FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

FH Aachen

Fachbereich Maschinenbau und Mechatronik

Studiengang Mechatronik

Masterarbeit

Application of Motion-Cueing Algorithm on a Cable-Driven parallel robot

vorgelegt von **Aadithya Ramamurthy**

Matrikel-Nr. **3304090**

Referent: Prof. Dr. rer. nat. Felix Hüning

Korreferent:

Externer Betreuer: Chrisitan Michael Lehnertz, M.Sc (Fraunhofer IPA)

Datum: 24.03.2025

In Zusammenarbeit mit

Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA

STATUTORY DECLARATION

I hereby declare that I have independently authored this master's thesis and have not used any sources or resources other than those explicitly stated. All material that has been quoted, either directly or in essence, from published or unpublished sources is clearly indicated and properly referenced. All images, drawings, and illustrations in this thesis were created by me or are accompanied by appropriate source references. This thesis has not been submitted, either wholly or in part, to any other examination authority in the same or a similar form.

Aachen, March 2025

_____

Aadithya Ramamurthy

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Dr. rer. nat. Felix Hüning at Fachhochschule Aachen and my supervisor Mr. Christian Michael Lehnertz, M.Sc. at Fraunhofer IPA for their continuous guidance and support throughout the course of my master's thesis. Their insights and encouragement were invaluable at every stage, from planning and programming to testing and documenting my work. Their help greatly contributed to the successful completion of this work.

I am also deeply thankful to my family for their unwavering support since the beginning of my master's program at FH Aachen. Their belief in my abilities and constant motivation enabled me to focus fully on my studies. I am especially grateful for their sacrifices and guidance, which allowed me to pursue my education without financial difficulties. My heartfelt thanks also go to my friends, who helped me adjust to a new environment and made life in Germany more comfortable and enjoyable.

Furthermore, I am grateful to Fraunhofer IPA for providing the necessary tools, workspace, financial assistance, and the opportunity to work within their robotics department particularly with industrial and cable-driven robots. I would also like to thank my teammates and colleagues at IPA, whose valuable input and support played a significant role in the success of my work.

ABSTRACT

Motion cueing algorithms (MCAs) are essential for motion simulation in applications such as pilot training, vehicle simulators, and virtual reality. While traditional hexapod-based platforms (Stewart-Gough platforms) are effective, they face limitations in workspace. Cable-Driven Parallel Robots (CDPRs), such as the IPAnema 3, offer higher scalability greater translational and rotational range, making them promising alternatives for realistic motion cueing.

This thesis presents the development and optimisation of a Classical Washout Algorithm (CWA) adapted for a CDPR. The process began with a literature review that identified a gap in applying MCAs to CDPRs, particularly in maintaining physical feasibility under cable force constraints. The proposed method was evaluated through Python-based simulations. The CWA was implemented using cascaded high-pass and low-pass filtering, scaling, and washout damping for both translational and rotational flight data obtained from an open-source simulator. Initial parameter tuning was performed manually through iterative adjustments based on cable force distribution feasibility.

A cost function was formulated using a modified Relative Absolute Error (RAE) to measure the normalised difference between the input forces and the forces after motion cueing. To reduce the influence of outliers and very low-force values, an offset was added in the denominator. Optimisation was performed using the Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Box constraints (L-BFGS-B) method to minimise the cost function, targeting only the washout damping parameters in the translational axes (X, Y, Z), while ensuring that all resulting platform motions remained physically feasible through cable force distribution checks.

The results indicated that force error factors after optimisation were either moderately improved or remained close to those obtained through manual tuning, with a slight improvement observed in the Y axis. However, the optimisation also introduced some trade-offs, as force factors increased slightly in the other axes, reflecting the challenges of managing inter-axis coordination in CDPR systems. Nevertheless, the cable force distribution plots after optimisation showed reduced fluctuations and greater stability, leading to more uniform and consistent cable forces along the entire trajectory. Additional tests on dynamic trajectories and higher payloads confirmed the robustness of the approach.

TABLE OF CONTENTS

STATUTORY DECLARATION ..... **I**
ACKNOWLEDGEMENT ..... **II**
ABSTRACT ..... **III**
LIST OF ABBREVIATIONS ..... **VII**
LIST OF FIGURES ..... **VIII**
LIST OF TABLES ..... **X**
1 INTRODUCTION **1**
1.1 Fraunhofer IPA 1
1.2 Motivation 1
1.3 Objectives 4
1.4 Thesis Structure 4
2 FUNDAMETNALS **5**
2.1 Motion Perception 5
2.2 Motion Cueing Algorithm 5
2.3 Classical Washout 6
2.3.1 Translational Channel 8
2.3.2 Tilt Coordination Channel 8
2.3.3 Rotational Channel 9
2.4 Other Algorithms 9
2.4.1 Adaptive Washout Algorithm 9
2.4.2 Optimal Control Algorithm 10
2.4.3 Model Predictive Control 11
2.5 Parallel Robots 11
2.6 Cable-Driven Parallel Robots 13
2.7 Optimisation 15
2.7.1 Cost Function Optimisation 15
2.7.2 Normalisation 16
3 LITERATURE AND STATE OF THE ART **18**
3.1 Motion Cueing Generation in Vehicle Simulators 18
3.2 Early Stages in Classical Washout Algorithm 19
3.3 Advancements of Classical Washout Algorithm 20
3.4 Integrating MCA with Platform Design 21

3.5 CDPR as Motion Platform 22
3.5.1 Cable Robot Simulator at the Max Planck Institute 22
3.5.2 IPAnema 3 at Fraunhofer IPA 23
3.6 Summary and Key Takeaways 24
4 METHODOLOGY AND IMPLEMENTATION 26
4.1 Collection and Processing of Dataset 28
4.1.1 Selection of Aircraft and Settings Configuration 29
4.1.2 Configuration of Data Logging (XML File) 29
4.1.3 Data Logging and Export 29
4.2 Setting up the Classical Washout Algorithm 30
4.2.1 Translation Channel Implementation 31
4.2.2 Rotational Channel Implementation 32
4.2.3 Tilt-Coordination Channel Implementation 33
4.3 Integration Force Distribution with CWA 34
4.3.1 Importing the WiPy Library 34
4.3.2 Defining IPAnema 3 Geometry 35
4.3.3 Calculation of Workspace 36
4.3.4 Defining and Calculation of Force Distribution 37
4.4 Cost Function-Based Parameter Optimisation for CWA 38
4.4.1 Defining the Cost Function 38
4.4.2 Optimisation of Washout Damping Parameters 40
5 EVALUATION AND RESULTS 43
5.1 Evaluation of Customised CWA 43
5.1.1 Evaluating Translational Accelerations 43
5.1.2 Evaluating Rotational Rates 47
5.2 Evaluation of Cost Function Optimisation 51
5.2.1 Force Factors as Cost Metric 51
5.2.2 Evaluation of Cost Function Optimisation Results 54
5.3 Testing on Different Trajectory 56
5.4 Testing on a High Payload 57
6 CONCLUSION 58
6.1 Summary 58
6.2 Future Work 59
A APPENDIX 61
A1 Data Logging using XML file. 61
A2 CWA for X acceleration and Pitch rate 61

A3 Validation of Force Distribution 62
A4 Cost Function Calculation 62
A5 Cost Function Optimisation 63
A6 Comparison of Final Parameters 64
BIBLIOGRAPHY 65

LIST OF ABBREVIATIONS

AWA	Adaptive Washout Algorithm
CAA	Coordinated Adaptive Algorithm
CDPR	Cable-Driven Parallel Robot
CRS	Cable Robot Simulator
CSV	Comma-Separated Values
DOF	Degrees of Freedom
HF	High Frequency
HP	High Pass
IPA	Institute for Manufacturing Engineering and Automation
LF	Low Frequency
L-BFGS-B	Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Box constraints
LP	Low Pass
MAE	Mean Absolute Error
MCA	Motion Cueing Algorithm
MPC	Model Predictive Algorithm
MPI	Max Planck Institute
MRAE	Mean Relative Absolute Error
MSE	Mean Square Error
NASA	National Aeronautics and Space Administration
OCA	Optimal Control Algorithm
PLC	Programmable Logic Controller
RAE	Relative Absolute Error
UTIAS	University of Toronto Institute for Aerospace Studies
VR	Virtual Reality

LIST OF FIGURES

Figure 1.1: Motion simulators using Hexapods - The Iowa Driving Simulator (left) [13] and its next generation, The National Advanced Driving Simulator (right) [14] 2

Figure 1.2 Motion simulators using serial kinematics – MPI Motion Simulator(left) [6] and its next generation MPI CyberMotion Simultor(right) [8] 2

Figure 1.3: The DLR Robot Motion Simulator during operation(left), with additional sidestick(right) [7] 2

Figure 1.4: The CDPR 'IPAnema 3' at Fraunhofer IPA [15] 3

Figure 1.5: CRS in operation at the Max Planck Institute for Biological Cybernetics [10] 3

Figure 2.1 : Location of the vestibular system in human ear. [18,19] 5

Figure 2.2: Translations and rotations of a six-component pose [28] 6

Figure 2.3: A general block diagram of a CWA [3] 8

Figure 2.4: ABB Parallel Robot (left) and the robot in operation (right) [37] 12

Figure 2.5: A parallel motion platform (Stewart-Gough design) [21] 12

Figure 2.6: Schematic representation of a CDPR with 'm' cables [12] 13

Figure 2.7: Cable-Driven SkyCam used in Sports events [12] 14

Figure 2.8: Giant Spherical Radio Telescope [12] 14

Figure 2.9: Cable-Driven Arm Exoskeleton for rehabilitation [12] 14

Figure 3.1 Motion cueing generation [21] 18

Figure 3.2: The CRS cross over configuration with 8 drive systems [10] 23

Figure 4.1: Flowchart of manually tuned CWA 26

Figure 4.2: Flow chart for integration of Cost function optimisation with the CWA 28

Figure 4.3: CWA Implementation block diagram 30

Figure 4.4: Translational Channel Implementation block diagram 31

Figure 4.5: Rotational Channel Implementation block diagram 32

Figure 4.6: Tilt-Coordination Channel Implementation block diagram 33

Figure 4.7: Example output for invalid poses 37

Figure 5.1: Comparison of input accelerations to filtered accelerations X axis translation 45

Figure 5.2: Comparison of input accelerations to filtered accelerations for Y axis translation 45

Figure 5.3: Positions obtained directly from Input accelerations to CWA 46

Figure 5.4: Positions obtained from the HP Filtered accelerations without washout damping 46

Figure 5.5: Final translational poses of the IPAnema 3 platform 47
Figure 5.6: Final velocities of the IPAnema 3 platform 47
Figure 5.7: Input Rotational Rates to the CWA 48
Figure 5.8: Rotational Rates after HP Filtering 48
Figure 5.9: Rotational angles obtained directly from input rotational rates 49
Figure 5.10: Rotational angles obtained from HP Filtered rotational rates without washout 49
Figure 5.11 Washout rotational angles without tilt 49
Figure 5.12: Tilt rates obtained from the tilt coordination channel 50
Figure 5.13 Tilt angles obtained from the tilt coordination channel 50
Figure 5.14: The final orientation of the IPAnema 3 platform 50
Figure 5.15: Comparison of input forces to the forces acting on the platform 52
Figure 5.16: Force Factors without Offset 53
Figure 5.17: Force Factors with Offset 53
Figure 5.18: Force distribution before optimisation 55
Figure 5.19: Force distribution after optimisation 56

Figure A.1: XML file code snippet for logging translational accelerations 61
Figure A.2: Python code for CWA 61
Figure A.3: Python code for force distribution validation 62
Figure A.4: Average force factor calculation for X axis in python 62
Figure A.5: Logic for Cost function optimisation in python 63

LIST OF TABLES

Table 1: CRS Parameters adapted from [10] 23
Table 2: IPAnema 3 Parameters adapted from [15, 53] 24
Table 3: Geometrical parameters of the IPAnema 3 adapted from [60] 35
Table 4: Parameters to evaluate translational accelerations 43
Table 5: Parameters to evaluate rotational rates 48
Table 6: Force factors and optimised washout damping factors 54
Table 7: Final parameter values for Trajectory 2 56
Table 8: Final parameter values for a high payload value 57

Table A-1: Comparison of final parameters 64

1 INTRODUCTION

This chapter contains a brief introduction about the Fraunhofer Institute and the department where the work for this Master thesis was conducted. Additionally, it presents the Motivation, Objectives, and Structure of the study.

1.1 Fraunhofer IPA

The Fraunhofer Institute for Manufacturing Engineering and Automation (Fraunhofer IPA), established in 1959, is one of the largest institutes within the Fraunhofer-Gesellschaft, employing approximately 1,200 professionals. Located in Stuttgart, Germany, the institute focuses on developing, testing, and implementing components, devices, and methods, up to entire machines and manufacturing plants, addressing organizational and technological challenges in production [1].

Within Fraunhofer IPA, the Robot and Assistive Systems department is dedicated to the development of robotic systems and automation solutions for both industrial and service sectors. With over 40 years of experience, the department specializes in creating innovative industrial robots, service robots, and intelligent machines, aiming to transfer robotic technologies into new products and solutions across various applications, including production, logistics, public environments, care facilities, and private homes [2].

1.2 Motivation

Motion simulators are systems designed to replicate real-world motion experiences by providing users with sensory feedback that mimics actual movement [3]. They are essential in the field of aviation, automotive engineering, virtual reality (VR), and robotics for training, research purposes and entertainment purposes. A motion simulator typically comprises a moving platform and a visualization system. The human vestibular system perceives inertial forces and angular velocities, playing a vital role in maintaining balance and spatial orientation [4]. These inertial forces, induced by acceleration and rotational movements, are referred to as motion cues. The control algorithm responsible for driving the simulator platform to replicate these accelerations and rotations is known as a Motion Cueing Algorithm (MCA).

Ideally, motion cues should be replicated in a one-to-one ratio to achieve a fully accurate perception of simulated motion. However, obtaining the pose of a simulator’s platform by double integrating translational and rotation accelerations often results in coordinates that exceed its physical limits [3]. To address this limitation and ensure that the simulated trajectory remains within the platform's operational boundaries, the raw inertial signals comprising acceleration and rotational data must be processed. Various types of MCAs are available for this purpose.

Traditionally, motion simulators have utilized hexapods, which are based on parallel kinematics, as demonstrated by [5], shown in the Figure 1.1(left) and its next generation in Figure 1.1(right). More recent advancements have incorporated industrial robots with serial kinematics, as shown by [6], seen in the Figure 1.2(left), [7] seen in Figure 1.3, and [8], seen in the Figure 1.2(right)

Figure 1.1: Motion simulators using Hexapods - The Iowa Driving Simulator (left) [13] and its next generation, The National Advanced Driving Simulator (right) [14]

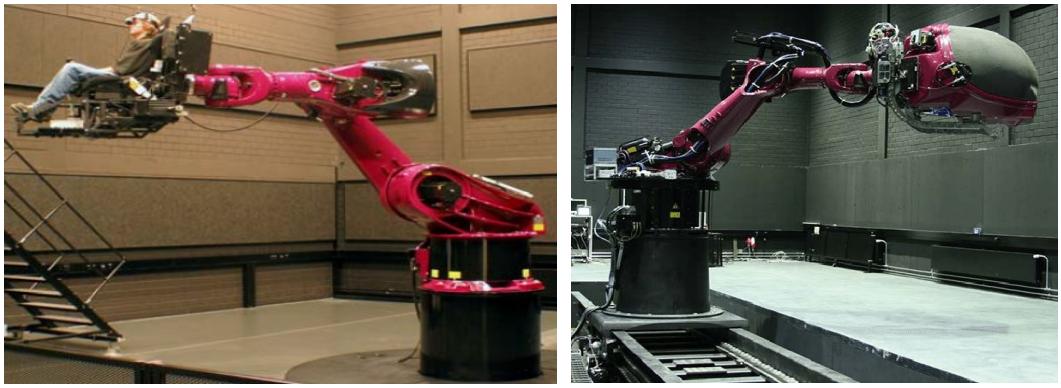

Figure 1.2 Motion simulators using serial kinematics – MPI Motion Simulator(left) [6] and its next generation MPI CyberMotion Simultor(right) [8]

Figure 1.3: The DLR Robot Motion Simulator during operation(left), with additional sidestick(right) [7]

However, there are some limitations with serial manipulators. Serial configurations often exhibit lower structural rigidity, low stiffness and load-bearing capacity due to their open kinematic chains and poor dynamic performance as shown by [9]. Unlike hexapods, which rely on rigid mechanical links, Cable-Driven Parallel Robots (CDPRs) use winch-driven cables and can provide a significantly larger range of motion than hexapods, particularly in the translational workspace. This is crucial for replicating realistic motion cues without exceeding platform limits [10]. This advantage is especially critical when simulating high-speed or long-duration manoeuvres involving sustained acceleration, which would push a hexapod faster to its physical capabilities [11,12].

