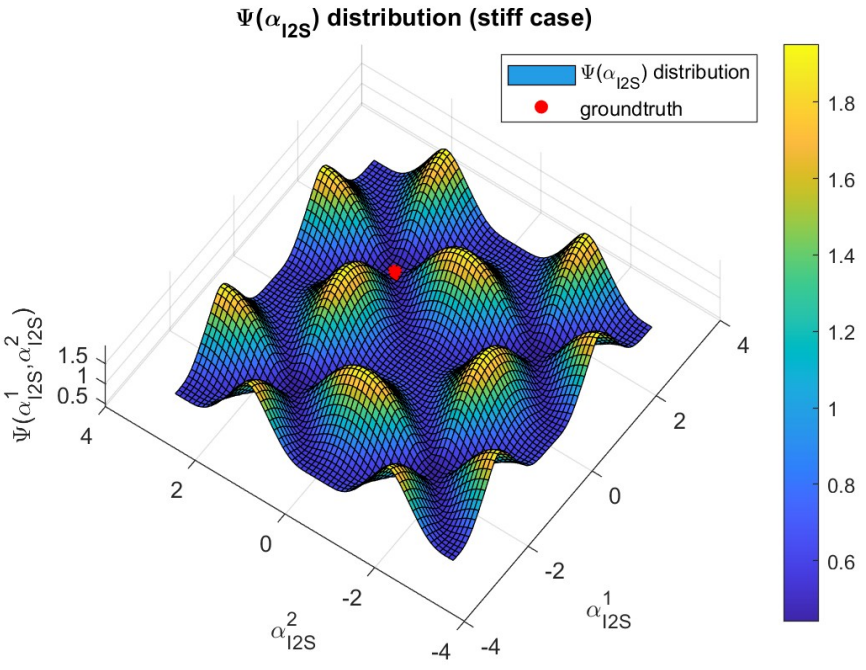


Figure 5-2: $\Psi(\alpha^1_{I2S}, \alpha^2_{I2S})$ distribution in stiff case

2-1 and acceleration constraint 2-3 still hold. This leads to multiple local minima in the optimization process while minimizing the sum of squared residuals $\Psi(\hat{\alpha}^1_{I2S}, \hat{\alpha}^2_{I2S})$ defined in 2-6, since the residual

is defined as the constraint deviation. In the mesh plot, the regions with values below 0.6 correspond to these axes. It is possible that even both I2S angles α^i_{I2S} deviate significantly from the ground truth, if they can derive the same axis using 3-9, the corresponding $\Psi(\alpha_{I2S})$ would represent a relatively small value. Therefore, in this case, the optimization method may achieve other local minima differs from the ground truth, which results in large average prediction error.

In conclusion, the constraint-based method cannot complete reliable I2S calibration in stiff case. Alternative methods are required to achieve more reliable and higher-precision calibration.

5-2 Deep Learning Method

5-2-1 Single IMU

The deep learning experiment for single IMU calibration includes determining whether input data requires normalization before being fed into the model, as well as selecting the appropriate sliding window length for input. The results of these two experiments are shown in Table 5-2 and Table 5-3 respectively. The values before and after the slash represent the result for IMU1 and IMU2.

input data type	**Normalized data**	**Non-normalized data**
MAE(°)	6.01/53.55	1.96/22.23
median error(°)	4.53/37.71	1.47/9.95

Table 5-2: Single IMU - input normalization (IMU1/IMU2)

sliding window length	**16**	**32**	**64**	**128**
MAE(°)	3.55/30.21	1.96/22.23	2.38/24.04	2.34/25.27
median error(°)	2.29/15.55	1.47/9.95	1.92/11.22	1.93/12.18

Table 5-3: Single IMU - optimal sliding window length (IMU1/IMU2)

From Table 5-2, it can be observed that using non-normalized data as input yields smaller mean and median errors compared to normalized data. The reason could be that different α^i_{I2S} correspond to different distributions of ω_i or a_i across three axes of I_i. When α^i_{I2S} differs while the segment performs the same motion, measurements from IMU will show more noticeable differences in magnitude rather than in trend of data. Once the input data is normalized, the differences in magnitude are eliminated, retaining mainly the characteristics of the trend variation. As a result, the performance after input normalization becomes worse. Therefore, in subsequent experiments, the input data will no longer undergo normalization.

Table 5-3 presents simulation results for different input window lengths. With the exception of the group with a length of 16, the results of other simulation groups (32, 64 and 128) are relatively similar. This may be because that when the window length exceeds 32, the amount of motion information contained in a single input becomes relatively saturated for deep learning model. Given similar error results, selecting a shorter input length is preferable, indicating that I2S calibration can be effectively achieved with shorter motion sequences. Consequently, in subsequent experiments, the sliding window length l will be set to 32.

In all experimental groups, the calibration error for IMU2 is significantly larger than that for IMU1. The reason could be that one end of the s_1 is fixed, resulting in relatively simpler motion, whereas the motion of s_2 is more complex. Consequently, the learning for IMU2's motion pattern in deep learning model is more challenging.

When comparing with results of I2S calibration using the constraint-based method in Table 5-1, the errors using deep learning method are evidently much smaller. Unlike constraint-based method, deep learning without constraint does not exploit the kinematic constraints between different IMUs' measurements. Instead, it learns the patterns that exist between sensor channels and across time series under various α^i_{I2S} through the neural network. Therefore, in stiff case, deep learning method can achieve more accurate I2S calibration compared to constraint-based method.

From Table 5-2 and 5-3, it can also be observed that the predicted median error is generally smaller than the mean error, indicating that the model performs well in most cases, but there are some outliers. In such situations, introducing kinematic constraints could provide additional prior knowledge to the model, limiting the solution space to a reasonable range and thus reducing the influence of outliers.

In summary, in deep learning experiment for single IMU calibration, the input data does not need to be normalized, and an appropriate input length is 32. Under these settings, the average errors for I2S calibration of IMU1 and IMU2 using single IMU deep learning method are 1.96° and 22.23°, while the median errors are 1.47° and 9.95°, respectively.

5-2-2 Dual IMU

This part of the experiment investigates different dual-IMU deep learning network architectures without integrating constraint. As illustrated in Figures 3-5, four different structures have been proposed. From Model 1 to Model 4, the degree of coupling between two IMU networks gradually increases. The simulation results for four structures are shown in Table 5-4.

Metric	Model 1	Model 2	Model 3	Model 4
MAE(°)	2.43/22.66	2.63/7.60	4.29/6.35	5.56/5.41
median error(°)	1.70/10.02	2.00/5.05	3.23/4.75	3.77/4.59
Loss_{con}	0.347	0.118	0.146	0.152
average error for 2 IMU(°)	12.55	5.11	5.32	5.49

Table 5-4: Dual IMU

In Model 1, the difference in error between two IMUs is the largest. In Model 1, where two IMU networks are entirely separate and only the outputs are combined, the results are similar to those from single IMU experiments in Table 5-3. As the degree of coupling between two IMU networks increases, the error gap between two IMUs gradually decreases. This is because motion of s_2 is based on s_1, and after two networks are merged, IMU2 can learn more motion information from IMU1's data. The measurements from both IMUs provide information to each other. However, in reality, the motion of s_1 is independent of motion of s_2, meaning that IMU2's data is irrelevant to IMU1. As a result, IMU1's prediction error slightly increases after learning some irrelevant information, while

IMU2's error decreases due to gaining additional information from IMU1. By Model 4, the errors of the two IMUs are almost equal.

As the degree of coupling between the two networks increases, the average calibration error for 2 IMUs first decrease then slightly increase. This may be because the calibration results of each IMU are more closely related to their own measurements and less related to the measurements of the other IMU, which the coupled network is not informed about. When the coupling degree between the two networks becomes too large, it can actually increase the average calibration error due to the learning of more irrelevant features. The trend of Loss_{con} aligns closely with changes in average error. As the average error increases, the deviation from the kinematic constraint also becomes larger, resulting in a higher loss.

Considering results from 4 models, Model 2 has the smallest average error for evaluation set. Therefore, Model 2 is selected as the base dual-IMU network structure for subsequent experiments. Compared to the results in the single IMU experiments in Table 5-3, dual-IMU experiments reduced the error gap between two IMUs and decreased the overall average error for two IMUs. Specifically, for IMU2, which is associated with more complex motion, the mean error decreased from 22.23° to 7.60°, a reduction of nearly 65%, while the median error also decreased, from 9.95° to 5.05°, showing a similar trend in error reduction. Compared to training each IMU independently, jointly training two IMUs achieves a lower average calibration error for 2 IMU, decreasing it from 12.1° to 5.11°. This improvement demonstrates that deep learning method can capture the hidden motion relationship between two segments by jointly training two IMUs.

5-2-3 Integrated with Constraint

This part of the simulation incorporates the kinematic constraint into the deep learning loss function, and different constraint weights w_{con} were selected. The mean and median errors for two IMUs, $Loss_{con}$ and overall average error with different w_{con} are shown in Figure 5-3, 5-4, 5-5 and 5-6 respectively. The case where the constraint weight w_{con} is equal to 0 serves as the control group for the experiment, which represents the scenario where the kinematic constraint is not integrated in the deep learning model. In figures, the control group's values are represented by a red dashed line for easier comparison with the other results. The detailed simulation results are provided in Appendix B.

For IMU1, when the constraint weight w_{con} falls within the range of 0 to 0.5, both the mean and median errors are smaller than those of the model without constraint. For IMU2, the errors are smaller when w_{con} is below 50. Moreover, for the overall average error for the two IMUs, it performs better than the control group within the w_{con} range of 0 to 5. Within these ranges, the errors show a pattern of initially decreasing slowly before gradually increasing. As w_{con} continues to increase, the error exceeds that of the model without constraints and gradually increases. The overall average error for the two IMUs reaches its minimum when w_{con} is 0.05.

The reason for this trend may be that when w_{con} is very small, the kinematic constraint plays a limited role, so results are similar to those without constraint. As w_{con} increases in range of 0 to 0.05, the model gradually reaches a balance between the original loss $\text{Loss}_{\text{MSE}}^{\text{dual}}$ and the constraint loss Loss_{con}. Based on the analysis in Section 3-3, the calibration results of two IMUs in stiff case can complement each other through the constraint 2-1. At this point, the deep learning model can learn motion patterns from IMU measurements while using the constraint to correct predictions that violate it, resulting in lower calibration errors for both IMUs. Within this weight range, it can be observed from Figure

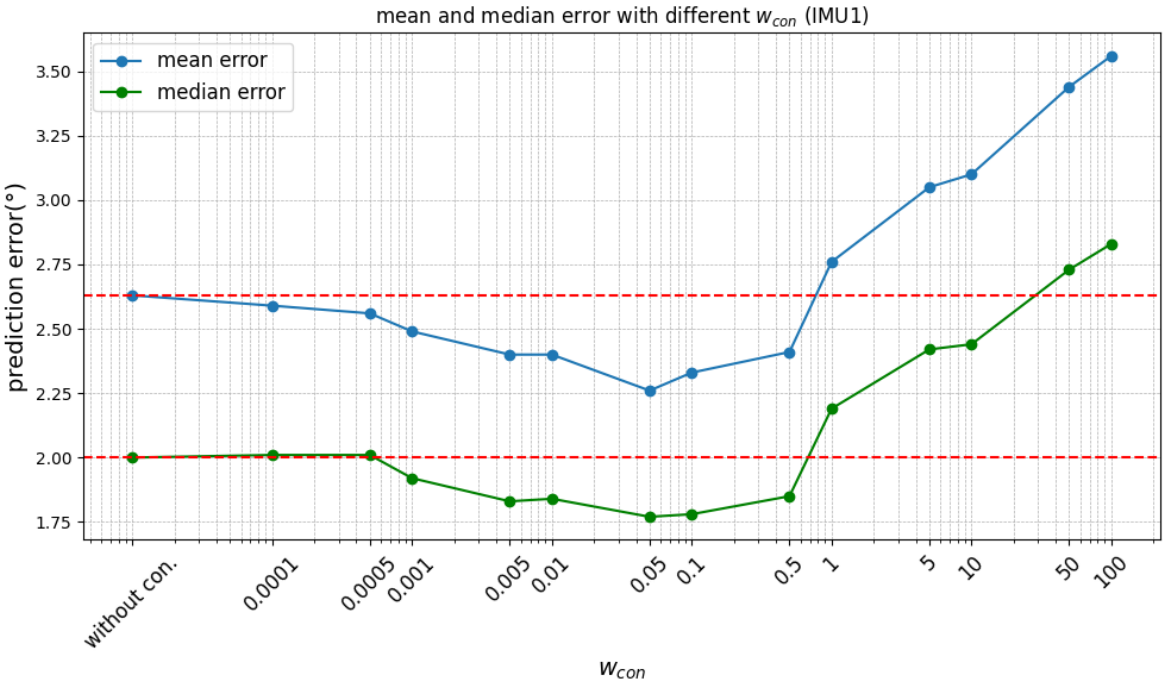


Figure 5-3: Mean and median error with different w_{con} (IMU1)