This thesis explores the implementation of motion simulation in CDPRs. In collaboration with the Fraunhofer IPA, the operational CDPR "IPAnema 3", shown in the Figure 1.4 will be utilized for this study. Fraunhofer IPA aims to develop its own motion simulator using a CDPR in the near future, as it is currently in possession of the hardware of the Cable Robot Simulator (CRS) as shown in the Figure 1.5, originally developed at the Max Planck Institute(MPI) for Biological Cybernetics. This thesis lays the path for Fraunhofer IPA to use the findings of the work on the CRS.

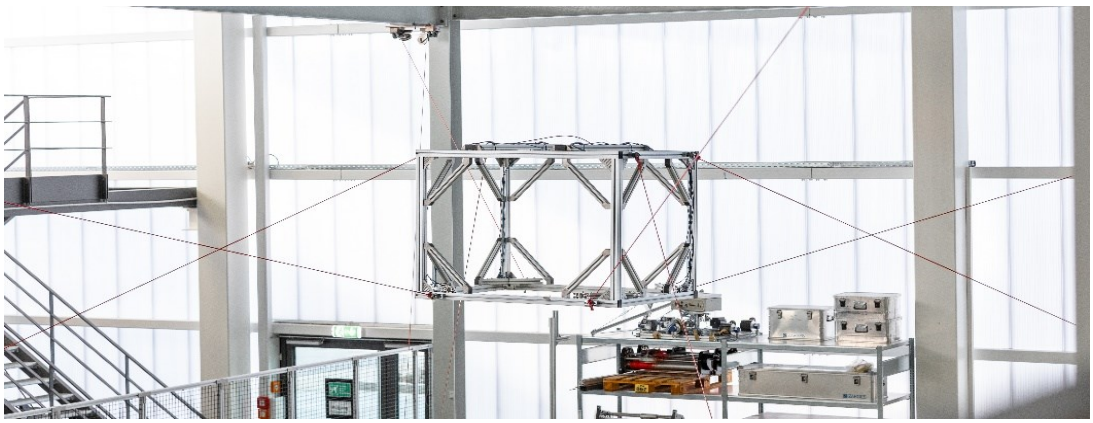

Figure 1.4: The CDPR 'IPAnema 3' at Fraunhofer IPA [15]

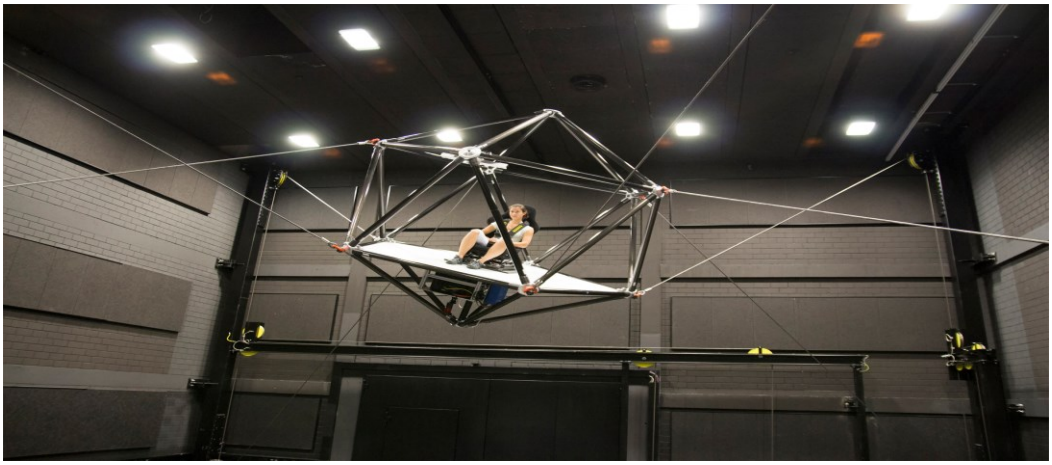

Figure 1.5: CRS in operation at the Max Planck Institute for Biological Cybernetics [10]

1.3 Objectives

The primary objective of this study is to implement a suitable MCA for the software simulation of the IPAnema 3 and further optimize it to maximize the utilization of its workspace. The algorithm's input will consist of a dataset containing translational acceleration and rotational rate representing a motion trajectory.

The initial phase involves collecting this dataset using an open-source software to capture realistic motion profiles. Subsequently, the MCA is implemented in Python within the software simulation of the IPAnema 3, utilizing existing Python libraries for CDPRs developed at Fraunhofer IPA. The geometry of the IPAnema 3 is defined within the simulation to ensure that the robot operates within the force limits of the cables while generating the required poses after applying the MCA.

Following the implementation, the MCA parameters are optimized to achieve optimal workspace utilization, marking the successful application of the algorithm on the IPAnema 3.

1.4 Thesis Structure

This work is organized as follows:

- Chapter 2 covers the fundamental concepts and theories essential for understanding the study.
- Chapter 3 presents the current state of the art, offering a comprehensive literature review on existing motion simulators and related research. It also analyses various MCAs, simulator platforms, and implementation techniques.
- Chapter 4 details the methodology employed in this study, including the step-by-step implementation of the MCA on IPAnema 3 and its optimization process.
- Chapter 5 discusses the evaluation of the MCA, and the results obtained.
- Chapter 6 concludes the work by summarizing the key findings and providing an outlook on potential future research.

2 FUNDAMETNALS

This chapter provides the theoretical foundation required to understand human motion perception, MCA, simulator platforms, and CDPRs. It also presents an overview of MCAs and optimization techniques.

2.1 Motion Perception

Visual cues play a crucial role in the perception of movement relative to the surrounding environment and estimation of an observer's position within the 3D environment. This is how humans interpret their own movement in space. However, human visual motion perception is tuned to velocity rather than acceleration [16]. As a result, the driving simulators, which rely heavily on their visual systems to provide accurate speed perception, perform best in scenarios where motion remains relatively constant [3]. The vestibular system, located within the inner ear as shown in the Figure 2.1, is responsible for detecting head movements, orientation, and acceleration, playing a key role in motion perception and balance. Any sudden changes or disturbances in motion are detected more quickly by the vestibular system than by the visual system [17]. Hence, specific forces generated from a range of acceleration cues can be replicated in a simulation using a motion system, a device specifically designed to mimic these forces and enhance the realism of the simulated environment [3].

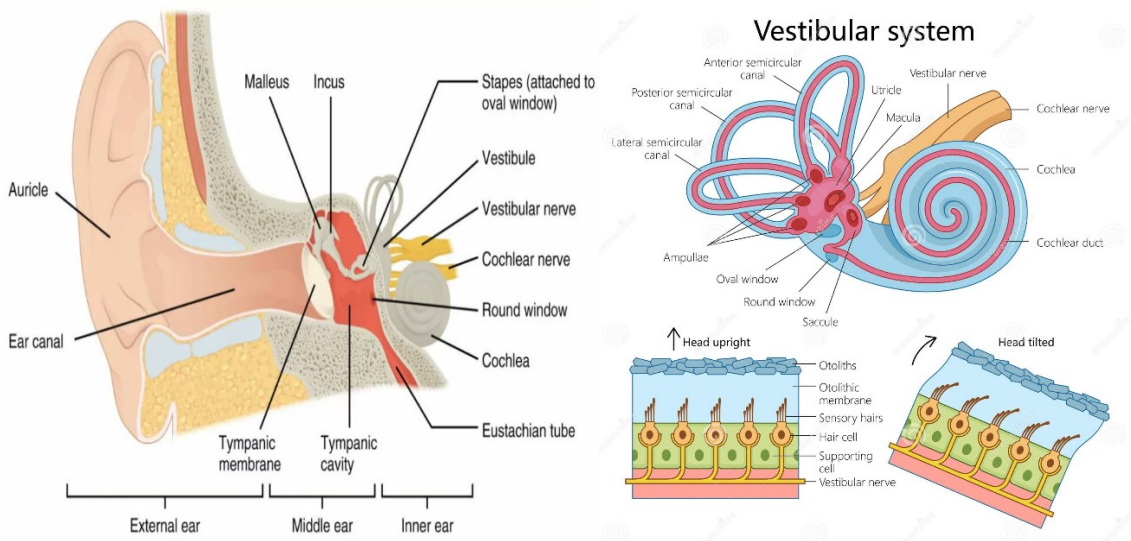

Figure 2.1 : Location of the vestibular system in human ear. [18,19]

2.2 Motion Cueing Algorithm

Motion cueing refers to the presentation of visual, auditory, vestibular, and haptic information(cues) to replicate real-world movements within virtual environments [20]. However, MCAs primarily consider vestibular information, as the remaining sensory cues are generated by

other components of the simulator [21]. Thus, the main aim of MCAs is to generate motion cues to match the forces experienced by a person on a simulator platform to that of the forces experienced in real-world vehicles. This comparison of the generated motion cues to the simulated vehicle dynamics is called as physical validity [22]. Due to the limited workspace of the motion platform of the simulator, the simulated physical state cannot be directly mapped onto it, except for certain rotational degrees of freedom (DOF). Although it varies individually from simulator to simulator. Hence, achieving a fully accurate replication of real-world motion is often not feasible. Additionally, simulated vehicles can produce high accelerations that exceed the capabilities of the motion platform's actuators. As a result, perceptual validity is prioritized, ensuring that the driver's multi-sensory experience of self-motion closely resembles real-world conditions rather than striving for exact physical replication. The primary objective is to deceive the operator's senses, creating a realistic piloting/driving experience. To achieve this, MCAs incorporate various techniques to manipulate sensory perception effectively [21].

An MCA takes as input certain physical quantities of the simulated vehicle, computed by the physics model, and outputs the desired pose of the motion platform. This pose is represented as a six-component vector, consisting of translational (position) and rotational (orientation) DOF. A six-component pose contains surge, sway, heave (translations along x-axis, y-axis, z-axis respectively) and roll, pitch, yaw (rotations along x-axis, y-axis, z-axis respectively) [3] as represented in the Figure 2.2. Usually, the inputs to MCAs are the linear acceleration and the angular velocity of the simulated vehicle [23]. These are the stimuli that the human vestibular system is capable of sensing. Other inputs such as angular acceleration, position, orientation are also used [21].

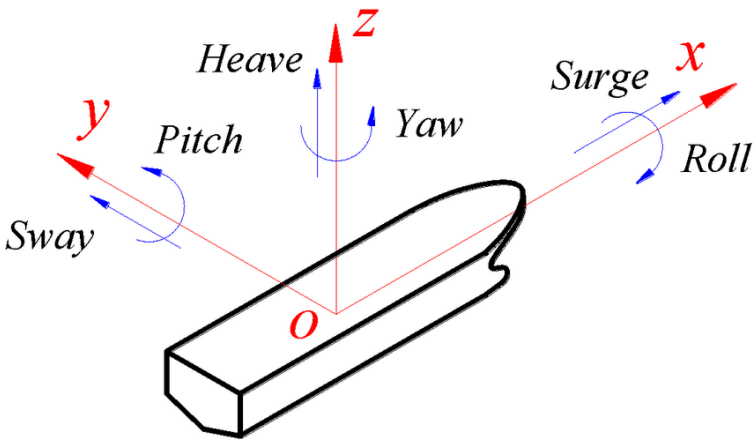

Figure 2.2: Translations and rotations of a six-component pose [28]

2.3 Classical Washout

Of all the MCAs that are in use today, particularly within the domain of flight simulation, the CWA is most widely used [24]. CWA is a foundational technique used in motion simulators. The pioneers of the first MCA were the engineers at the National Aeronautics and Space

Administration (NASA), Schmidt and Conrad. [25, 26]. They were the first to mathematically define motion cueing and develop a structured MCA for simulators. They recognized that human motion perception (via the vestibular system) plays a key role in simulator realism. This algorithm is the CWA. Later, other researchers modified and improved the CWA, keeping its basic ideas. The work of Schmidt and Conrad was revised and studied in depth by the researchers Reid and Nahon at University of Toronto Institute for Aerospace Studies (UTIAS) [23, 27]. Their work provided mathematical justification for the parameter tuning of washout filters to improve motion realism. They also proposed adaptive versions of the CWA, where filter parameters change dynamically based on the motion requirements. There are some variations from different implementations, but in general the principles are similar [21].

Four basic ideas are used in a CWA as described by [21]:

1) **Scaling of inputs** – The algorithm reduces the magnitude of input signals relative to the simulator's physical capabilities to minimize excessive platform movement and prevent exceeding its mechanical limits.
2) **Filtering of low-frequency (LF) accelerations** – LF acceleration components are removed to avoid large displacements which are attained from sustained or very high accelerations that would drive the motion platform beyond its operational constraints.
3) **Tilt coordination** – The previously removed LF accelerations are partially recreated through platform tilting, utilizing the gravity vector to simulate sustained accelerations.
4) **Automatic return to neutral position** – When no further motion input is detected, the algorithm gradually returns the platform to its neutral pose, ensuring stability and readiness for the next movement—hence the term "washout."

The CWA processes the motion signals from a simulated vehicle and transforms them into commands for the motion platform. CWA is split into 3 main channels to achieve this transformation [23, 27]: a translational, a rotational, and a tilt coordination channel as seen in the block diagram of the Figure 2.3.

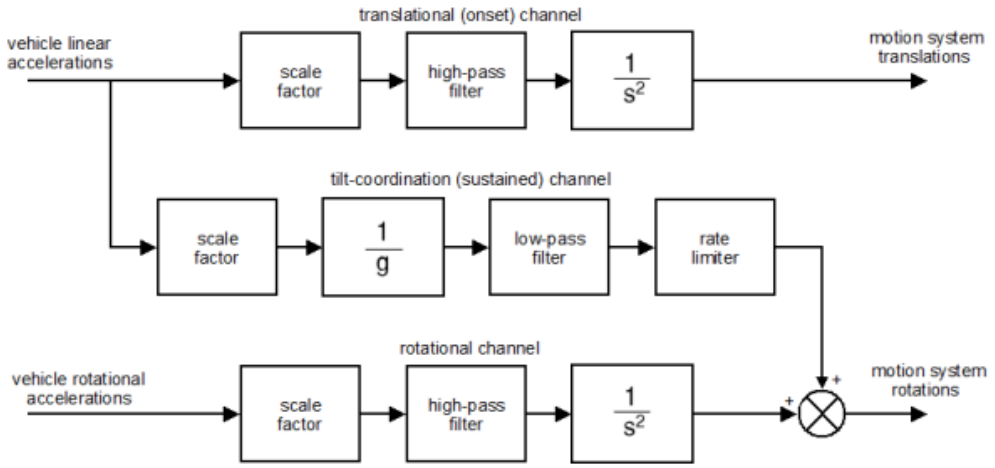

Figure 2.3: A general block diagram of a CWA [3]

2.3.1 Translational Channel

The translational channel is responsible for generating translational motion by processing linear accelerations through high-pass (HP) filters to replicate high-frequency (HF) translational movements [21]. Initially, the specific force (originating from the accelerations) may be scaled down, helping to keep the simulator's motion within the hardware constraints of the motion platform. Following this, the signal undergoes processing through a second-order HP filter, which eliminates LF components that could otherwise result in large, unrealistic displacements of the platform as the pose would not be achievable by the simulator. This ensures that only rapid changes in acceleration are retained. To achieve the desired washout effect, a damping factor is incorporated into the filter [29]. The filtered acceleration signals are then processed through a double integrator, which calculates the final sway, surge, and heave values that dictate the translational movements of the motion platform [21].