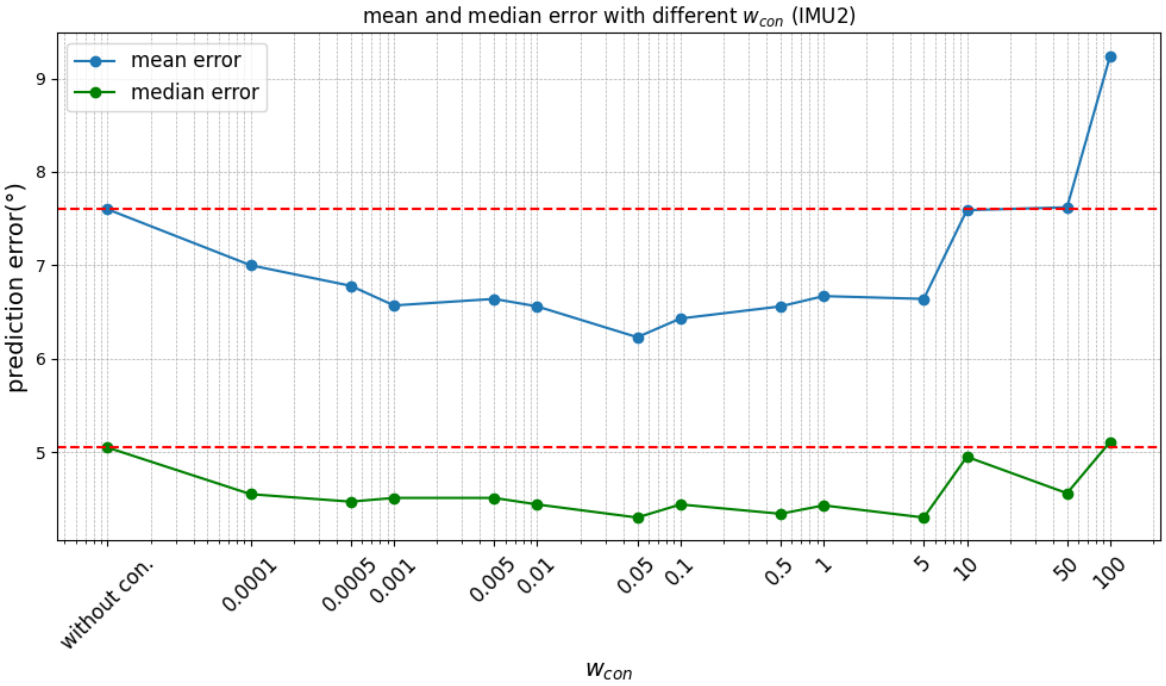


Figure 5-4: Mean and median error with different w_{con} (IMU2)

5-5 that Loss_{con} is also continuously decreasing. Analyzing both factors together, it suggests that the model reduces calibration errors by aligning its prediction more closely with the constraint.

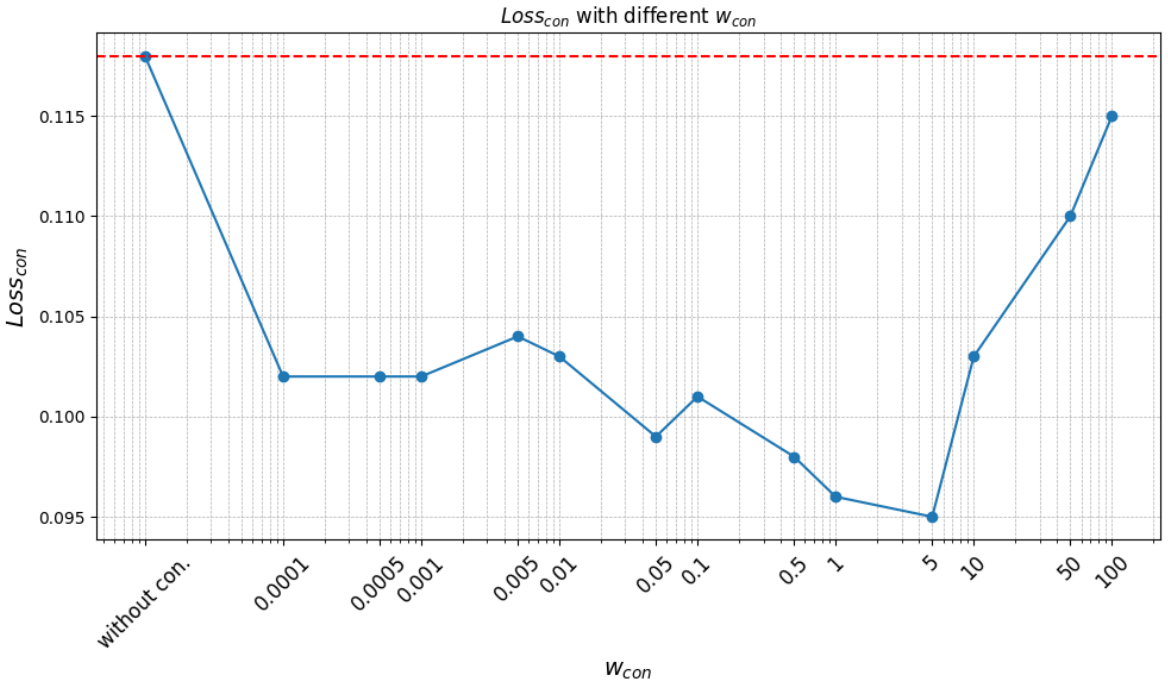


Figure 5-5: $Loss_{\text{con}}$ with different w_{con}

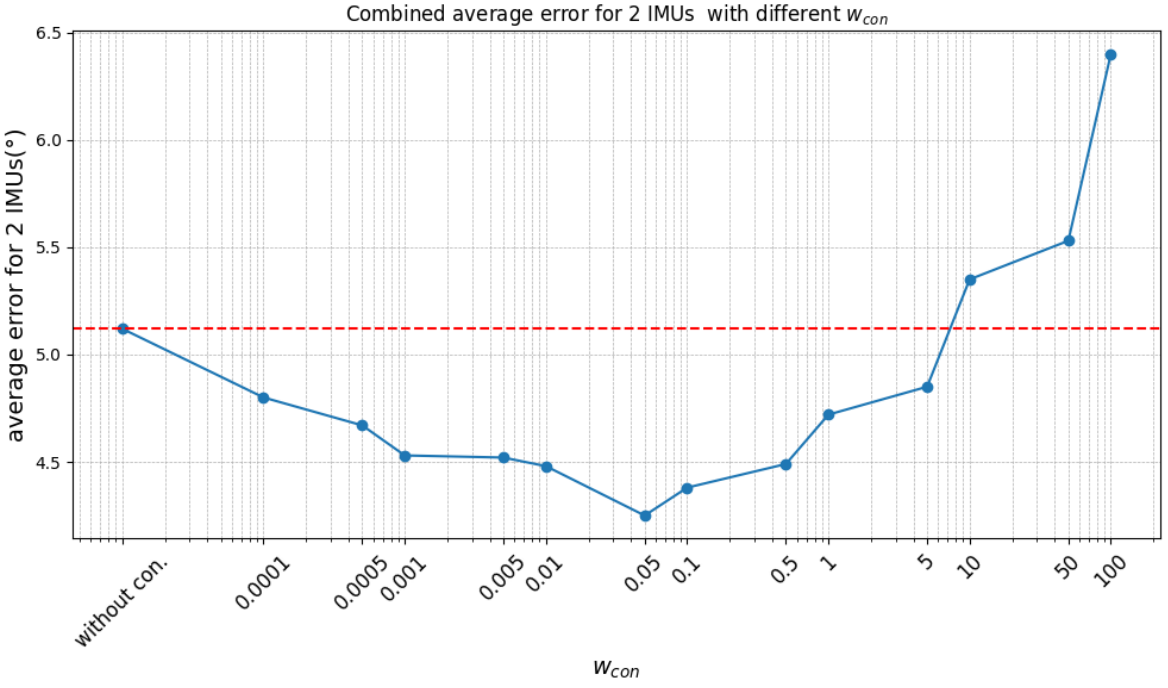


Figure 5-6: Combined average error for 2 IMUs with different w_{con}

When w_{con} exceeds 0.05, the calibration errors for both IMUs begin to increase, eventually surpassing the error of the control group. When w_{con} becomes larger, the model may focus excessively on

minimizing Loss_{con}, potentially compromising its ability to learn the underlying patterns in IMU measurements. As analyzed in Section 5-1, when the joint is stiff and both I2S angles α^1_{I2S} and α^2_{I2S} are unknown, Loss_{con} derived from kinematic constraint can have multiple local minima as shown in Figure 5-2. In this case, due to the reduced relative weight of the original deep learning loss Loss_{MAE}, the model might not be guided toward the ground truth and could ultimately converge to other local minima that far from the ground truth. Within this weight range, Loss_{con} continues to decrease initially, but starts to increase when w_{con} exceeds 5. The reason why Loss_{con} decreases even as the mean error increases may be that, with higher w_{con}, the model becomes more inclined to optimize Loss_{con}, leading to its continuous reduction. However, this focus on optimizing Loss_{con} may cause the model to overlook the motion characteristics in data, resulting in an increase in mean error. When w_{con} becomes too large, Loss_{con} starts to rise, which could be due to the excessive weight causing the optimization path to deviate from the global optimum and settle in a suboptimal region, where the loss is larger. Moreover, the high w_{con} forces the model to strictly satisfy the kinematic constraint, but the increased calibration error at this point may introduce greater deviations when adjusting predictions, ultimately causing Loss_{con} to rise.

Among all weights tested in the experiment, 0.05 is the optimal choice, where the overall average for 2 IMUs reaches the minimum. The inclusion of the constraint with appropriate w_{con} has a greater impact on improving the calibration error for IMU2 compared to IMU1. Additionally, the weight range within which the constraint improves performance for IMU2 is broader than that for IMU1. At the weight of 0.05, IMU1's mean angle error decreased by 0.37° (a reduction of 14%), while IMU2's error decreased by 1.37° ((a reduction of 18%). This may be because IMU2 has more complex motion, resulting in larger prediction errors from deep learning model without constraint, thus providing more room for improvement. As demonstrated in Section 3-3, the calibration results of two IMUs can compensate for each other. In this case, IMU1's more accurate calibration helps guide IMU2 toward a more accurate result.

Compared to the median error of the two IMUs, the mean error showed a greater reduction after incorporating the constraint. The optimal w_{con} reduced the mean calibration error for IMU1 and IMU2 by 14% and 18%, and the median error by 11.5% and 15%, respectively. This difference suggests that the inclusion of the kinematic constraint likely provides stronger prior information, reducing the probability of the model producing extreme outliers. As a result, the reduction in the mean error is more significant.

From the observed impact of w_{con} variations on I2S calibration results, it can be concluded that in stiff case, incorporating the angular velocity constraint 2-1 with appropriate w_{con} (0-5) can improve the accuracy of I2S calibration to some extent. However, the primary driver of model performance remains the original deep learning loss Loss_{MAE}. If w_{con} becomes too large, the model may converge to incorrect local minima, similar to the constraint-based method. Thus, kinematic constraints serve as a supplementary factor, improving the model's performance within a limited range.

Additionally, the w_{con} range identified in this study that enhances I2S calibration accuracy may not be universally applicable. It is likely specific to the particular scenario addressed in this thesis. Determining the suitable weight range for other scenarios would require additional experiments tailored to those specific conditions.

Chapter 6

Conclusion and Future Work

The simulation results demonstrate that traditional constraint-based method results in relatively large errors for I2S calibration when there is no relative motion between two segments. In this stiff case, the deep learning approach used in previous studies for single IMU I2S calibration can be extended to this scenario, achieving significantly lower calibration errors compared to the constraint-based method. The calibration error for IMU2 is higher than that for IMU1. This may be due to differences in the complexity of motion between the two segments.

Joint training of two IMUs enables each IMU network to learn from the other's motion information, reducing the difference in calibration error between two IMUs. An effective dual-IMU model configuration is to couple two single-IMU models after RNN layers. This configuration decreases the calibration error of IMU2, achieving a lower average error of two IMUs. Since motion of lower segment is based on upper segment, joint training of two IMUs can provide IMU2's network with more effective motion information.

Integrating kinematic constraints with appropriate weights into the deep learning model can further enhance I2S calibration accuracy. Analyzing the variation of constraint loss at experiments with different weights reveals that adding the angular velocity constraint 2-1 allows the deep learning model to align its predictions more closely with the constraint, thereby reducing the calibration error. Among all weights tested in the experiment, 0.05 is the optimal choice. Compared to the deep learning model without constraint, this weight reduced the mean calibration error for IMU1 and IMU2 by 14% and 18%, and the median error by 11.5% and 15%, respectively.

In conclusion, in stiff case, the constraint-based method leads to significant errors in I2S calibration. Under such conditions, deep learning is a more reliable and accurate calibration approach. Building on the deep learning method, joint training of two IMUs and integrating kinematic constraints into the loss function provides the deep learning model additional kinematic information between two segments, which helps reduce calibration errors. In this combined calibration approach, deep learning plays a dominant role, while kinematic constraints with an appropriate weight serve as a complementary component, offering a moderate improvement in calibration accuracy.

The limitations of this study include the following:

1. The IMUs in the experiments were constrained to rotate around a single DoF for simplification. To enable practical applications, this approach needs to be extended to three DoF, allowing automatic I2S calibration for IMUs placed in arbitrary orientations.

2. The proposed model is specifically trained for cases where the two segments are completely stiff. Due to the absence of training data that includes scenarios with slight relative motion between the segments or sufficient relative motion, the trained model may not be applicable to these cases.

3. Both the training and evaluation processes in this study rely solely on simulated data. The performance of the proposed model on real motion data remains unknown.

4. In non-stiff case, the constraint-based method achieves calibration errors within 2° [33]. However, the average error for deep learning in stiff case tends to be higher, particularly for IMU2, which has more complex motion patterns.