2.3.2 Tilt Coordination Channel

This channel is responsible for processing the LF components of linear acceleration, which are not managed by the translational channel. To simulate sustained acceleration, tilt coordination aligns the gravity vector with the direction of the specific force by tilting the motion platform through slow rotations [21, 29]. Initially, the scaled-down acceleration is divided by gravitational acceleration to compute the roll and pitch angles required for platform rotation to generate a gravity vector component equivalent to the desired specific force [3]. The computed signal is then processed through a LP filter to remove HF components. Additionally, a rate limiter is applied to constrain the tilt rotation, ensuring it remains within the perceptual threshold of 3°/s angular velocity and 0.3°/s² angular acceleration as higher values would lead to rotational false cues [30]. Keeping the values below the threshold effectively tricks the vestibular system into perceiving continuous or sustained acceleration.

2.3.3 Rotational Channel

This channel processes the angular velocities or accelerations encountered during the rotational movements of the simulated vehicle. Functionally similar to the Translational Channel, it is designed to reproduce HF angular displacements while preventing LF drifts [21]. Since rotational acceleration is typically short-lived, with body roll stabilizing quickly as the vehicle enters a curve, HP filtering alone is sufficient, as the primary motion cues exist within the HF range [3]. Initially, the angular components are scaled down before being processed through a second-order HP filter, where a damping factor is applied to achieve the desired washout effect. The filtered signal is then integrated to compute the Euler angles (roll, pitch, and yaw). While the yaw angle is taken as the output directly, the roll and pitch angles are further combined with the output of the Tilt Coordination Channel to generate the final angular output for the motion platform [21].

2.4 Other Algorithms

While the CWA is widely used in motion cueing due to its simplicity and effectiveness, certain limitations affect its overall performance. Firstly, it does not consider the specific shape of the motion platform's reachable workspace, making it highly adaptable but less optimized for a given hardware configuration. Secondly, the algorithm requires offline parameter tuning before execution, ensuring that the generated motion remains within platform limits, regardless of the input signals [21]. Another drawback is the occurrence of false cues which are unintended motion sensations experienced by users in the opposite direction during washout, that degrade the quality of simulation [31, 32]. To address these challenges, several enhancements to the CWA have been proposed.

2.4.1 Adaptive Washout Algorithm

CWA utilizes linear filters and scaling elements with time-invariant coefficients, combined with non-linear limiting and transformation processes such as tilt coordination. In contrast, adaptive washout algorithms (AWA) incorporate time-varying filter or scaling coefficients to dynamically adjust motion cueing parameters [21]. The concept of AWA was first introduced by [33]. This approach involves modifying the fixed parameters of the CWA, allowing filter settings to be continuously adjusted in order to minimize a cost function [3]. The UTIAS adaptive scheme, described by [34], utilizes adaptive filter gains that are varied in real time to minimize a cost function, which penalizes motion error (the discrepancy between platform motion and simulated vehicle motion), motion magnitude, and excessive variations in adaptive parameters from their initial values [21].

The adaptive filter for the translational channel utilizes a single adaptive parameter and is designed to align the translational acceleration input with the filter output while penalizing excessive translational displacement and velocity, ensuring that the simulator remains within its motion envelope [29]. Meanwhile, the adaptive filter for the rotational channel incorporates two adaptive

parameters, one for tilt coordination and another for the rotational velocity set point. The tilt coordination parameter ensures that motion cues remain below the pilot's perception threshold, while the filter works to match the angular velocity and tilt coordination set points while penalizing large angular displacements and velocities [29].

AWA has demonstrated superior performance compared to other washout implementations. However, its adoption is less widespread than CWA due to its higher complexity [29].

2.4.2 Optimal Control Algorithm

CWA and AWA do not incorporate human perception, as the vestibular system is not considered in their design [21]. In contrast, Optimal Control algorithms (OCA) are developed to minimize a cost function that accounts not only for motion state and motion cueing errors but also integrates a motion perception model to enhance realism [3, 35].

[36] proposed an optimal washout approach, treating the motion cueing generation system as a control problem, where the system's output is compared to the expected output, and the discrepancy is minimized using control techniques. Instead of optimising an existing non-optimal structure, this method aims to find an optimal structure [21], as demonstrated by [36].

[35] addressed this issue by introducing a new method that incorporates a vestibular model, which was not utilized in [21, 36]. The core concept involves defining a function for the washout filter that minimizes motion error while ensuring that the simulator operates within its physical constraints, essentially achieving cost function minimization. [35] further integrates a mathematical model of the human vestibular system to reduce the sensation error between a simulator user and a real vehicle operator. Despite its conceptual simplicity, the mathematical implementation remains complex [21].

Similar to the CWA, OCA produces fixed-parameter filters that do not fully utilize the motion platform's capabilities and must be adjusted based on the worst-case scenario. The washout filter structure is determined offline, using mathematical models that represent the system components, including the simulator and the vehicle operator [21].

Despite being labelled as "optimal", this approach is not the most widely used MCA, likely due to its mathematical complexity and the lack of evidence proving it to be the most effective method. One major drawback is that it treats all aspects of motion simulation as mathematical models, even though human motion perception is not yet fully understood, and an accurate model for motion perception has yet to be established [21].

2.4.3 Model Predictive Control

Previous MCAs extract a specific frequency range from the vehicle's acceleration signal. However, this process introduces phase shifts and can lead to false cues due to the washout effect [21]. To address these limitations and prevent exceeding the physical constraints of the motion platform, a Model Predictive Control (MPC) strategy has been proposed [3]. This approach was first implemented by [37] in the Renault ULTIMATE Driving Simulator.

The proposed algorithm closely replicates the reference signal as long as the motion platform can follow it, while ensuring smooth deceleration near its physical limits, thereby preventing false motion cues. As a result, the MCA parameters are explicitly defined by the motion platform constraints and motion detection thresholds, eliminating the need for worst-case scenario, tuning and allowing adjustments to be made by non-experts [21]. This is made possible by its ability to handle a multivariable, constrained optimization problem. The input accelerations are predicted in advance and closely matched to the corresponding platform motion [3].

The algorithm is designed for implementation in a digital controller and operates with a fixed time interval. At each time step, the MPC computes a feasible control sequence and a state sequence over a prediction horizon of a number of steps. The control input is optimized to minimize the squared perception error, which is the difference between the acceleration perceived in the simulator and the actual acceleration in the vehicle, while ensuring that the motion platform comes to a complete stop before reaching its physical limits [21]. The first value of the control sequence is applied at each time step. If the algorithm predicts that the platform is approaching its limits, it initiates a washout manoeuvre, gradually returning the platform toward its neutral position while staying below the motion perception threshold for the remaining prediction horizon. However, this process inevitably introduces motion conflicts, as the system must gradually decelerate to prevent the platform from exceeding its operational boundaries [21].

Although MCA based on the MPC strategy show promise, they are still in the early stages of development and require further refinement. The advantages of this approach over the well-established CWA remain uncertain [3]. Additional experiments and evaluations are necessary to thoroughly assess its performance in comparison to existing MCA methods [21].

2.5 Parallel Robots

Parallel robots consist of two or more closed-loop kinematic chains, where the end-effector (Moving Platform) is connected to the fixed base via at least two independent kinematic chains, known as limbs or legs [9]. Each limb is controlled by a single actuator, with all actuators positioned at or near the fixed base. Since external loads are distributed among multiple actuators, parallel manipulators generally offer high load-carrying capacity [9].

These mechanisms are primarily utilized for pick-and-place tasks or for handling heavy payloads within a confined workspace. Examples include the ABB Parallel Robot, commonly employed in handling, assembly, packaging, and pick-and-place operations, as illustrated in Figure 2.4[37], and the Stewart-Gough Platform commonly known as Hexapod, widely used as a motion platform, as shown in Figure 2.5[21].

Figure 2.4: ABB Parallel Robot (left) and the robot in operation (right) [37]

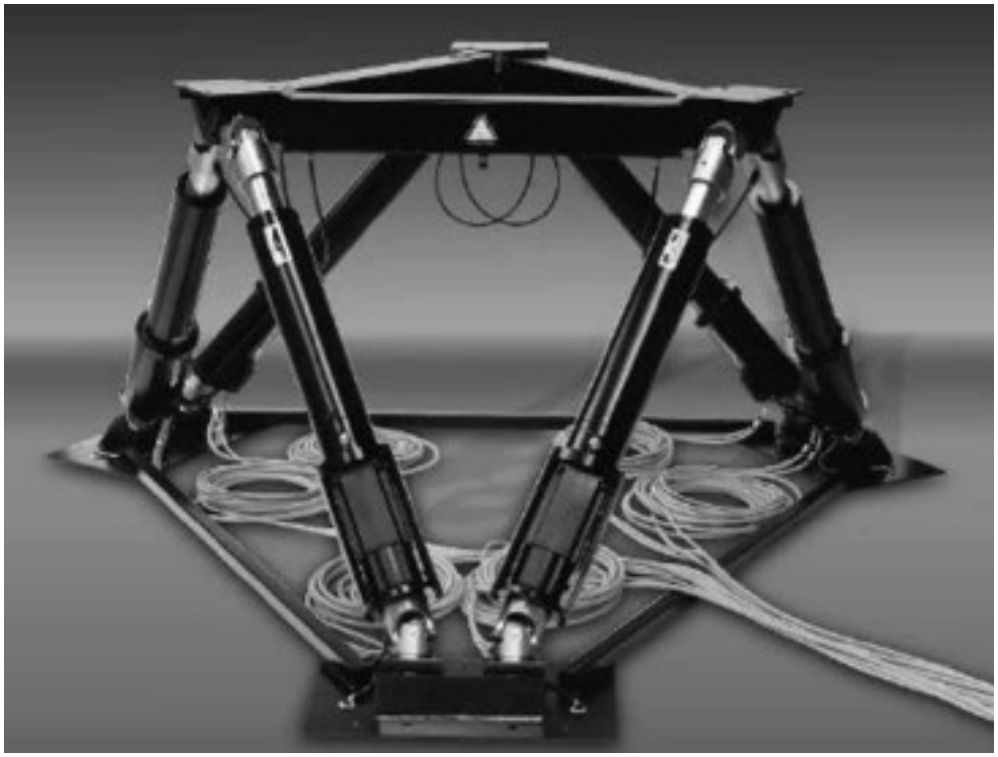

Figure 2.5: A parallel motion platform (Stewart-Gough design) [21]

Although parallel robots offer key advantages such as high stiffness, and significant payload capacity, they also present challenges, including limited workspace, design and control difficulties due to its complex kinematics and the mathematics behind it. [9].

2.6 Cable-Driven Parallel Robots

CDPRs are a specialized class of parallel robots where the end-effector is suspended and manipulated using multiple flexible cables, replacing the rigid links found in traditional rigid-link parallel robots [12]. Compared to conventional parallel robots, CDPRs offer a higher payload-to-weight ratio and superior dynamic performance [12, 38]. Additionally, the extended reach and the flexibility in the placement of the robot's pulleys allow CDPRs to be set up in different configuration. It is utilized in complex applications that demand to cover large workspaces. The variable configuration makes it particularly suitable for challenging tasks requiring extensive mobility [11, 12]. Cable robots are considered flexible because their configuration can be easily modified. The workspace can be adjusted simply by repositioning the pulleys, allowing for versatile setups. Since only the mobile platform is moved—typically lightweight—the system can utilize large, floor-mounted motors to achieve dynamic motion. The design of the platform and the motor power play a crucial role in determining the maximum achievable speeds, as well as the range of movements and rotations.

A typical CDPR system consists of three main components: a fixed platform, a mobile platform, and multiple cables that connect them. The cable lengths are adjusted using winches driven by motors mounted on the fixed platform, enabling controlled movement of the mobile platform, as illustrated in Figure 2.6[12].

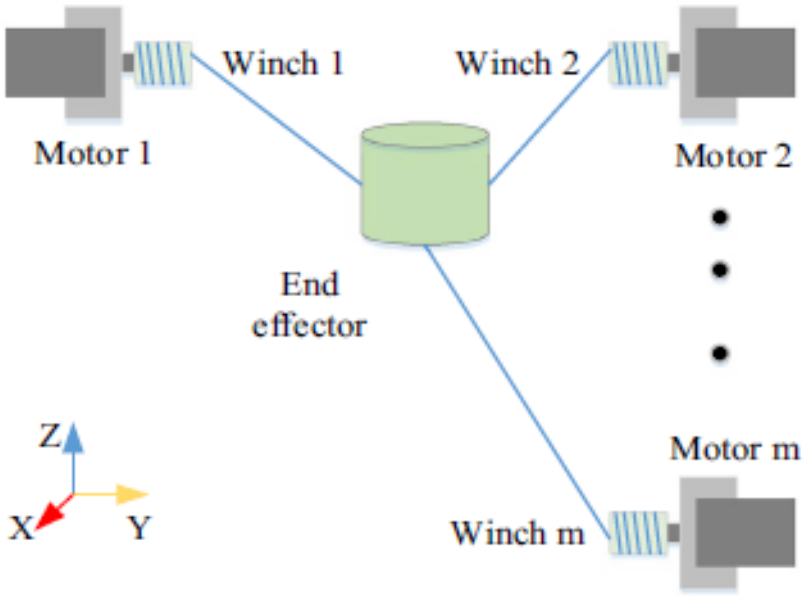

Figure 2.6: Schematic representation of a CDPR with 'm' cables [12]

CDPRs are utilized in various applications, including surface finishing very large objects, additive manufacturing of large components, and camera positioning for filming sports events (see Figure 2.7) [12, 38]. Other notable applications include giant radio telescopes (see Figure 2.8), rescue systems, rehabilitation robotics (see Figure 2.9) [12, 38].

Figure 2.7: Cable-Driven SkyCam used in Sports events [12]

Figure 2.8: Giant Spherical Radio Telescope [12]

Figure 2.9: Cable-Driven Arm Exoskeleton for rehabilitation [12]

Unlike rigid links, cables can only sustain tension, making tension maintenance and sag prevention a significant challenge. This limitation affects the development and broader application of CDPRs. However, advancements in optimal design and control theory over the past decade have led to

notable improvements in the kinematic and dynamic performance of CDPRs. Despite these advancements, CDPRs remain less commonly used in industrial manufacturing compared to serial robots and rigid-link parallel robots [12].

2.7 Optimisation

The term optimisation is often linked to the idea of improvement, but in mathematical terms, it specifically refers to the process of identifying the best possible solution by modifying controllable variables, typically within predefined constraints. Due to its importance in decision-making, optimization is applied across a wide range of domains, driven by the human pursuit of efficiency. Any scenario that requires choosing the best course of action can be formulated as an optimization problem. While some simple cases can be solved using analytical methods, most real-world problems are too complex for such direct approaches [39].

The progression of numerical computing, coupled with advancements in optimisation algorithms, has enabled the solution of increasingly complex problems. Numerical optimization was first developed in the field of operations research, which addresses challenges such as pricing strategies, network distribution design, scheduling, and route optimization. Beyond operations research, optimization also plays a fundamental role in areas such as optimal control and machine learning [39].

2.7.1 Cost Function Optimisation

A cost function, also referred to as a loss function or objective function, is a mathematical function that quantifies the error between the predicted and real values. The goal of the cost function optimisation is to minimize the cost function which refers to bringing the error rate as close as possible to 0 [40]. Most used cost functions are:

- **Mean Absolute Error (MAE):** MAE calculates the average absolute difference between the predicted and actual values [42] as seen in Equation (1).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | y_i - \overline{y}_i | \quad (1)$$

where:

- n = number of data points
- y_i = actual value
- $\overline{y_i}$ = predicted value

- **Mean Square Error (MSE):** MSE calculates the average squared difference between the predicted and actual values [42] as seen in Equation (2).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2 \quad (2)$$

where:

- n = number of data points
- y_i = actual value
- $\overline{y_i}$= predicted value

In MSE, since each error is squared, it helps to penalize even small deviations in prediction when compared to MAE. But if the dataset has outliers that contribute to larger prediction errors, then squaring this error further will magnify the error many times more and also lead to higher MSE error. In MAE, since the average sum of absolute differences between the two values are measured, the possibility of negative error is eliminated [41].

2.7.2 Normalisation

Data normalisation is a process of converting all data features into a standardised format, ensuring that all variables in the data have a similar range. In cost functions, it is a preprocessing step that prevents the influence of the inconsistencies that are observed when calculating the error. This reduces the impact of the outliers in the cost function leading to its better optimisation [43].