In future work, the deep learning model could be extended from the completely stiff case to scenarios with slight relative motion between segments. For cases with sufficient relative motion, constraint-based method is already well-established and highly accurate. A potential approach could combine the applicability of both methods: first analyzing the motion data to determine whether sufficient relative motion exists, and then automatically selecting either the deep learning method or the constraint-based method to perform I2S calibration. This I2S calibration approach could also be extended from simulated data to real motion data for application in practical scenarios.

Additionally, integrating various types of kinematic constraints, such as acceleration constraints, velocity constraints and constraints related to motion range, into deep learning model and different methods of incorporating these constraints could be explored. By incorporating different combinations of constraints with varying weights, it would be possible to compare and analyze the contributions of each constraint to I2S calibration process.

This thesis uses a model where two segments are connected by a hinge joint. In the future, this method could be extended to other human body models that include joints with different DoF and additional segments.

Appendix A

Code

A-1 Code for Constraint-based Method

```
% matlab
% load data
data = load('GN_stiff.mat');
angle_error = zeros(100,2);
for k = 1:100
% two IMU measurements(angular velocity and acceleration)
omega1 = reshape(data.dataset(k,:,1:3),251,3);
omega2 = reshape(data.dataset(k,:,7:9),251,3);
a1 = reshape(data.dataset(k,:,4:6),251,3);
a2 = reshape(data.dataset(k,:,10:12),251,3);
% groundtruth output
angle_output = data.output(k,:);

mini_loss = 100;
for m = 1:300

% initial guess(random value)
params0 = rand([4,1])*2*pi-pi;

% set max iteration step and tolerance
max_iter = 100;
tol = 1e-7;

% calculate true joint axis
j1_true=[-sind(angle_output(1));0;cosd(angle_output(1))];
j2_true=[-sind(angle_output(2));0;cosd(angle_output(2))];

% use Gauss-Newton method
[params1, cost1,alpha1] = gaussNewton(omega1,omega2,a1,a2, params0,
    max_iter, tol);
```

```
30 params2_0 = [params1(1);params1(2);-params1(3);params1(4)+pi];
31 [params2, cost2,alpha2] = gaussNewton(omega1,omega2,a1,a2, params2_0,
       max_iter, tol);
32
33 % check two possible minimum and select better one
34 if cost1<cost2
35     paramsf = params1;
36     cost = cost1;
37 else
38     paramsf = params2;
39     cost = cost2;
40 end
41
42 if cost<mini_loss
43     params = paramsf;
44     mini_loss = cost;
45 end
46
47 end
48
49 % calculate estimated joint axis
50 j1 = [cos(params(1))*cos(params(2));cos(params(1))*sin(params(2));sin(
       params(1))];
51 j2 = [cos(params(3))*cos(params(4));cos(params(3))*sin(params(4));sin(
       params(3))];
52
53
54 % calculate prediction error
55 angle_error(k,1)=rad2deg(acos(abs(dot(j1,j1_true))));
56 angle_error(k,2)=rad2deg(acos(abs(dot(j2,j2_true))));
57
58 end
59
60 % calculate mean prediction error
61 error_mean = mean(angle_error,1);
62
63 % Gauss-Newton algorithm
64 function [params, cost_final,alpha_check] = gaussNewton(omega1,omega2,a1,
       a2, params0, max_iter, tol)
65
66     params = params0;
67     cost = zeros(max_iter+1,1);
68     alpha_check = zeros(max_iter,1);
69
70
71     tau = 0.5;
72     c = 1e-5;
73
74     for iter = 1:max_iter
75
76         alpha = 1;
77         % calculate sum of squared residuals
78         y_pred = model(omega1,omega2,a1,a2, params);
```

```
79          cost(iter+1,1) = (norm(y_pred))^2/251;
80          % Jacobian Matrix
81          J = jacobian(omega1,omega2,a1,a2, params);
82
83          delta_p = inv(J' * J) * J' * y_pred;
84
85
86          while true
87              % calculate new prediction
88              params_new = params - alpha * delta_p;
89              y_pred_new = model(omega1,omega2,a1,a2, params_new);
90              cost_new = (norm(y_pred_new))^2/251;
91
92              if cost_new <= cost(iter+1,1) - c * alpha * (J' * y_pred)' *
                    delta_p
93                  break;
94              else
95                  alpha = tau * alpha;
96              end
97          end
98
99          alpha_check(iter,1) = alpha;
100         params = params - alpha * delta_p;
101
102         cost_final = cost(iter+1,1);
103         % check if converge
104         if abs(cost(iter+1,1)-cost(iter,1)) < tol
105             break;
106         end
107     end
108 end
109
110 % calculate residual based on measurements and estimated joint axis
111 function y = model(omega1,omega2,a1,a2,params)
112     w1 = 10;
113     w2 = 1;
114     y = zeros(size(omega1,1)*2,1);
115     phi1 = params(1);
116     theta1 = params(2);
117     phi2 = params(3);
118     theta2 = params(4);
119     j1 = [cos(phi1)*cos(theta1);cos(phi1)*sin(theta1);sin(phi1)];
120     j2 = [cos(phi2)*cos(theta2);cos(phi2)*sin(theta2);sin(phi2)];
121     for i = 1 : size(omega1,1)
122         y(i,1) = w1 * (norm(cross(omega1(i,:)',j1))-norm(cross(omega2(i
                ,:)',j2)));
123         y(size(omega1,1)+i,1) = w2 * (j1'*a1(i,:)'-j2'*a2(i,:)');
124     end
125 end
126
127 % calculate Jacobian matrix
128 function J = jacobian(omega1,omega2,a1,a2, params)
129     w1 = 10;
```

```
130     w2 = 1;
131     J = zeros(size(omega1,1)*2,4);
132     phi1 = params(1);
133     theta1 = params(2);
134     phi2 = params(3);
135     theta2 = params(4);
136     j1 = [cos(phi1)*cos(theta1);cos(phi1)*sin(theta1);sin(phi1)];
137     j2 = [cos(phi2)*cos(theta2);cos(phi2)*sin(theta2);sin(phi2)];
138     par_j = [-sin(phi1)*cos(theta1),-sin(phi1)*sin(theta1),cos(phi1)
            ,0,0,0;
139              -cos(phi1)*sin(theta1),cos(phi1)*cos(theta1),0,0,0,0;
140              0,0,0,-sin(phi2)*cos(theta2),-sin(phi2)*sin(theta2),cos(phi2
                 );
141              0,0,0,-cos(phi2)*sin(theta2),cos(phi2)*cos(theta2),0];
142     for i = 1 : size(omega1,1)
143         J(i,:) = w1 * (par_j * [cross(cross(omega1(i,:)',j1),omega1(i,:)
                ')/norm(cross(omega1(i,:)',j1));
144                                 cross(cross(omega2(i,:)',j2),omega2(i,:)
                                    ')/norm(cross(omega2(i,:)',j2))])';
145         J(size(omega1,1)+i,:) = w2 * (par_j * [a1(i,:)';a2(i,:)']);
146     end
147 end
```

A-2 Code for Deep Learning Method

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[ ]:
5
6
7  import tensorflow as tf
8  from tensorflow.keras.layers import Input, Dropout, Conv1D, LSTM, Dense,
       Concatenate,Flatten,Reshape,Layer
9  from tensorflow.keras.models import Model
10 from tensorflow.keras.optimizers import Adam
11 from tensorflow.keras.callbacks import ModelCheckpoint
12 from tensorflow.keras.callbacks import EarlyStopping
13
14 # Custom layer to calculate additional loss
15 class AdditionalLossLayer(Layer):
16     def __init__(self, **kwargs):
17         super(AdditionalLossLayer, self).__init__(**kwargs)
18         self.trainable = False
19
20     def call(self, inputs):
21         input1, input2, original_output = inputs
22         # constraint weight setting
23         w1 = tf.constant(0.1,dtype=tf.float32)
24
25         # input
26         omega1 = tf.cast(input1[:,:,0:3],dtype=tf.float32)
```

```
27          omega2 = tf.cast(input2[:,:,0:3],dtype=tf.float32)
28
29          # From output to rotation matrix
30          sin1 = tf.squeeze(original_output[:, 0])
31          cos1 = tf.squeeze(original_output[:, 1])
32          sin2 = tf.squeeze(original_output[:, 2])
33          cos2 = tf.squeeze(original_output[:, 3])
34
35          # calculate estimated joint axis
36          angle1 = tf.atan2(sin1,cos1)
37          angle2 = tf.atan2(sin2,cos2)
38          j1 = tf.stack([-tf.math.sin(angle1), tf.zeros_like(angle1), tf.
                math.cos(angle1)], axis=1)
39          j2 = tf.stack([-tf.math.sin(angle2), tf.zeros_like(angle2), tf.
                math.cos(angle2)], axis=1)
40
41          # calculate constraint loss
42          def angular_con(inputs):
43              omega1,omega2,j1,j2 = inputs
44              j1_32 = tf.tile(tf.expand_dims(j1, 0), multiples=[32, 1])
45              j2_32 = tf.tile(tf.expand_dims(j2, 0), multiples=[32, 1])
46              con_omega=tf.abs(tf.norm(tf.linalg.cross(omega1, j1_32),axis
                    =-1) - tf.norm(tf.linalg.cross(omega2, j2_32),axis=-1))
47              mean_con_omega = tf.reduce_mean(con_omega, axis=-1, keepdims=
                    True)
48              return mean_con_omega
49
50          ang_con = w1 * tf.map_fn(angular_con, (omega1, omega2, j1, j2),
                fn_output_signature=tf.float32)
51
52          return tf.concat([original_output, ang_con], axis=-1)
53
54  # Custom loss function
55  def custom_loss():
56      def loss(y_true, y_pred):
57          # original loss
58          mse_loss = tf.reduce_mean(tf.square(y_true[:, :-1] - y_pred[:,
                :-1]), axis=-1)
59          # additional loss from constraints
60          additional_loss = tf.reduce_mean(y_pred[:, -1], axis=-1)
61          # add all loss
62          total_loss = mse_loss + additional_loss
63          return total_loss
64      return loss
65
66  # Function to create IMU model
67  def create_imu_model():
68      input_imu = Input(shape=(32,6))
69      # three CNN layers
70      x = Conv1D(filters=16, kernel_size=4 , activation='relu')(input_imu)
71      x = Conv1D(filters=16, kernel_size=4 , activation='relu')(x)
72      x = Conv1D(filters=16, kernel_size=4 , activation='relu')(x)
73      # two LSTM layers
```

```
74      x = LSTM(32, return_sequences=True)(x)
75      x = LSTM(32)(x)
76      output = Dense(units=32, activation='linear')(x)
77      model = Model(inputs=input_imu, outputs=output)
78      return model
79
80  # Create two IMU models
81  imu_model1 = create_imu_model()
82  imu_model2 = create_imu_model()
83
84  # Inputs
85  input_imu1 = Input(shape=(32,6))
86  input_imu2 = Input(shape=(32,6))
87
88  # Branches
89  output_imu1 = imu_model1(input_imu1)
90  output_imu2 = imu_model2(input_imu2)
91
92  # Concatenate outputs
93  combined_output = Concatenate()([output_imu1, output_imu2])
94
95  # two FC layers
96  x = Dense(32, activation='relu')(combined_output)
97  x = Dense(32, activation='relu')(x)
98
99  # Output layer
100 output = Dense(units=4, activation='linear')(x)
101
102 additional_loss_output = AdditionalLossLayer()([input_imu1,input_imu2,
        output])
103
104 # Create main model
105 model = Model(inputs=[input_imu1, input_imu2], outputs=
        additional_loss_output)
106
107
108 # Compile the model
109 optimizer = Adam(learning_rate=0.001)
110 model.compile(optimizer=optimizer, loss=custom_loss())
111
112 # Print model summary
113 model.summary()
114
115
116 # set checkpoint
117 checkpoint = ModelCheckpoint("checkpoint.h5", save_best_only=True)
118 # early stopping
119 early_stopping = EarlyStopping(monitor='val_loss', patience=50, mode='min
        ', verbose=1)
120 # model training
121 history = model.fit(x = [input1,input2], y = output_data, epochs = 2000,
        batch_size = 512, validation_split=0.3, callbacks=[checkpoint,
        early_stopping])
```