Relative Absolute Error (RAE) is a normalized error metric that measures the deviation of predicted values from actual values relative to the magnitude of the actual values [39]. Mean Relative Absolute Error (MRAE) is an extension of the Relative Absolute Error (RAE), used to measure the average deviation between predicted and actual values across an entire dataset. MRAE is nothing but dividing MAE with the magnitude of actual value. It is seen in Equation (3).

$$MRAE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \overline{y}_i|}{|y|} \quad (3)$$

where:

- n = number of data points
- y_i = actual value

- $\overline{y_i}$= predicted value

RAE measures how much the predicted value deviates from the actual value, relative to the actual value itself. A lower RAE indicates better model performance. RAE = 0 means the prediction is perfect.

3 LITERATURE AND STATE OF THE ART

This chapter presents a comprehensive review of the existing literature, previous studies, and state-of-the-art research in the field of MCAs, their implementation, and optimization. It begins by examining research on MCAs, with a primary focus on the CWA and its development. The discussion then shifts to studies related to motion platforms, with an emphasis on CDPRs. Additionally, the chapter explores optimization techniques applied to CWA for motion simulators, highlighting strategies to enhance performance. Finally, it identifies challenges and research gaps, underscoring areas where further advancements are required. This chapter serves as the scientific foundation for the methodology and implementation of MCAs in this thesis.

3.1 Motion Cueing Generation in Vehicle Simulators

Motion cueing generation is a critical component in vehicle simulators enabling the realistic replication of physical motion cues, which has been briefly explained in [21]. This process involves three key elements: a physics model, a motion platform, and an MCA.

The physics model is responsible for computing the vehicle dynamics, based on the operator's (pilot/driver's) control inputs. It solves the differential equations of motion to determine the simulated physical state of the vehicle [44]. The MCA translates the simulated vehicle motion into commands for the motion platform. The complete motion cueing generation process is illustrated in Figure 3.1, where the physics model, MCA, and motion platform interact.

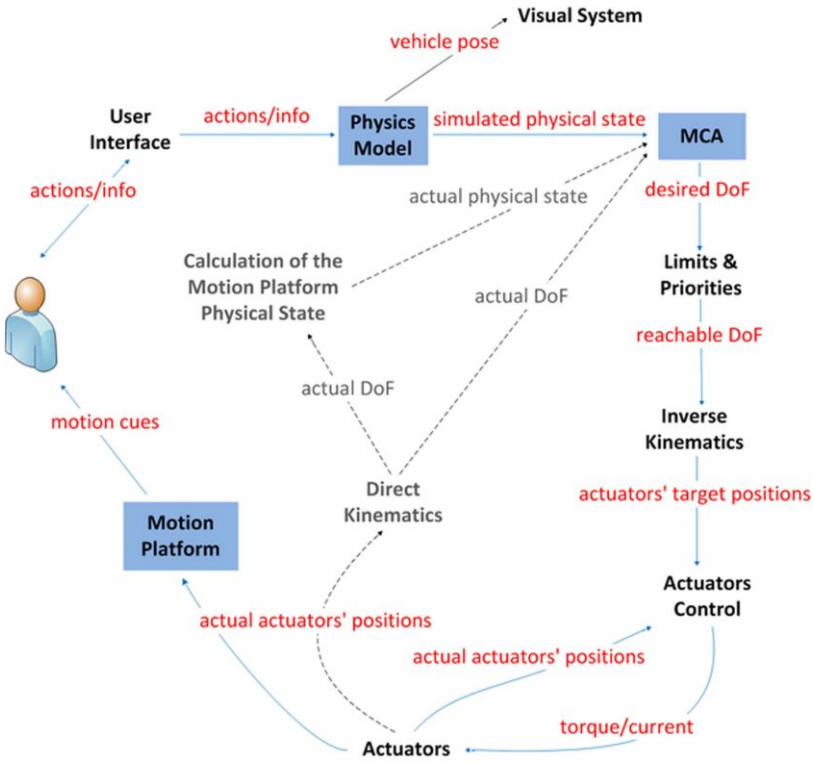

Figure 3.1 Motion cueing generation [21]

- The User Interface receives actions and control inputs from the operator (e.g., a pilot or driver) and these inputs are sent to the physics model for further processing.
- The Physics Model calculates the simulated physical state of the vehicle which includes parameters like translational and rotational acceleration.
- The result is a vehicle pose that is sent to the Visual System for rendering the simulated scene and to the MCA for generating motion cues. It takes into account the limits, reachable DOF and inverse kinematics of the simulator that converts desired motion into actuator-specific movements.
- The actuator control system sends commands to the actuators, adjusting torque and current as needed and they move the motion platform.
- The motion platform physically moves, replicating the intended motion cues. This feedback is sent back to the calculation system to ensure accuracy and stability.

3.2 Early Stages in Classical Washout Algorithm

In the late 1960s and early 1970s, Stanley F. Schmidt and Bjorn Conrad conducted pioneering research to enhance the realism of motion cues in piloted flight simulators. Their work culminated in two significant NASA reports [25, 26] and one with J.G. Douvillier in 1973 [45], laying the foundation for modern MCAs.

Their approach was to initially develop a mathematical framework to calculate the motion drive signals, ensuring that the simulator's motion closely mimicked actual flight dynamics in [25] and later the mathematical models were enhanced in [26]. In [45], they developed an advanced washout circuit design for multi-DOF simulators, introducing tilt coordination as a key component for motion cueing.

The key contributions of Schmidt and Conrad are:

- Introduced the concept of washout filters, which process motion signals to keep the simulator within physical constraints while providing realistic cues [25] and later improved the washout filter design [26].
- Presented methods to simulate sustained accelerations by tilting the simulator cabin, leveraging gravity to replicate continuous motion sensations [25] and discussed techniques to ensure tilt rotation remained below the perceptual threshold of the pilot [45].
- Introduced the motion scaling factor and concluded that the order of 0.3 to 0.4 is reasonable enough for MCAs for older platforms [45].

- Proposed coordinated control strategies for translational and rotational movements [26], and later multi-axis coordination strategies to enhance the overall realism of simulator motion [45].

3.3 Advancements of Classical Washout Algorithm

The evolution of CWA was significantly shaped by the works of L.D. Reid, M.A. Nahon, and their collaborators, who introduced refinements and optimizations to the CWA and its implementation in flight simulators.

In their study [23], they introduced a mathematical model that incorporated optimal control theory. They extended this work in [27] which proposed a systematic parameter selection method that considered human perception models. [46] provided empirical validation through pilot evaluations revealing that coordinated adaptive algorithms were preferred over conventional CWA due to their improved motion realism. In 1990 they published [47] which offered in-depth analysis of various CWAs used in flight simulators, focusing on their practical implementation and performance. Later in [34], integrating AWA with control mechanisms that enabled real-time adjustments to motion cues based on simulator state and pilot inputs, was studied. In 1997, L.D. Reid with P.R. Grant published [48] focusing on the critical process of tuning washout filters in flight simulators.

The key contributions of L.D. Reid, M.A. Nahon, and their contributors are:

- Proposed a washout algorithm based on OCA which utilized linear optimal control theory to design filters that considered human vestibular system dynamics, thereby improving motion cue accuracy [23]. Their work represented shift towards more sophisticated, perception-based motion cueing strategies.

- Emphasized the critical role of selecting appropriate system parameters, such as filter constants and scaling factors and methodologies for tuning these parameters [27]. They incorporated models of the human vestibular system into their parameter selection process.

- Assessed the impact of different washout algorithms on pilot performance and subjective perception. While performance metrics showed minimal variation across different algorithms, pilots generally favoured the coordinated adaptive algorithm (CAA), that adjusts the parameters in real-time based on pilot's input [46].

- The authors conducted a comprehensive comparison of three prevalent motion-drive algorithms, CWA, OCA and CAA. The study [47] incorporated pilot feedback to assess the subjective realism and effectiveness of each algorithm. It is concluded that all three algorithms studied could be adjusted to give good performance in terms of pilot ratings, despite the test results in [47], which indicated that the OCA performed the worst, the CAA yielded the best outcomes.

- [47] also highlighted that the CWA excelled in terms of simplicity, ease of adjustment, and execution speed, making it the preferred choice for scenarios requiring quick results.

- Later, in [34] they developed a versatile MCA capable of operating as a simple CWA with minimal parameters, allowing for quick adjustments to achieve optimal motion performance. This was a hybrid of CWA and AWA. The algorithm incorporates adaptive gain filters and various cost functions, enabling real-time adjustments based on pilot inputs.

- A supervisory software system was developed to enable quick and interactive testing and adjustment of MCAs. Additionally, it was enhanced to automatically adjust based on flight phases, conditions, or pilot preferences [34].

- The scaling factors were increased from the 0.3–0.4 range suggested in [45] to 0.7 due to improvements in platform design.

- Created a structured method for tuning washout filters, ensuring a systematic and effective approach to adjust motion in simulators. [48] emphasized the importance of minimizing motion cueing errors, highlighting the relationship between filter tuning and the accuracy of simulated motion cues.

3.4 Integrating MCA with Platform Design

The development of MCAs is inherently tied to the design of the motion platform, as its mechanical constraints and workspace limitations directly impact the feasibility of generating realistic motion cues. Earlier research studies often focused on MCA development alone, without considering the influence of platform design or workspace constraints [21]. To address this gap, Advani, Hosman, and Haeck introduced a co-design approach that integrates motion cueing system development with motion platform geometry, ensuring optimal performance in flight simulators.

Their key contributions are detailed in the works [49, 50]. They:

- Presented a workspace-based design approach, where the motion envelope of the simulator platform was matched with motion cueing requirements. Instead of simply limiting cues to fit the available platform motion, they optimized the motion base geometry to enhance motion reproduction accuracy [49, 50].

- Tailored motion cues based on pilot sensory feedback models, to refine the washout filter parameters.

- Proposed a method to define platform workspace based on representative flight manoeuvres, ensuring that motion cues remain within the platform’s physical limits.

In 2004, the study [55], introduced an innovative washout filter design tailored for 6-DOF motion simulators, particularly those with limited workspaces. Traditional washout filters often face challenges in such constrained environments, leading to potential inaccuracies in motion cueing. To address this, the authors proposed integrating real-time workspace boundary detection into the washout filter's operation.

By employing inverse kinematics, the system continuously monitors the simulator's workspace boundaries. This real-time data is fed into the washout filter, allowing it to adjust motion cues dynamically, ensuring they remain within the physical constraints of the simulator [55].

3.5 CDPR as Motion Platform

As this thesis primarily focuses on the implementation of MCA on CDPRs a comprehensive literature review was conducted to examine relevant studies in this area. This section explores existing research on CDPR-based motion platforms.

3.5.1 Cable Robot Simulator at the Max Planck Institute

The CRS at the Max Planck Institute (MPI) in Tübingen, Germany, serves as a platform for human motion perception research and virtual reality experiments. It features a mobile platform that is suspended by eight cables, each attached to fixed winches and guided through redirection pulleys (seen in Figure 3.2). The winch motors can be operated in either velocity-control or torque-control mode, allowing for precise motion adjustments. Each motor is equipped with rotary encoders, which provide real-time angular position data, while force sensors located at the redirection pulleys continuously measure cable forces. Additionally, the mobile platform is fitted with an inertial measurement unit, which delivers real-time data on accelerations and angular velocities, enabling accurate motion tracking and control [51]. The simulator could be adapted to a variety of applications. It has been extensively used in human perception and cognition research, flight simulation, and virtual reality applications, demonstrating the potential of CDPRs in advanced motion simulation environments [52].

The paper [10] provides a comprehensive system overview, covering aspects such as kinematics and system dynamics of the CRS. The integration of cable robot technology in motion simulators has facilitated the development of systems with large workspaces and high structural rigidity. The use of winches with a combined power of 384 kW enables the generation of cable forces up to 12,000 N, allowing for maximum accelerations of 14 m/s^2 while handling a payload of 500 kg [10]. However, in practical operation, the CRS is typically run at a maximum acceleration of 5 m/s^2 within a workspace of 4m × 5m × 5m, with rotational limits of ±40º for roll and pitch, and ±5º for yaw [52]. Some important parameters are summarised in Table 1.

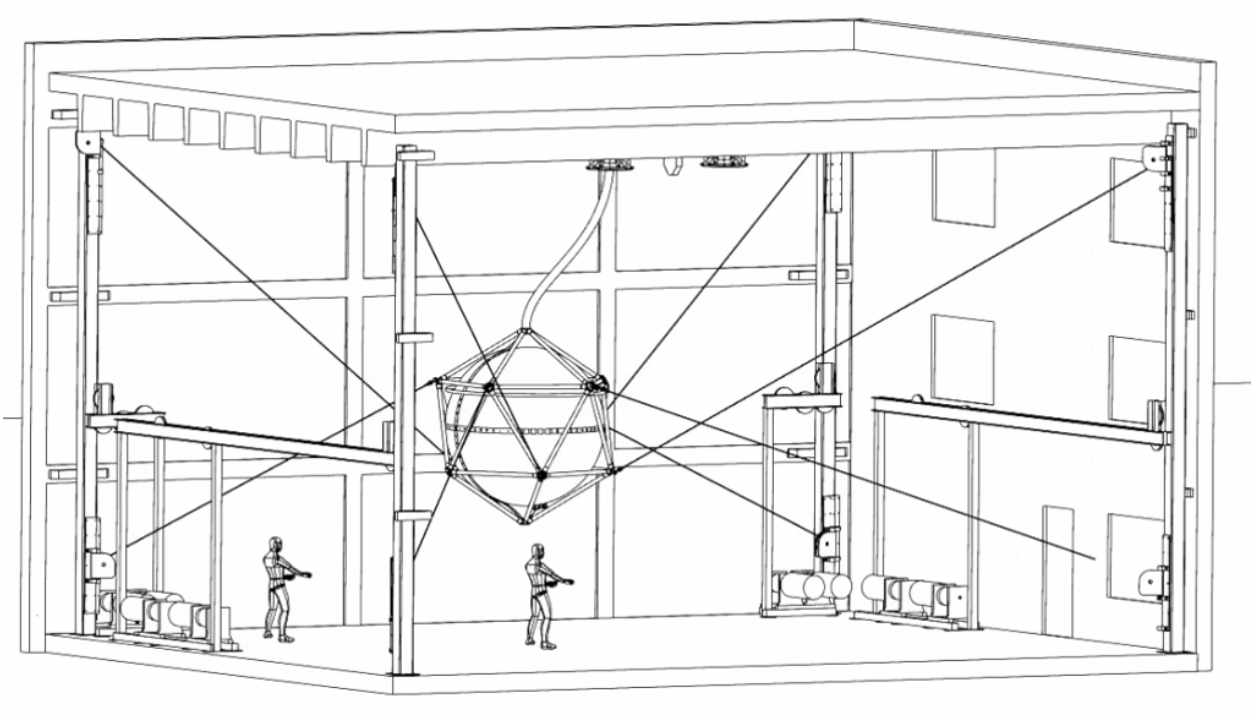

Figure 3.2: The CRS cross over configuration with 8 drive systems [10]

The cabin's icosahedron truss structure, constructed with carbon fiber rods and aerospace alloy nodes, minimizes weight while maximizing stability and the cabin volume [52]. [10] provided an overview of the first functional prototype designed for human transport and motion simulation, focusing on its dynamic properties.

Table 1: CRS Parameters adapted from [10]

Parameters	Values
Max. acceleration [translational, rotational]	[14 m/s², 100°/s²]
Max. speed [translational, rotational]	[5 m/s, 100°/s]
Safe translational workspace [x, y, z]	[4m, 5m, 5m]
Rotational workspace [roll, pitch, yaw]	[±40°, ±40°, ±5°]
Feasible cable tension [min, max]	[1000N, 14000N]
Max Payload	500kg

3.5.2 IPAnema 3 at Fraunhofer IPA

For the implementation of MCA, the CDPR IPAnema 3 which is developed by Fraunhofer IPA will be selected. The IPAnema 3 boasts a substantial operational workspace measuring 16 meters

in length, 11 meters in width, and 6 meters in height. Depending on its configuration, the robot can achieve speeds of up to 10 m/s and accelerations exceeding 10 g. The IPAnema 3 is engineered to handle payloads of up to 250kg [15]. Parameters are summarised in Table 2.

Table 2: IPAnema 3 Parameters adapted from [15, 53]

Parameters	**Values**
Max. Acceleration	> 10g
Max. speed	10 m/s
Operational workspace [x, y, z]	[16m, 11m, 6m]
Feasible cable tension [min, max]	[100N, 3000N]
Max Payload	250kg

The IPAnema 3 has been successfully utilized for both additive and subtractive manufacturing. It operates using a real-time capable CNC control system, which is easily programmable through the well-established G-Code standard (DIN 66025). Additionally, a programmable logic controller (PLC) enables seamless integration with existing industrial systems, enhancing its adaptability and ease of use [15].

Additional ropes can easily be inserted into the cable robot to increase safety against failure of individual components in safety-critical applications [54], such as safety of the person on the platform.

3.6 Summary and Key Takeaways

- A comprehensive review of the literature confirms that there is no universally accepted MCA, as each approach has its advantages and trade-offs. [47] emphasized that different MCAs can be fine-tuned for effective performance, with the CWA standing out for its simplicity and ease of adjustment, making it a preferred choice in many applications.

- Since this thesis focuses on the offline implementation of MCA, parameter optimization can be achieved using cost function optimisation for a prerecorded simulated trajectory. Unlike AWA or OCA, which emphasize real-time adaptation, this approach allows for precomputed optimizations tailored for a specific trajectory.

- [49, 50, 55] demonstrated that integrating platform geometry and workspace constraints into MCA is feasible. This is particularly relevant for CDPR-based systems, where the workspace dynamically shifts based on payload variations and cable force distributions.

- Currently, the CRS is the only motion simulator utilizing a CDPR platform. Given the availability of the IPAnema 3 at Fraunhofer IPA and its versatile capabilities, it serves as an ideal test platform for MCA implementation, transforming it into a functional motion simulator.

- Based on this extensive literature review, the selected MCA for this research is the CWA, which will be implemented using a customized method that integrates the workspace and geometry constraints of the IPAnema 3. Additionally, cost function optimization will be employed to fine-tune the CWA parameters, ensuring optimal performance in the simulation environment.

- This work lays the groundwork for the future implementation of MCA on the IPAnema 3 hardware. The focus is entirely on the software-based implementation, utilizing the IPAnema 3 simulation libraries available at Fraunhofer IPA for testing and validation.

4 METHODOLOGY AND IMPLEMENTATION

This chapter outlines the methodology for the software-based implementation of a customized CWA on the IPAnema 3 platform. A high-level overview of the process is provided here, with detailed explanations presented in the subsequent sections. Initially, the CWA was implemented with manually tuned parameters, without cost function optimization, to ensure the motion cueing algorithm functions correctly as a standalone process. This serves as a foundation for further enhancements. The process begins with the collection of datasets from simulated flight trajectories, specifically capturing translational accelerations and rotational rates, which serve as inputs to the CWA. The algorithm processes these inputs and generates the corresponding poses for each setpoint in the dataset.

To ensure feasibility, the force distribution on the IPAnema 3 cables is evaluated using these poses as input. If the computed forces exceed the allowable limits, parameters are iteratively adjusted through trial and error until the forces remain within the permissible range. The resulting validated poses form the trajectory for the IPAnema 3 platform, enabling it to accurately replicate the motion of the simulated flight trajectories. The overall process for manually tuned CWA is illustrated in the flowchart in Figure 4.1.

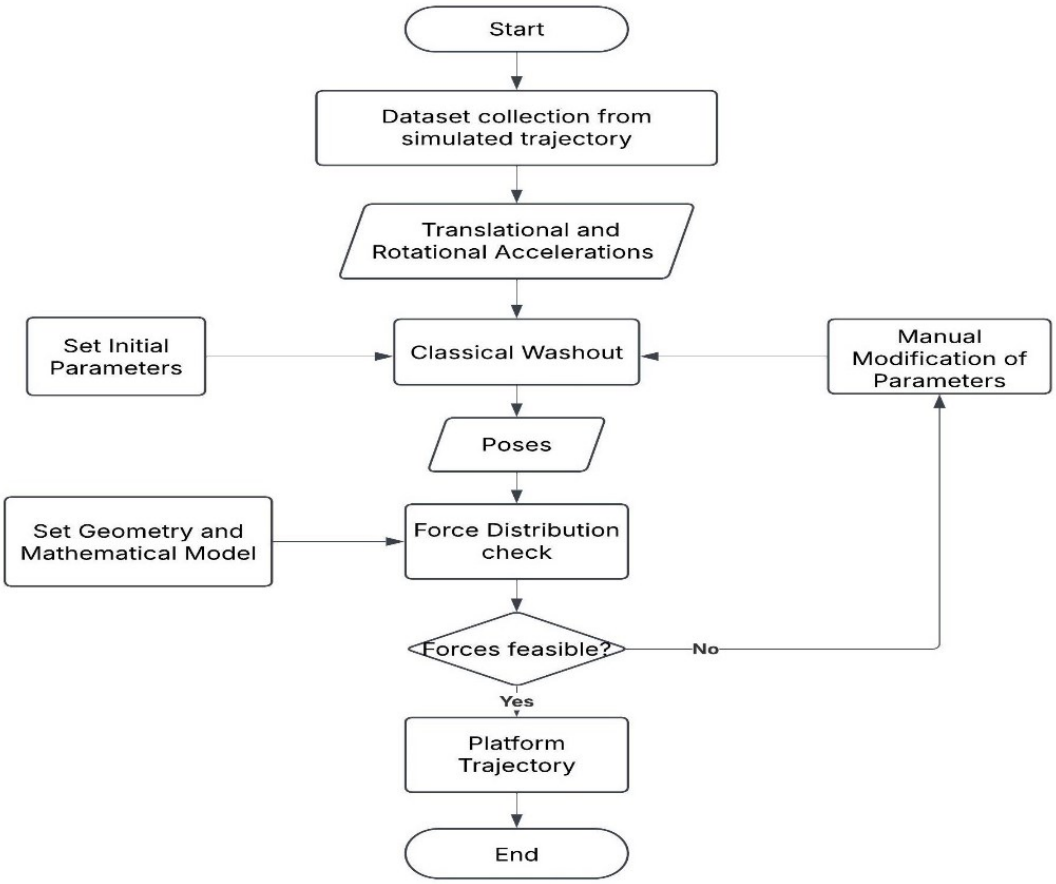

Figure 4.1: Flowchart of manually tuned CWA

Once the manually tuned CWA was developed for the IPAnema 3, the next step involved optimizing its parameters, primarily the damping factors. These were selected for optimization as they directly control the washout effect, whereas scaling factors primarily influence motion intensity, and cut-off frequencies only serve to separate fast and slow-motion components rather than affecting how motion dissipates over time. Manually adjusting scaling factors and cut-off frequencies is a more practical approach, as it provides greater flexibility in adapting to different motion scenarios without adding complexity to the optimization process. In contrast, damping factors significantly impact system response dynamics and determine how effectively the washout algorithm behaves, making them ideal candidates for optimisation. The cost function optimization process was then integrated with the manually tuned CWA.

The upper and lower bounds for the parameters were initially set based on practical or physically meaningful limits, while the starting values for the optimization were chosen from manually tested parameters. The cost factor is computed based on the forces before and after motion cueing. An optimization procedure is then performed to minimize this cost factor further. After each optimization step, the algorithm re-evaluates the force distribution. The process iteratively refines the parameters, ensuring a reduction in cost while maintaining feasible force limits.

If, at any stage, the optimized poses result in forces exceeding allowable thresholds, the algorithm terminates and retains the previously feasible parameters. On the other hand, when the cost is minimum, meaning the optimisation process reaches a point where successive iterations no longer result in a reduction in cost, it is considered to have converged, and the process stops. The final optimized parameters ensure an optimal balance between motion cue accuracy and compliance with force constraints. This optimisation process is illustrated in the flow chart in the Figure 4.2.

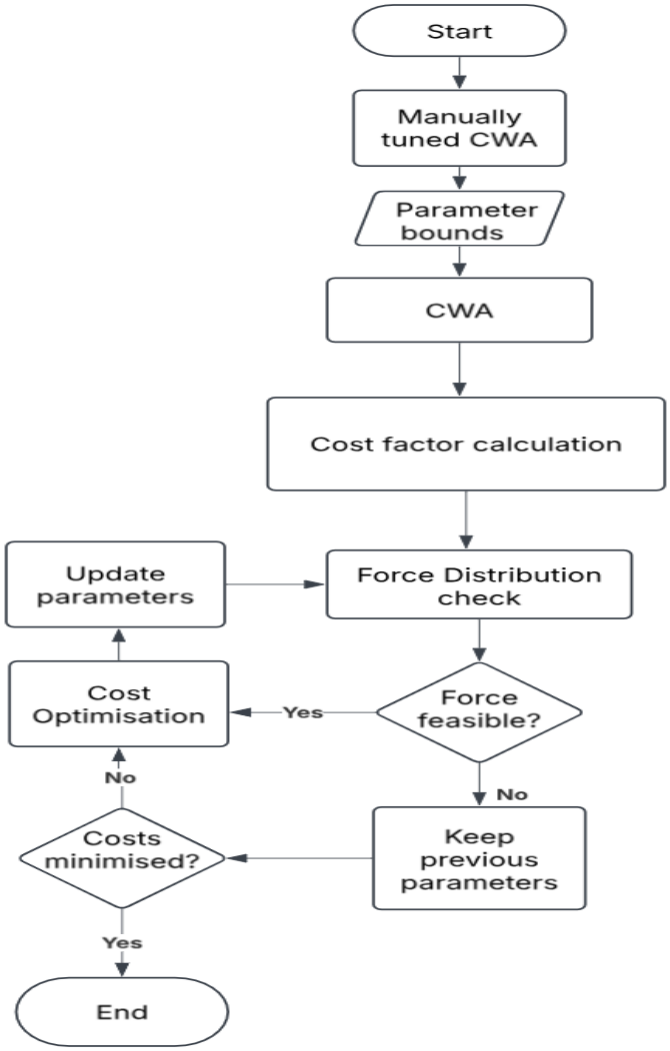

Figure 4.2: Flow chart for integration of Cost function optimisation with the CWA

4.1 Collection and Processing of Dataset

The goal here is to create a dataset containing translational and rotational accelerations of a simulated vehicle. After a thorough review of potential data collection methods and sources, FlightGear, an open-source flight simulator known for its realistic flight dynamics, comprehensive aircraft modeling, and accurate environmental simulation, was selected [56]. FlightGear generates high-fidelity aircraft trajectories, providing detailed translational and rotational acceleration data, making it an ideal platform for dataset collection in this thesis. Steps involved for this are explained in the following sub-sections.

4.1.1 Selection of Aircraft and Settings Configuration

The Cessna 172P, a default aircraft in FlightGear, was chosen for generating the trajectories. The following command was executed in the FlightGear settings terminal to initiate the simulation using the customized logging configuration file.

```
--config=C:\Users\mrf-ar\AppData\Roaming\flightgear.org\Export\logging_configuration.xml
```

This command instructs FlightGear to use the specified XML configuration file (see code snippet in A1), enabling the logging of selected flight parameters.

4.1.2 Configuration of Data Logging (XML File)

An XML-based configuration file was created to specify exactly which parameters from the simulation need to be logged. This configuration defines the parameters to be recorded and their output format.

The following properties are selected for data logging:

- **Translational Accelerations acting on the pilot (recorded in feet per second squared):**
  - Acceleration_X: Logged from property `/accelerations/pilot/x-accel-fps_sec`
  - Acceleration_Y: Logged from property `/accelerations/pilot/y-accel-fps_sec`
  - Acceleration_Z: Logged from property `/accelerations/pilot/z-accel-fps_sec`
- **Rotational Rates of the flight (recorded in degrees per second):**
  - Roll_rate: Logged from property `/orientation/roll-rate-degps`
  - Pitch_rate: Logged from property `/orientation/pitch-rate-degps`
  - Yaw_rate: Logged from property `/orientation/yaw-rate-degps`

4.1.3 Data Logging and Export

Upon running the flight simulation, FlightGear uses the above configuration to continuously log specified parameters time intervals, in this case it was roughly 100 milliseconds. The output is automatically recorded into a Comma-Separated Values (CSV) file, which contains the translational acceleration and rotational data in that forms the input dataset for the CWA.

Multiple flight trajectories were recorded, including trajectories with standard take-off and landing procedures characterized by minimal dynamic variations in translational and rotational rate, as well as trajectories involving rapid changes in translational and rotational accelerations.

4.2 Setting up the Classical Washout Algorithm

The CWA was implemented using Python, chosen due to its powerful numerical capabilities and extensive availability of libraries for scientific computing and signal processing (see code snippet in A2). This section provides a detailed overview of the software implementation of the CWA, highlighting how the collected flight simulation dataset is processed to generate motion cues and determine the final poses for the IPAnema 3 platform.

Prior to implementing the CWA, the translational and rotational acceleration data was imported from the previously collected dataset stored in a CSV file. The algorithm consists of three distinct channels—translational, rotational, and tilt coordination—each responsible for handling different motion components as per Reid-Nahon UTIAS implementation [23] which is detailed in Chapter 3. Butterworth filters were selected for both high-pass and low-pass filtering processes due to their maximally flat frequency response and effectiveness in avoiding distortion in the passband [57]. The libraries for the Butterworth filter can be imported in python by the following line:

```
from scipy.signal import butter, filtfilt
```

A block diagram representing the CWA used in this work, that comprises the combination of all the three channels is presented in the Figure 4.3.

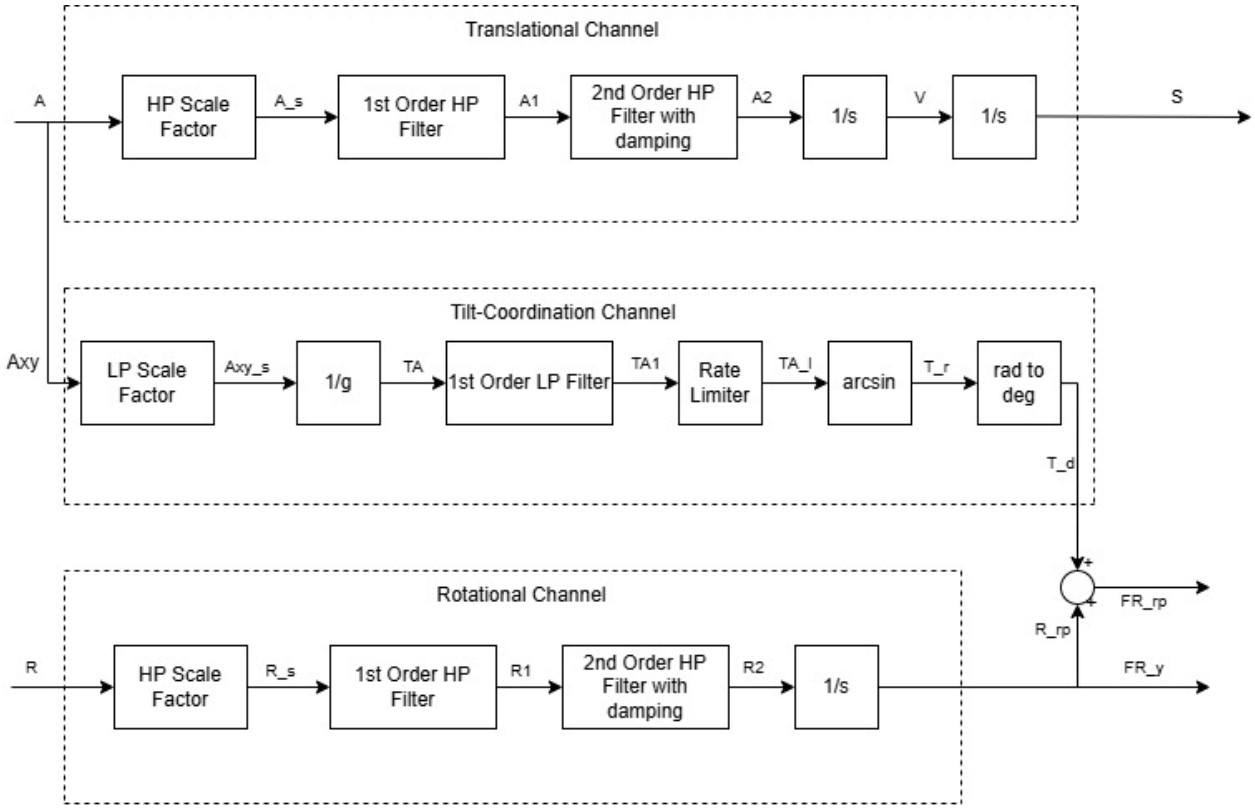

Figure 4.3: CWA Implementation block diagram

4.2.1 Translation Channel Implementation

The Translational Channel implementation is illustrated in the block diagram in the Figure 4.4.