```
122 # record training loss and validation loss
123 train_loss = history.history['loss']
124 val_loss = history.history['val_loss']
```

Appendix B

Simulation Results - Deep Learning Integrated with Constraint

w_{con}	**0**	**0.0001**	**0.0005**	**0.001**	**0.005**
MAE(°)	2.63/7.60	2.59/7.00	2.56/6.78	2.49/6.57	2.40/6.64
median error(°)	2.00/5.05	2.01/4.55	2.01/4.47	1.92/4.51	1.83/4.51
Loss$_{con}$	0.118	0.102	0.102	0.102	0.104
average error for 2 IMU(°)	5.12	4.80	4.67	4.53	4.52
w_{con}	**0.01**	**0.05**	**0.1**	**0.5**	**1**
MAE(°)	2.40/6.56	2.26/6.23	2.33/6.43	2.41/6.56	2.76/6.67
median error(°)	1.84/4.44	1.77/4.30	1.78/4.44	1.85/4.34	2.19/4.43
Loss$_{con}$	0.103	0.099	0.101	0.098	0.096
average error for 2 IMU(°)	4.48	4.25	4.38	4.49	4.72
w_{con}	**5**	**10**	**50**	**100**	**/**
MAE(°)	3.05/6.64	3.10/7.59	3.44/7.62	3.56/9.24	/
median error(°)	2.42/4.30	2.44/4.95	2.73/4.56	2.83/5.11	/
Loss$_{con}$	0.095	0.103	0.110	0.115	/
average error for 2 IMU(°)	4.85	5.35	5.53	6.40	/

Table B-1: Simulation results - integrated with constraint (IMU1/IMU2)

Bibliography

[1] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[2] Jacob S Arlotti, William O Carroll, Youness Afifi, Purva Talegaonkar, Luciano Albuquerque, John E Ball, Harish Chander, Adam Petway, et al. Benefits of imu-based wearables in sports medicine: Narrative review. *International Journal of Kinesiology and Sports Science*, 10(1):36–43, 2022.

[3] Samit Bhanja and Abhishek Das. Impact of data normalization on deep neural network for time series forecasting, 2019.

[4] Brice Bouvier, Sonia Duprey, Laurent Claudon, Raphaël Dumas, and Adriana Savescu. Upper limb kinematics using inertial and magnetic sensors: Comparison of sensor-to-segment calibrations. *Sensors*, 15(8):18813–18833, 2015.

[5] Francesco Caputo, Alessandro Greco, Egidio D 'Amato, Immacolata Notaro, and Stefania Spada. Imu-based motion capture wearable system for ergonomic assessment in industrial environment. In *Advances in Human Factors in Wearable Technologies and Game Design: Proceedings of the AHFE 2018 International Conferences on Human Factors and Wearable Technologies, and Human Factors in Game Design and Virtual Environments, Held on July 21–25, 2018, in Loews Sapphire Falls Resort at Universal Studios, Orlando, Florida, USA 9*, pages 215–225. Springer, 2019.

[6] Vivek Chandel, Arijit Sinharay, Nasimuddin Ahmed, and Avik Ghose. Exploiting imu sensors for iot enabled health monitoring. In *Proceedings of the First Workshop on IoT-enabled healthcare and wellness technologies and systems*, pages 21–22, 2016.

[7] Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Jeff Schneider, David Bradley, and Nemanja Djuric. Deep kinematic models for kinematically feasible vehicle trajectory predictions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10563–10569. IEEE, 2020.

[8] W.H.K. de Vries, H.E.J. Veeger, A.G. Cutti, C. Baten, and F.C.T. van der Helm. Functionally interpretable local coordinate systems for the upper extremity using inertial magnetic measurement systems. *Journal of Biomechanics*, 43(10):1983–1988, 2010.

[9] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[10] TensorFlow Developers. Tensorflow. *Zenodo*, 2022.

[11] Ján Drgoňa, Aaron R Tuor, Vikas Chandan, and Draguna L Vrabie. Physics-constrained deep learning of multi-zone building thermal dynamics. *Energy and Buildings*, 243:110992, 2021.

[12] I Arun Faisal, T Waluyo Purboyo, and A Siswo Raharjo Ansori. A review of accelerometer sensor and gyroscope sensor in imu sensors on motion capture. *J. Eng. Appl. Sci*, 15(3):826–829, 2019.

[13] Bingfei Fan, Haisheng Xia, Junkai Xu, Qingguo Li, and Peter B Shull. Imu-based knee flexion, abduction and internal rotation estimation during drop landing and cutting tasks. *Journal of Biomechanics*, 124:110549, 2021.

[14] J Favre, R Aissaoui, BM Jolles, JA de Guise, and K Aminian. Functional calibration procedure for 3d knee joint angle description using inertial sensors. *Journal of Biomechanics*, 42(14):2330–2335, 2009.

[15] J. Favre, B.M. Jolles, R. Aissaoui, and K. Aminian. Ambulatory measurement of 3d knee joint angle. *Journal of Biomechanics*, 41(5):1029–1035, 2008.

[16] J. Favre, B.M. Jolles, Olivier Siegrist, and Kamiar Aminian. Quaternion-based fusion of gyroscopes and accelerometers to improve 3d angle measurement. *Electronics Letters*, 42:612 – 614, 06 2006.

[17] Chenyu Gu, Weicong Lin, Xinyi He, Lei Zhang, and Mingming Zhang. Imu-based motion capture system for rehabilitation applications: A systematic review. *Biomimetic Intelligence and Robotics*, 3(2):100097, 2023.

[18] Chenyu Gu, Weicong Lin, Xinyi He, Lei Zhang, and Mingming Zhang. Imu-based motion capture system for rehabilitation applications: A systematic review. *Biomimetic Intelligence and Robotics*, 3(2):100097, 2023.

[19] Andrew S Jackson and Michael L Pollock. Generalized equations for predicting body density of men. *British journal of nutrition*, 40(3):497–504, 1978.

[20] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. 10 2017.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[22] Arne Küderle, Sebastian Becker, and Catherine Disselhorst-Klug. Increasing the robustness of the automatic imu calibration for lower extremity motion analysis. *Current Directions in Biomedical Engineering*, 4:439–442, 09 2018.

[23] Daniel Laidig, Philipp Müller, and Thomas Seel. Automatic anatomical calibration for imu-based elbow angle measurement in disturbed magnetic fields. *Current Directions in Biomedical Engineering*, 3, 09 2017.

[24] Tsubasa Maruyama, Haruki Toda, Wataru Ishii, and Mitsunori Tada. Inertial measurement unit to segment calibration based on physically constrained pose generation. *SICE Journal of Control, Measurement, and System Integration*, 13(3):122–130, 2020.

[25] Tim McGrath, Richard Fineman, and Leia Stirling. An auto-calibrating knee flexion-extension axis estimator using principal component analysis with inertial sensors. *Sensors*, 18:1882, 06 2018.

[26] Markus Miezal, Bertram Taetz, and Gabriele Bleser. On inertial body tracking in the presence of model calibration errors. *Sensors*, 16(7), 2016.

[27] Thomas Moeslund, Adrian Hilton, and Volker Krueger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126, 11 2006.

[28] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2):90–126, 2006. Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour.

[29] Philipp Müller, Marc-André Bégin, Thomas Schauer, and Thomas Seel. Alignment-free, self-calibrating elbow angles measurement using inertial sensors. *IEEE Journal of Biomedical and Health Informatics*, 21(2):312–319, 2017.