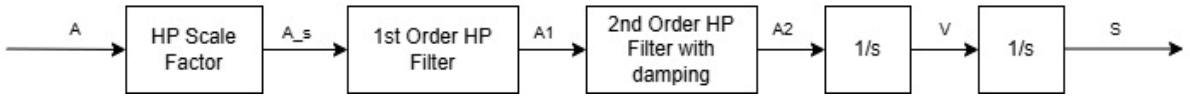

Figure 4.4: Translational Channel Implementation block diagram

The implementation of the translational channel begins by importing translational acceleration data from the collected dataset. The raw acceleration values along the x, y, and z axes, initially recorded in feet per second squared, are first converted to meters per second squared to ensure consistency with SI units. After this unit conversion, the accelerations are scaled using a manually adjustable scaling factor, which ranges from 0 to 1, where a value of 1 represents a 1:1 reproduction of the input translational accelerations. This is optional as it is useful when input accelerations are high, resulting in excessive forces that may exceed the platform's physical constraints. The choice of scaling factors is application-dependent and is adjusted based on the characteristics of the given trajectories.

Subsequently, the scaled input acceleration data passes through a first-order HP Butterworth filter, where the parameter HP cut-off frequency can be manipulated. The filter eliminates the low frequency components from the input signal and primarily generates the initial cue signalling the onset of translational motion. After the initial filtering, the signals pass through a second-order HP filter, which includes an adjustable damping parameter, and the HP cut-off frequency can also be modified. This filter smoothly controls the washout behaviour to ensure the platform returns to its neutral position Finally, the filtered acceleration signals are integrated twice—first to obtain velocities and then to determine the translational positions—thus generating the translational trajectory for the IPAnema 3 platform. The notations in the block diagram (see Figure 4.4) are:

- A is the raw translational accelerations from the dataset
- A_s is the scaled down translational accelerations.
- A1 is the translational accelerations after the 1st order HP filter.
- A2 is the translational accelerations after the 2nd order HP filter and washout.
- V is the resulting velocities.
- S is the resulting translational positions of the platform (sway, surge and heave)
- 1/s is the integrator block.

4.2.2 Rotational Channel Implementation

The Rotational Channel implementation is illustrated in the block diagram in the Figure 4.5.

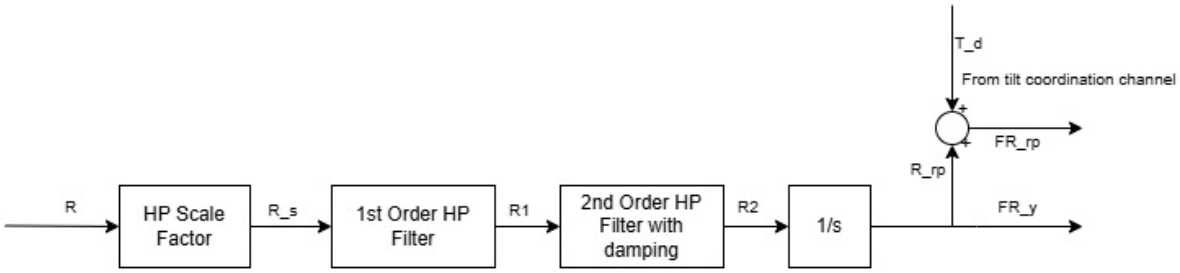

Figure 4.5: Rotational Channel Implementation block diagram

The rotational channel follows a methodology similar to that of the translational channel, processing the rotational rate data extracted from the dataset. Initially, the rotational rates from the CSV file are scaled using a manually adjustable scaling factor, ranging from 0 to 1, where a value of 1 corresponds to a direct 1:1 reproduction of the input rotational rates. This scaling is optional but can be beneficial for trajectories with significant rotational variations, where the raw rotational rates from the flight simulation may be too high for practical implementation.

Following scaling, these signals pass through a first-order HP Butterworth filter, where the HP cut-off frequency can be adjusted. HP filtering in the rotational channel removes LF drift and prevents continuous rotation of the platform, thus helping to maintain rotations within its physical limits. HP filters ensure that only rapid rotational changes are transmitted. Subsequently, the signals are further processed through a second-order HP Butterworth filter. This filter includes an adjustable damping parameter and allows modification of the cut-off frequency. The damping parameter provides smooth washout behaviour, returning the platform gently to its neutral rotational position when rotational inputs diminish.

After filtering, the processed rotational signals are integrated once to obtain the corresponding roll, pitch, and yaw angles. Finally, the roll and pitch angles from this rotational channel are combined with the tilt coordination roll and pitch angles to be computed. The notations in the block diagram (see Figure 4.5) are:

- R is the rotational (roll, pitch, yaw) rates in degrees per second.
- R_s is the rotational rates after the scale factor.
- R1 is the rotational rates after the 1st order HP filter
- R2 is the rotational rates after the 2nd order HP filter
- R_rp is the roll and the pitch angles from the rotational channel

- FR_rp is the final roll and pitch angles after adding the tilt angles from the tilt-coordination channel.

- FR_y is the final yaw angle from the rotational channel.

4.2.3 Tilt-Coordination Channel Implementation

The Tilt-Coordination Channel implementation is illustrated in the block diagram in the Figure 4.6

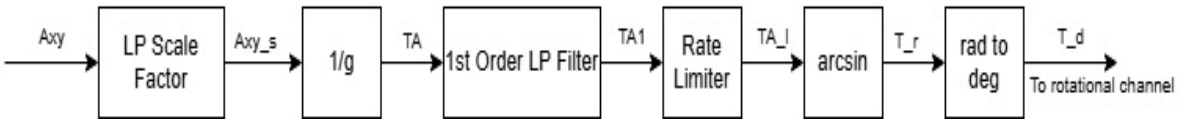

Figure 4.6: Tilt-Coordination Channel Implementation block diagram

The tilt-coordination channel aims to simulate sustained, LF translational accelerations, which are typically filtered out by the high-pass filtering in the translational channel. The approach is based on using gravity to mimic prolonged acceleration sensations.

Initially, the translational accelerations along the X and Y axes, expressed in meters per second squared, are first scaled down using a manually adjustable scaling factor ranging from 0 to 1. This factor is typically set to the same value as in the translational channel. However, if significant low-frequency components are present, it may be further reduced beyond the scaling applied in the translational channel. The vertical (Z-axis) acceleration is excluded from this channel, as gravitational effects do not directly influence yaw orientation.

The scaled X and Y acceleration signals are then divided by the gravitational acceleration constant ‘g’ (9.81m/s²) to convert the translational accelerations into equivalent tilt angular acceleration. This operation represents the fraction of gravity needed to simulate a sustained translational acceleration. These calculated values are subsequently processed through a LP Butterworth filter. The LP filter cutoff frequency is matched to the HP filter cutoff frequency used in the translational channel, ensuring a complementary and coherent frequency separation.

After filtering, a rate limiter is used to ensure these tilt angles remain within perceptual threshold (0.00524 rad/s²) converted from 0.3°/s² [30]. These angles represent the desired tilt rotational accelerations needed to replicate the sensation of prolonged translational acceleration. The tilt angles are computed using the arcsine function, which determines the inverse sine of a given value. This operation converts the ratio of the scaled acceleration to gravitational acceleration into an angle, expressed in radians. This operation converts the ratio of acceleration to gravity into a tilt angle (roll and pitch). These computed tilt angles are converted from radians to degrees. Finally, these tilt angles are passed to the rotational channel, where they are combined with the roll and pitch angles from that channel, to provide the final platform orientation of IPAnema 3. The notations in the block diagram (see Figure 4.6) are:

- Axy is the raw translational accelerations in X and Y directions from the dataset.
- Axy_s is the scaled down accelerations in X and Y directions.
- TA is the tilt angular accelerations.
- TA1 is the tilt angular accelerations after the 1st order LP filter.
- TA_l is the tilt angular accelerations after the rate limiter.
- T_r is the tilt angles in radians.
- T_d is the tilt angles in degrees.

After implementing the CWA in Python, the resulting poses are then saved into a CSV file for its integration with the Force Distribution check.

4.3 Integration Force Distribution with CWA

After the software implementation of the customised CWA, the next critical step is integrating it with the existing force distribution models specific to the IPAnema 3. Ensuring that the motion platform remains within the cable force limits is crucial for operational safety, and structural integrity. This section describes how the IPAnema 3's geometric configuration and force distribution calculations were integrated with the CWA to verify the feasibility of generated poses.

WireX is an open-source software initiative comprising multiple tools and libraries tailored specifically for CDPRs developed at Fraunhofer IPA. Among these components, WiPy acts as a Python-based interface that enables easy access to core computational functions provided by WireLib [58]. WireLib itself is a comprehensive class library offering essential functions and computational elements for scientific and engineering analyses related to CDPRs. By providing seamless integration with Python, WiPy enhances user accessibility, streamlines computational processes, and facilitates efficient design and analysis workflows in CDPR research and development. This can be accessed via a Gitlab repository [59].

This section details how force distribution calculations on the IPAnema 3 were performed on Python using the WiPy library from WireX using the poses generated by CWA. The steps are as follows:

4.3.1 Importing the WiPy Library

The initial step involves importing the WiPy library into Python. Specifically, the modules `Irobot`, `Ikin`, and `Iws` from WiPy3 are imported to facilitate the computation of CDPRs geometry, kinematics, workspace evaluation, and force distribution analysis. The following command is used:

- `import WiPy3`
- `from WiPy3 import Irobot, Ikin, Iws`

4.3.2 Defining IPAnema 3 Geometry

This step involves defining the geometric configuration and parameters of the IPAnema 3 platform within the WiPy library. The following procedures are executed sequentially:

- **Creating the Robot:** `WiPy3.createRobot()` initializes a new cable robot configuration within WiPy for subsequent geometry and workspace definitions.
- **Setting the Motion Pattern:** `Irobot.setMotionPattern(8, 5)` specifies the IPAnema 3 as having 8 cables and 5 here corresponds to having 3 rotational and 3 translational DOFs.
- **Defining the Proximal (Base) and Distal (Platform) Anchor Points:**

  `Irobot.setBase(i, x, y, z)` sets the geometry of each leg's(i) base. `Irobot.setPlatform(I, x, y, z)` sets the geometry of each leg's(i) platform.

  The values for the proximal and distal anchor points for IPAnema 3 are given in Table 3.

Table 3: Geometrical parameters of the IPAnema 3 adapted from [60]

Cable Index (i)	Proximal Anchor points in m [x, y, z]	Distal Anchor Points in m [x, y, z]
0	[7.6029, 5.5952, 3.5665]	[0.5488, 0.46, -0.48]
1	[7.5238, -5.5789, 3.5745]	[0.5488, -0.46, -0.48]
2	[-7.4614, -5.5621, 3.7536]	[-0.5488, -0.46, -0.48]
3	[-7.3114, 5.6113, 3.7408]	[-0.5488, 0.46, -0.48]
4	[7.6126, 5.5812, -0.7817]	[0.36, 0.7488, 0.48]
5	[7.5427, -5.5851, -0.8525]	[0.36, -0.7488, 0.48]
6	[-7.7641, -5.5694, -0.9384]	[-0.36, -0.7488, 0.48]
7	[-7.6959, 5.6008, -0.9036]	[-0.36, 0.7488, 0.48]

- **Kinematic and Elasticity Model Settings:** `Ikin.setKinematicsModel(0)` which represents a fixed kinematic model. The anchor points on both the base and platform are considered fixed, meaning the cable attachment points do not change positions during operation. `Ikin.setElasticityModel(1)` indicates that the linearly elastic model is chosen for the cables. In this model, cables are treated as having linear elasticity, meaning the cable elongation is directly proportional to the tension.

- **Force Limit Configuration:** `Iws.setForceLimits(100, 3000)` sets the minimum (100 N) and maximum (3000 N) allowable cable tension forces, which are critical to ensure the safe operation and accurate force distribution computation of the IPAnema 3 platform.

- **Setting the Wrench:** `Iws.setWrench(0, 0, -1000, 0, 0, 0)` sets the external wrench on the IPAnema 3 platform. Specifically, it applies external forces and torques. Here, Z is set at -1000N which represents the force in Z due to payload. This can be varied depending on the payload like a dummy in the platform or a person.

4.3.3 Calculation of Workspace

In this step, the workspace for the IPAnema 3 platform is calculated using WiPy's built-in routines. The following procedures are executed:

- **Setting Calculation Method:** `Iws.setMethod(4)` defines the calculation method and criteria for force distributions. It controls the method for workspace calculation of IPAnema 3. The method '4' refers to quadratic programming [59], which is a type of mathematical optimization problem where the objective function is quadratic, and the constraints are linear.

- **Defining Workspace Criterion:** `Iws.setWorkspaceCriterion(0)` sets the criterion for IPAnema 3 workspace evaluation. The criterion '0' refers to the force feasibility [59].

- **Number of Iterations:** `Iws.setIterations(6)` specifies the number of iterations for the workspace computation [59]. In this case it is 6 as it was sufficient for the calculations to converge on an accurate workspace solution. Higher the iterations would improve the accuracy slightly, but it results in more computation time.

- **Executing Workspace Calculation:** `Iws.calculateWorkspace()` initiates the calculation based on the previously defined parameters and constraints, resulting in computed workspace and associated cable forces for IPAnema 3 [59].

4.3.4 Defining and Calculation of Force Distribution

After defining the geometry and calculating the workspace of IPAnema 3, the next step involves calculating the force distribution on its cables using the WiPy library. The following procedures executed sequentially:

- **Importing Poses:** The previously computed CWA trajectory data (poses) is imported from a CSV file containing translational positions (x, y, z) and rotational orientations (roll, pitch, yaw) of the IPAnema 3 platform. These values serve as input to determine the required cable tensions for each pose.

- **Force Distribution Computation:** `Irobot.getForceDistribution(x, y, z, roll, pitch, yaw)` calculates the necessary tension forces for each of the eight cables of IPAnema 3 based on each specified pose. This computation considers the previously defined geometry of IPAnema 3 and the previously its calculated workspace.

  If the pose is feasible, the function returns the corresponding force values for each cable. If the pose is not feasible (i.e., at least one cable force exceeds or doesn't reach the allowable range 100N to 3000N), the function returns False, indicating that the platform cannot maintain the specified pose within its mechanical constraints.

- **Force Distribution Validation:** To validate the entire recorded trajectory, the force distribution is checked for each pose. The invalid pose coordinates and orientations are recorded. This aids in debugging of the CWA parameters, allowing trial-and-error adjustments before the optimization phase (see code snippet in A3). This helps to refine the algorithm for better force distribution. An example output for this is shown in Figure 4.7.

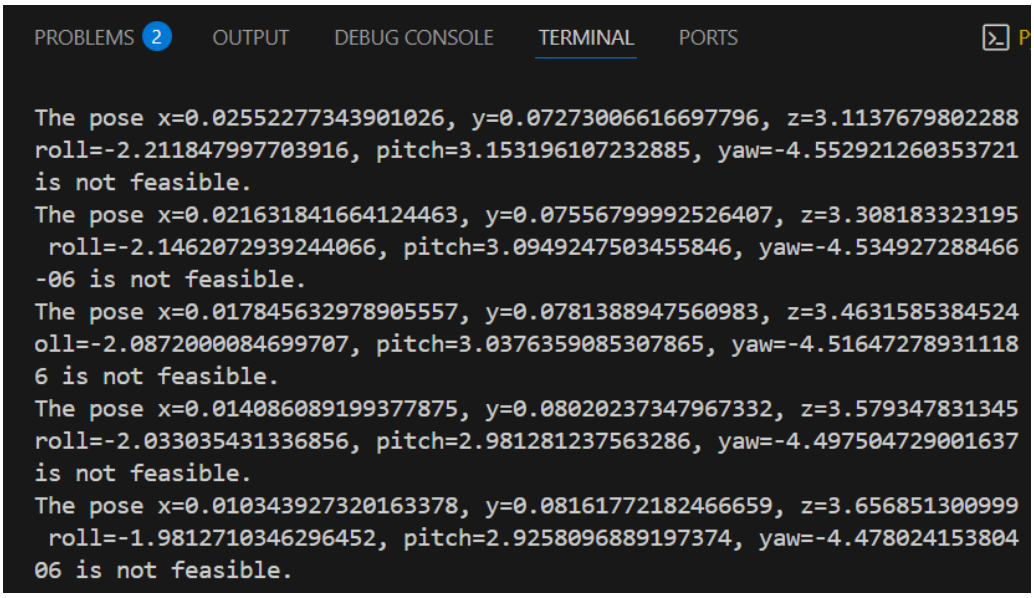

Figure 4.7: Example output for invalid poses

- **Output CSV Generation:** The calculated cable force distribution values for the entire trajectory are stored in a separate CSV file for further analysis. If a pose is deemed invalid, the corresponding force values are recorded as zeros.

To summarise, by computing the tension forces in each cable for every pose, the process helps verify whether the IPAnema 3 platform can maintain the intended motion without exceeding force limits or allowing slack cables. Additionally, this step provides valuable insights into parameter adjustments for the CWA, allowing for fine-tuning of scaling factors, cut-off frequencies, and washout damping ratios. This ensures that the customised CWA operates effectively while maintaining a valid force distribution, laying the groundwork for optimisation.

4.4 Cost Function-Based Parameter Optimisation for CWA

This section outlines the implementation of the optimisation process for the previously developed CWA, emphasizing its integration with force distribution constraints specific to the IPAnema 3 platform. In this study, the optimization focuses on tuning the damping ratios for the washout process in the platform's translational motion.

4.4.1 Defining the Cost Function

Initially the cost functions are defined for each translational axis (X, Y, and Z). These functions aim to minimize the difference between the specific forces before and after motion cueing, ensuring that the generated motion cues closely replicate the simulated trajectory. Each cost function follows a similar approach across the three translational axes. The optimisation process involves:

- **Initialising the Damping Parameter:** The initial damping ratios for washout are set based on manual adjustments through a trial-and-error approach, which is conducted prior to the optimization process.

- **Calculating the Original Forces:** The original forces acting on each axis before the CWA are computed using Equation (4):

$$F_{axis} = a_{axis} \times m \tag{4}$$

where:

- a_{axis} are the translational accelerations input to CWA along the respective axis.
- m is the total mass of the platform, (dividing the previously set force on Z axis, by ‘g’)

- **Calculating the Processed Forces:** The final poses generated by the CWA are differentiated twice to obtain acceleration values for each axis. However, the calculation of processed forces after motion cueing differs slightly for the Z-axis compared to X and Y, due to the influence of gravitation. For X and Y, LP Filtered accelerations are incorporated ensuring the tilt coordination effects are considered (see code snippet in A4).

  Final acceleration values for X and Y axes are given by Equation (5):

$$Final\_a_{x,y} = a_{x,y\_post} + (LP\_a_{x,y} \times g) \tag{5}$$

  where:

  - a_{x,y_post} are the translational accelerations from the output of CWA in X and Y
  - $(LP\_a_{x,y})$ are the LP Filtered accelerations from tilt-coordination
  - g is the gravitational constant

  Final acceleration values for Z axis are given by Equation (6):

$$Final\_a_z = a_{z\_post} + g \tag{6}$$

  where:

  - a_{z_post} are the translational accelerations after motion cueing
  - g is the gravitational acceleration constant

  The post motion cueing forces is computed by Equation (7):

$$F_{axis\_post} = Final\_a_{axis} \times m \tag{7}$$

  where:

  - $Final\_a_{axis}$ are the final translational accelerations along the respective axes
  - m is the total mass of the platform

  These forces are calculated for each data entry in the dataset used in the input and output from CWA.

- **Computing the Force Factor (Cost):** The force factor, which is the cost function, is then calculated as the absolute difference between the original and processed forces, normalized by the original force magnitude. A lower force factor signifies a more accurate replication

of the input forces in the output forces generated by the CWA. The force factor is calculated by slightly modifying the RAE method for normalisation by adding a small offset. This is shown in Equation (8):

$$Force\ Factor\ (Cost) = \frac{|\ F_{axis\_post} - F_{axis}\ |}{|\ F_{axis}\ | + Offset} \tag{8}$$

The offset is added to the force factor calculation for two key reasons:

1. Avoiding Division by Zero: Since some force values may be very small or even zero, adding an offset ensures that the denominator is never zero, preventing mathematical errors in the computation.

2. Mitigating Outliers in Small Forces: When dealing with small force values, division can amplify minor differences disproportionately. For example, if the initial force is 0.01 N and the post-processed force is 1 N, the resulting force factor becomes 99, suggesting a large deviation when, in reality, both values are relatively small and practically negligible. To address this issue, a small offset is added to normalize the ratio and prevent such distortions.

- **Defining the final cost of a trajectory:** The overall cost for each axis in a given trajectory is determined by calculating the mean force factor error across all data points in the dataset. This value represents how accurately the processed motion replicates the original forces, providing a measure of the deviation introduced by the motion cueing process.

In this section, the cost function has been formulated to quantify the deviation between the original and processed forces along the X, Y, and Z axes. By utilising a modified RAE with an offset, the function ensures appropriate scaling for both large and small force values, preventing distortions caused by minor magnitudes while enhancing cost stability. The final computed cost for each trajectory serves as the basis for optimizing the washout damping parameters in the next stage.

4.4.2 Optimisation of Washout Damping Parameters

This section will focus on the optimisation process, detailing how the washout damping parameters are adjusted iteratively to minimise the cost while ensuring force feasibility constraints within the IPAnema 3 platform. The optimisation is conducted for the washout damping ratios in the X, Y, and Z translational directions (see the programming logic in A5). The optimisation process that is setup in python follows these steps:

- **Defining the Optimisation Function:** The optimization is performed using the `minimize()` function from `scipy.optimize`. This performs numerical optimization by finding the minimum of a given objective function. In this work, the method used is

Limited-memory Broyden–Fletcher–Goldfarb–Shanno with Box constraints (L-BFGS-B). This is an optimization method used to find the minimum of a function while keeping the values within certain limits (bounds). It is efficient for large problems because it uses less memory and still works well without needing complex calculations [61].

- **Force Distribution Validation and Cost Evaluation:** A nested function, `validate_damping(),` is implemented inside the optimisation function to assess the feasibility of the given damping values before proceeding with cost computation. This function performs the following steps:

  1) The function assigns the manually tested damping value as the initial value to the corresponding washout damping parameter (x, y, or z)

  2) These damping values are applied, and the processed motion trajectory is generated. The computed platform positions (x, y, z) and orientations (roll, pitch, yaw) for IPAnema 3 are retrieved for each time step in the trajectory.

  3) Force distribution is checked for each pose in the trajectory. If any pose results in an invalid force distribution, the function assigns a penalty and terminates further evaluation. This ensures that the optimization later done will disregard damping values that lead to infeasible force distributions of IPAnema 3.

  4) If the forces in all cables are within feasible limits, the function returns the computed cost for the given damping. The goal is to minimize this cost function.

- **Minimising the cost:** The optimisation of cost function is carried out individually for each axis at the same time using the `minimize()` function. The steps are as follows:

  1) The function `validate_damping()` is passed as an objective function to the `minimize()` function.

  2) The upper bound and lower bound are set for each damping parameter in X, Y and Z. This ensure that the optimizer searches for values within the bounds.

  3) If `minimize()` finds a solution where `validate_damping()` returns a valid cost value, the optimized damping parameter is stored. If no feasible solution is found, the previous manually tuned damping value is retained.

  4) The newly optimized damping values are assigned to the washout algorithm for the further evaluation and displayed on the terminal.

This section presented a structured approach to refining the damping parameters of the CWA by integrating cost function minimization with force distribution validation. A two-step validation

was applied to ensure that the optimized damping values not only minimized the cost function but also maintained force feasibility across all motion cues. If the force distribution check returned invalid results, the optimizer adjusted the parameters further until a valid solution was obtained. This approach ensured that the CWA parameters were optimized while keeping the IPAnema 3 platform within its physical constraints.

5 EVALUATION AND RESULTS

This chapter presents the evaluation of the implemented customized CWA on the IPAnema 3 with integration of force distribution and its optimization. In this evaluation, the parameters scaling factors, cut-off frequencies and damping factors for rotational rates were manually adjusted and the damping factors for translational accelerations were first manually adjusted and later optimised using cost function optimisation keeping the cable forces within feasible limits. Various test cases, including different payloads and trajectory types, were examined to analyse the CWA.

5.1 Evaluation of Customised CWA

This section evaluates the functionality of the customized CWA when integrated with force distribution constraints. To conduct this assessment, a normal take-off and landing trajectory with a stable flight path and minimal rotational rate variations was chosen as a representative test case. To maintain consistency in the evaluation, scaling factors, HP and LP filter cut-off frequencies were fixed and set to a defined value throughout the testing phase. Force in Z direction was set to -1000N, which represents the force in Z due to payload which would be 101.93kg.

The evaluation process involved manually tuning the washout damping parameters until the force distribution produced feasible force values across all cables. This initial testing phase was to ensure that the CWA generated appropriate motion cues while maintaining valid force constraints. Once the CWA was validated, further refinements and optimization would be performed in the next phase.

5.1.1 Evaluating Translational Accelerations

The parameter values used for evaluating translational accelerations in the CWA are presented in Table 4. These values were determined after multiple tests runs with varying parameters. Through a trial-and-error approach, these specific values were identified as force-feasible.

Table 4: Parameters to evaluate translational accelerations

Parameters	Values
Scaling factors [X, Y, Z]	[0.8, 0.8, 0.8]
HP cut-off frequencies in Hz [X, Y, Z]	[0.05, 0.05, 0.05]
LP cut-off frequencies in Hz [X, Y, Z]	[0.05, 0.05, 0.05]
Washout damping factor [X, Y, Z]	[0.001, 0.005, 0.001]

- **Effect of Cutoff Frequencies on Translational accelerations:**

To analyse the behaviour of the HP and LP filters for translational accelerations within the CWA, acceleration data is visualized across multiple stages of filtering. Figure 5.1 and Figure 5.2 present a comparative analysis of the input accelerations, HP-filtered accelerations, and LP-filtered accelerations for the X and Y axes of the platform, respectively.

Each figure contains three plots:

1. **Raw acceleration data** – The input accelerations fed into the CWA.
2. **HP-filtered accelerations** – The HF components retained after filtering.
3. **LP-filtered accelerations** – The LF components retained after filtering.

The HP-filtered accelerations (highlighted in green) show rapid fluctuations around a mean value, indicating the extraction of short-duration, high-frequency motion components. These variations contribute to the translational movement of the IPAnema 3 platform.

Conversely, the LP-filtered accelerations (highlighted in red) retain slower-changing trends, such as gradual slopes and sudden peaks, representing long-duration, low-frequency motion cues. These are primarily processed in the tilt-coordination channel to generate platform tilt.

The separation of these components confirms that the filtering mechanism is functioning as intended, effectively isolating motion cues that are then used by different channels within the CWA.

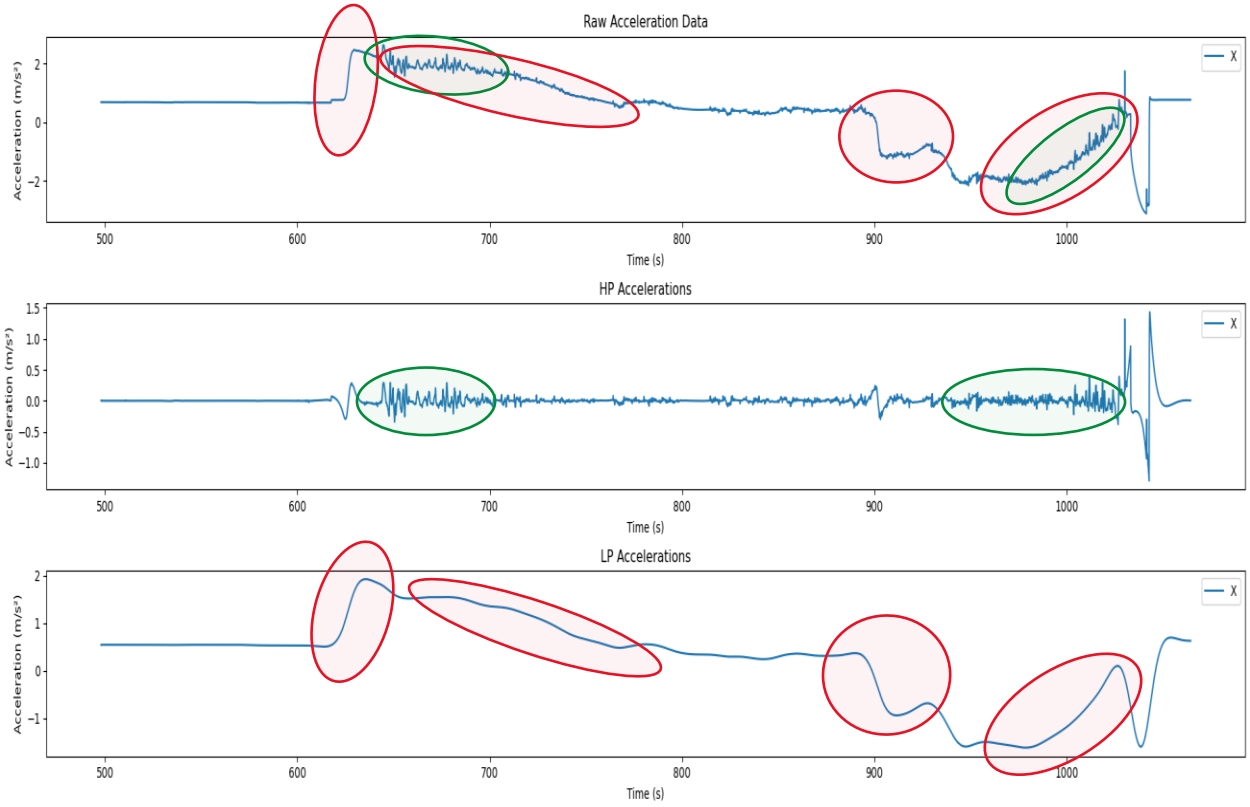

Figure 5.1: Comparison of input accelerations to filtered accelerations X axis translation

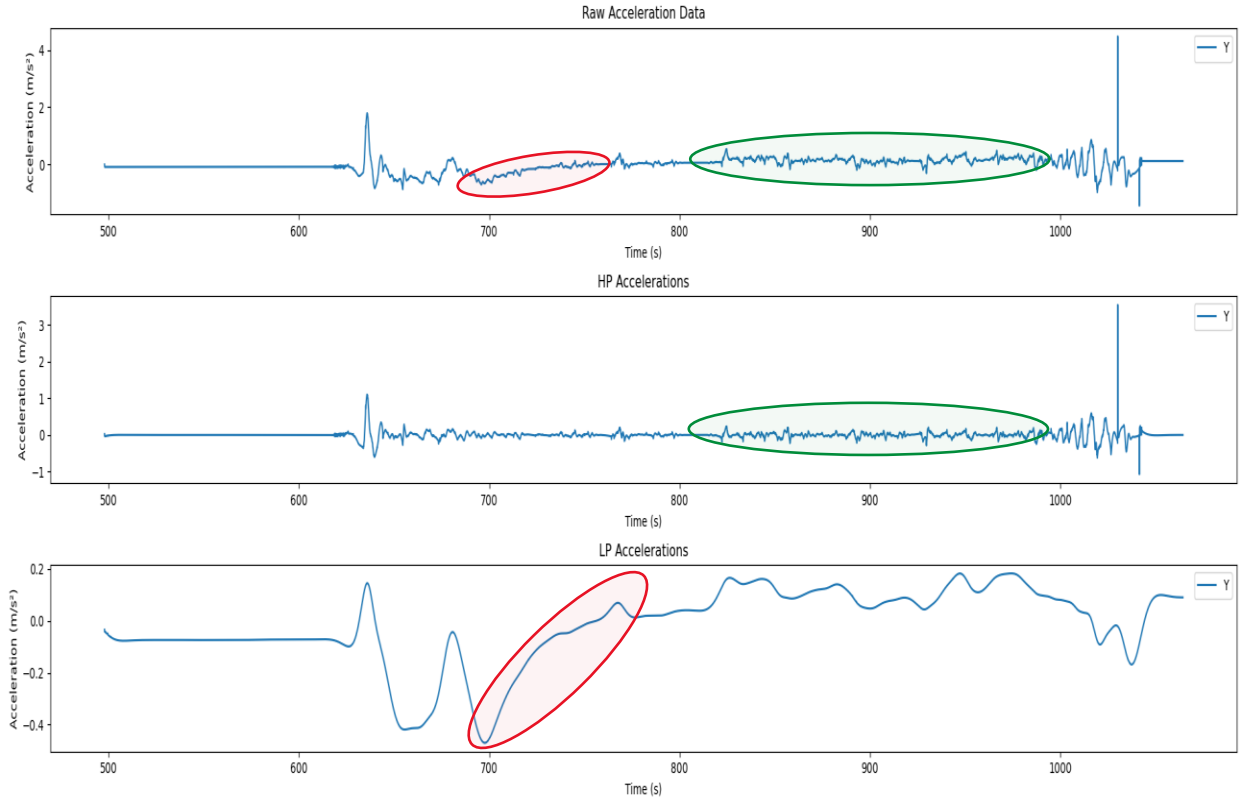

Figure 5.2: Comparison of input accelerations to filtered accelerations for Y axis translation

The effect of HP filtering was also observed by computing positions through direct double integration of the input acceleration values. These positions represent how the platform would behave without any filtering or washout damping. As seen in the plot in Figure 5.3 the resulting positions exhibit extremely large values, especially in the X-axis, reaching tens of thousands of meters. This clearly demonstrates the influence of LF components in the input accelerations, leading to unrealistic motion that cannot be replicated by the IPAnema 3 platform.