[30] Danny Nowka, Manon Kok, and Thomas Seel. On motions that allow for identification of hinge joint axes from kinematic constraints and 6d imu data. In *2019 18th European Control Conference (ECC)*, pages 4325–4331, 2019.

[31] JS Olson and S Redkar. A survey of wearable sensor networks in health and entertainment. *MOJ Appl. Bionics Biomech*, 2(5):280–287, 2018.

[32] Fredrik Olsson, Manon Kok, Thomas Seel, and Kjartan Halvorsen. Robust plug-and-play joint axis estimation using inertial sensors. *Sensors*, 20(12), 2020.

[33] Fredrik Olsson, Thomas Seel, Dustin Lehmann, and Kjartan Halvorsen. Joint axis estimation for fast and slow movements using weighted gyroscope and acceleration constraints. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8, 2019.

[34] Eduardo Palermo, Stefano Rossi, Francesca Marini, Fabrizio Patanè, and Paolo Cappa. Experimental evaluation of accuracy and repeatability of a novel body-to-sensor calibration procedure for inertial sensor-based gait analysis. *Measurement*, 52:145–155, 2014.

[35] Enrica Papi, Denise Osei-Kuffour, Yen-Ming A Chen, and Alison H McGregor. Use of wearable technology for performance assessment: A validation study. *Medical engineering & physics*, 37(7):698–704, 2015.

[36] Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. Multisensor-fusion for 3d full-body human motion capture. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 663–670, 2010.

[37] P. Raghavendra, M. Sachin, P. S. Srinivas, and Viswanath Talasila. Design and development of a real-time, low-cost imu based human motion capture system. In H.R . Vishwakarma and Shyam Akashe, editors, *Computing and Network Sustainability*, pages 155–165, Singapore, 2017. Springer Singapore.

[38] Samir A Rawashdeh, Derek A Rafeldt, and Timothy L Uhl. Wearable imu for shoulder injury prevention in overhead sports. *Sensors*, 16(11):1847, 2016.

[39] Daniel Roetenberg, Henk Luinge, and Per Slycke. Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technol. BV Tech. Rep.*, 3, 01 2009.

[40] Sarvenaz Salehi, Gabriele Bleser-Taetz, Attila Reiss, and Didier Stricker. Body-imu autocalibration for inertial hip and knee joint tracking. volume 2, 09 2015.

[41] Thomas Seel, Jörg Raisch, and Thomas Schauer. Imu-based joint angle measurement for gait analysis. *Sensors*, 14(4):6891–6909, 2014.

[42] Thomas Seel, Thomas Schauer, and Jörg Raisch. Joint axis and position estimation from inertial measurement data by exploiting kinematic constraints. In *2012 IEEE International Conference on Control Applications*, pages 45–49, 2012.

[43] Hongqing Song, Shuyi Du, Jiaosheng Yang, Yang Zhao, and Mingxu Yu. Evaluation of hydraulic fracturing effect on coalbed methane reservoir based on deep learning method considering physical constraints. *Journal of Petroleum Science and Engineering*, 212:110360, 2022.

[44] Bertram Taetz, Gabriele Bleser-Taetz, and Markus Miezal. Towards self-calibrating inertial body motion capture. 07 2016.

[45] Bertram Taetz, Michael Lorenz, Markus Miezal, Didier Stricker, and Gabriele Bleser-Taetz. Jointtracker: Real-time inertial kinematic chain tracking with joint position estimation. *Open Research Europe*, 4(33):33, 2024.

[46] Salvatore Tedesco, Oscar Manzano Torre, Marco Belcastro, Pasqualino Torchia, Davide Alfieri, Liudmila Khokhlova, Sokratis Dimitrios Komaris, and Brendan O'flynn. Design of a multisensors wearable platform for remote monitoring of knee rehabilitation. *IEEE Access*, 10:98309–98328, 2022.

[47] Laura Valencia, Felipe Schneider, Arnaldo Leal Junior, Pablo Caicedo Rodríguez, Wilson Sierra, Luis Rodriguez Cheu, Teodiano Bastos-Filho, and Anselmo Frizera. Sleeve for knee angle monitoring: An imu-pof sensor fusion system. *IEEE Journal of Biomedical and Health Informatics*, PP:1–1, 04 2020.

[48] Maria Inês Varela-Silva and Barry Bogin. Leg length and anthropometric applications: Effects on health and disease. In *Handbook of anthropometry: Physical measures of human form in health and disease*, pages 769–783. Springer, 2012.

[49] Adrian Waegli, Stéphane Guerrier, and Jan Skaloud. Redundant mems-imu integrated with gps for performance assessment in sports. In *2008 IEEE/ION Position, Location and Navigation Symposium*, pages 1260–1268. IEEE, 2008.

[50] G.F. Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *Computer Graphics and Applications, IEEE*, 22:24 – 38, 12 2002.

[51] Wikipedia contributors. Mean squared error — Wikipedia, the free encyclopedia, 2024. [Online; accessed 18-December-2024].

[52] Richard B Woodward, Sandra J Shefelbine, and Ravi Vaidyanathan. Pervasive monitoring of motion and muscle activation: Inertial and mechanomyography fusion. *IEEE/ASME Transactions on Mechatronics*, 22(5):2022–2033, 2017.

[53] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315, 08 2007.

[54] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[55] Tobias Zimmermann, Bertram Taetz, and Gabriele Bleser. Imu-to-segment assignment and orientation alignment for the lower body using deep learning. *Sensors*, 18(1), 2018.

[56] . Xsens mti-100 imu, 2024.

Glossary

List of Acronyms

DCSC	Delft Center for Systems and Control
IMU	Inertial Measurement Unit
I2S	IMU-to-Segment
DoF	Degree of Freedom
PCA	Principal Component Analysis
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
FC	Fully Connected
PGNN	Physics-Guided Neural Network
MSE	Mean Squared Error
API	Application Programming Interface
HPC	High Performance Computer
MAE	Mean Absolute Error
DoF	Degree of Freedom
ACL	Anterior cruciate ligament

List of Symbols

α^i_{I2S}	I2S angle, as well as the rotation angle around the z-axis of I_i that aligns it with S_i, $i = 1,2$
β_{s_1,s_2}	relative angle between s_1 and s_2
$\omega_{i,t}$	measured angular velocity at time t in frame I_i, $i = 1,2$
ω_{s_1,s_2}	relative angular velocity between two segments

ϕ_{ver}	angle by which s_1 rotates around the vertial axis
θ_{proj}	relative angle between s_1 and its projection onto the horizontal plane
$a_{i,t}$	measured acceleration at time t in frame I_i, $i = 1,2$
$a_{i,t}^{\text{cen}}$	joint center acceleration at time t in frame I_i, $i = 1,2$
G	global frame
g^G	gravitational acceleration in G
I_i	local frame of IMU i , $i = 1,2$
j_0	joint axis in segment frame (hinge joint)
j_i	joint axis in frame I_i, $i = 1,2$ (hinge joint)
$j_{1,2}, j_{2,2}$	two joint axes in segment frame (2-DoF joint)
$l_{i,t}$	length of s_i, $i = 1,2$
R^{I_iG}	rotation matrix from G to I_i, $i = 1,2$
r_i	relative position between frame I_i and joint center, $i = 1,2$
S_i	local frame of segment i , $i = 1,2$
s_i	segment i , $i = 1,2$
w_{con}	weight of constraint loss