To mitigate this, the plot in Figure 5.4 shows the positions obtained after removing the LF components using HP filters. The resulting positions, derived from the HP filtered accelerations without applying washout damping, are significantly more constrained and appear more realistic. However, a noticeable drift remains in the Y axis, which is likely due to external influences such as wind forces in the FlightGear simulation or by the specific flight dynamics of the aircraft model used in the simulation. Factors such as rudder inputs, slight asymmetric thrust from the engine, or propeller-induced slipstream effects may cause minor lateral movement and not fly perfectly in X direction, contributing to the observed deviation in the Y axis. This highlights the need for washout damping to further stabilize the motion, ensuring that any remaining drift is eliminated and that the final poses remain feasible for execution on the IPAnema 3 platform.

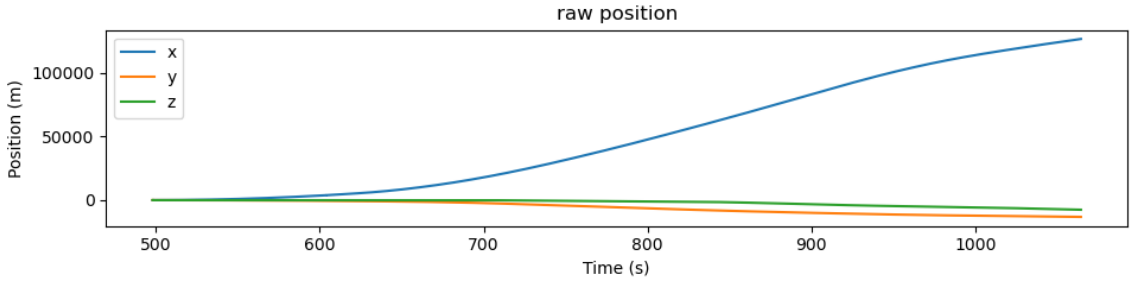

Figure 5.3: Positions obtained directly from Input accelerations to CWA

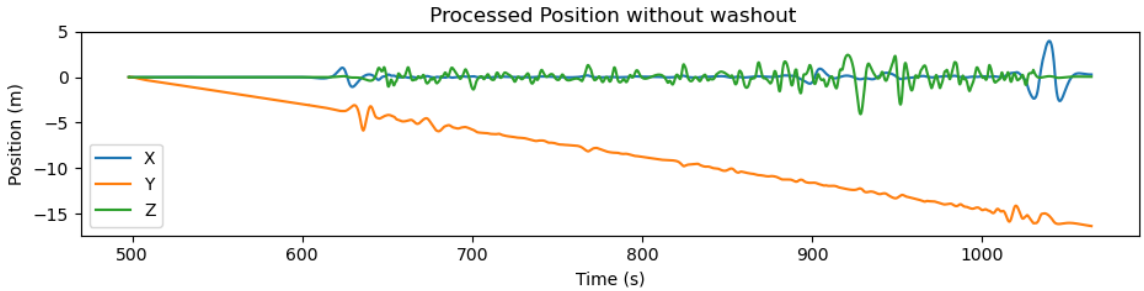

Figure 5.4: Positions obtained from the HP Filtered accelerations without washout damping

- **Effect of Washout Damping Factors in Translation Channel**

The HP filtered accelerations undergo a washout process to generate the final platform poses, achieved by introducing washout damping. The damping parameters for the translational channel are listed in Table 4. The damping is applied using an exponential decay function, which gradually

attenuates HF components, ensuring a controlled return to the neutral position. The plot in Figure 5.4 illustrates the processed positions without washout damping, where a noticeable drift along the Y-axis is observed. To mitigate this, a higher damping factor was chosen for the Y-axis to counteract the drift.

After applying the washout damping, the resulting translational trajectory of the IPAnema 3 platform is shown in the plot in Figure 5.5. It can be observed that the drift in the Y-axis is now progressively returning to a neutral position. Additionally, the final pose values for the X and Z axes exhibit a controlled reduction, signifying that the washout damping for translational accelerations in the customised CWA works as intended, ensuring that the simulated platform behaviour remains within feasible limits. The velocity was also analysed to ensure that it remains within the operational limits of IPAnema 3, which has a maximum allowable velocity of 10 m/s (see Table 2). The velocity plot, shown in Figure 5.6, confirms that the translational velocities remain well below this threshold for the selected trajectory.

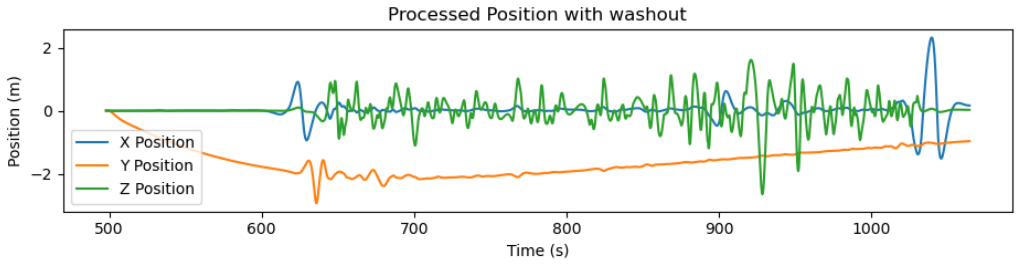

Figure 5.5: Final translational poses of the IPAnema 3 platform

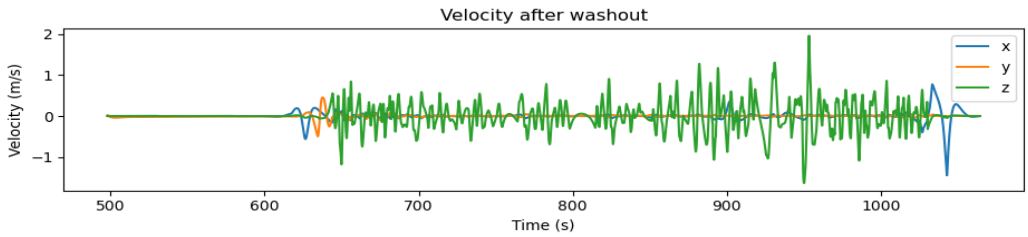

Figure 5.6: Final velocities of the IPAnema 3 platform

5.1.2 Evaluating Rotational Rates

The force feasible parameter values utilized for assessing rotational rates in the CWA are outlined in Table 5. Additionally, this section examines and evaluates the rotation angles derived from the tilt coordination channel.

Table 5: Parameters to evaluate rotational rates

Parameters	Values
Scaling factors [Roll, Pitch, Yaw]	[0.8, 0.8, 0.65]
HP cut-off frequencies in Hz [Roll, Pitch, Yaw]	[0.05, 0.05, 0.05]
Washout damping factor [Roll, Pitch, Yaw]	[0.005, 0.007, 0.01]

- **Effect of Cutoff Frequencies on Rotational Rates:**

Given that the trajectory used in this analysis featured a smooth take-off and landing with minimal rotational movement, it was found that setting a very low HP cut-off frequency still resulted in a force-feasible solution for the IPAnema 3. Since the input data lacked rapid rotational changes or significant HF rotational components, it was observed that increasing the HP cut-off frequency had little impact. The plot in Figure 5.7 illustrates the input rotational rates, while the plot in Figure 5.8 presents the HF components of the rotational rates, where no significant difference is observed other than scaling reductions.

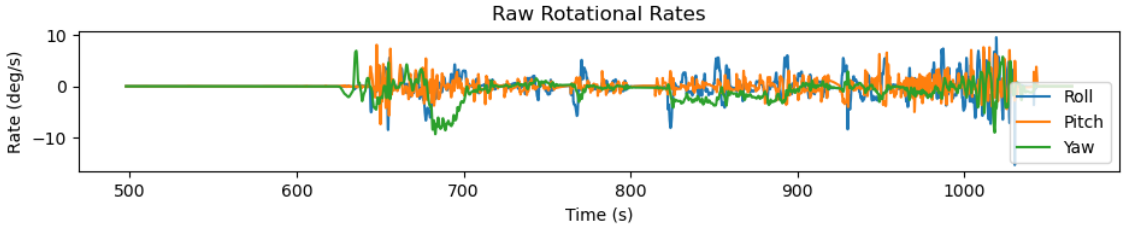

Figure 5.7: Input Rotational Rates to the CWA

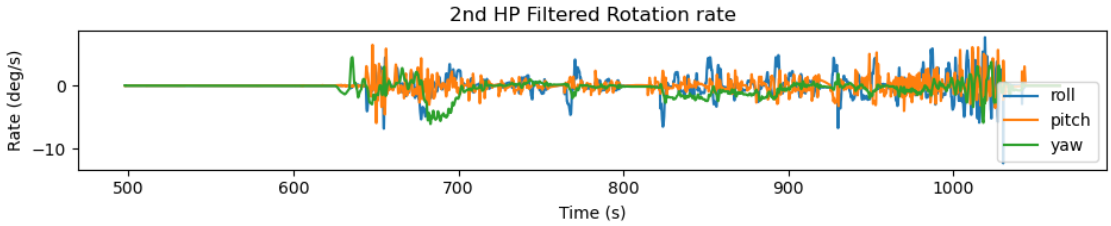

Figure 5.8: Rotational Rates after HP Filtering

In the simulation, the flight would take off and land on the same runway, resulting in a total yaw angle of 360° along the trajectory. This is evident in Figure 5.9 where the input rotational rates are integrated to obtain the rotational angles of the flight simulation. Due to this, test runs indicated the need to scale down the yaw angle slightly more than the roll and pitch angles. The HP-filtered rotational rates were then integrated before applying washout to obtain the rotation angles, as shown in the plot in Figure 5.10. It was observed that the yaw rate remained relatively high, highlighting the necessity of washout damping.

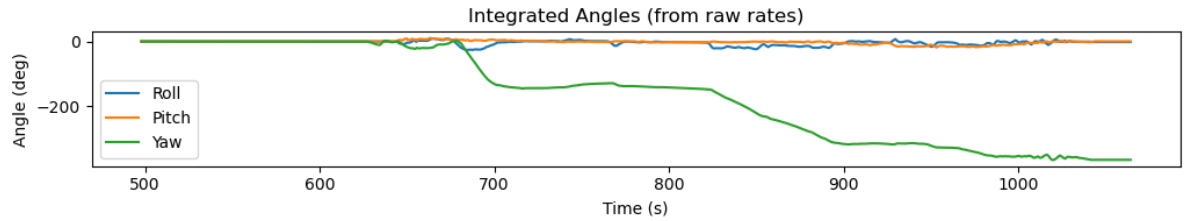

Figure 5.9: Rotational angles obtained directly from input rotational rates

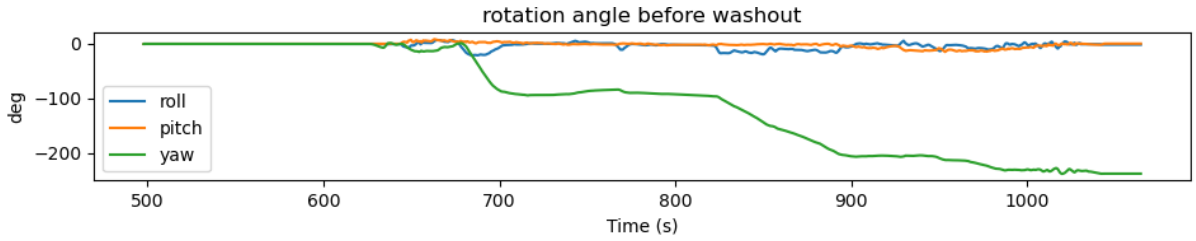

Figure 5.10: Rotational angles obtained from HP Filtered rotational rates without washout

- **Effect of Washout Damping Factors in Rotational Channel**

The HP-filtered rotational rates undergo a washout process to produce the rotational angles, that will be later added with the tilt angles. The damping parameters for the rotational channel are provided in Table 5. Similar to the translational channel, an exponential decay function is applied to gradually attenuate high-frequency components, ensuring a controlled return to the neutral position. The plot in Figure 5.10 presents the processed rotational angles without washout damping, where a significantly high yaw angle is noticeable. To address this, a higher damping factor was applied to the yaw axis compared to roll and pitch.

After applying washout damping, the resulting rotational angles, before incorporating the tilt angles, are shown in the plot in Figure 5.11. It can be observed that the yaw angle has been significantly reduced and is gradually returning to a neutral position. Additionally, the roll and pitch angles also demonstrate a controlled decrease, which is essential since these angles will later be combined with the tilt angles. This confirms that the washout damping for rotational rates is operating effectively as intended.

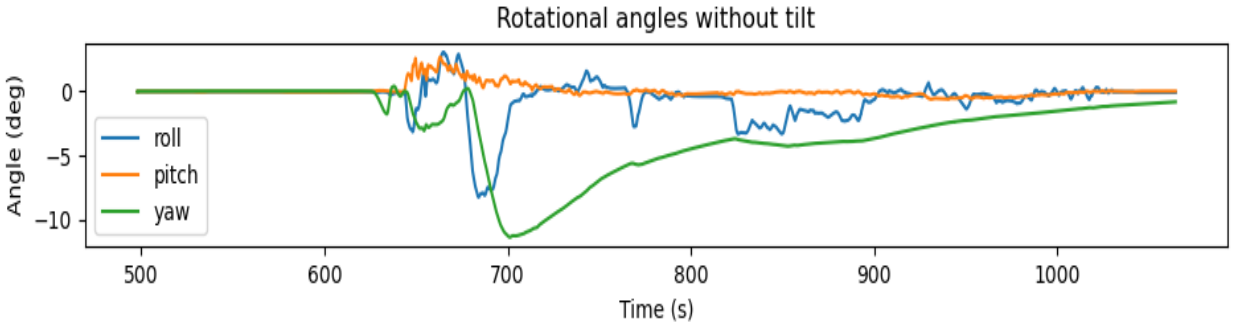

Figure 5.11 Washout rotational angles without tilt

- **Summing Tilt Angles in Rotational Channel**

The final step in determining the orientation of the IPAnema 3 platform involves adding the tilt angles from the tilt coordination channel to the rotational angles produced by the rotational channel. The tilt angles are derived from the low-frequency components of the translational accelerations—specifically, roll from the Y-axis and pitch from the X-axis. As shown in Figure 5.12, the tilt rate remains below 2°/s, which is safely under the perceptual threshold of 3°/s [30]. Hence, the use of rate limiter is not needed for this trajectory. The resulting tilt angles are illustrated in Figure 5.13. When these angles are combined with the roll and pitch from the rotational channel, the final platform orientation is achieved, shown in the plot in Figure 5.14. The influence of the tilt angles on the overall orientation is clearly evident from the observation of this plot, emphasizing the significant role of tilt coordination in the effectiveness of the CWA implementation.

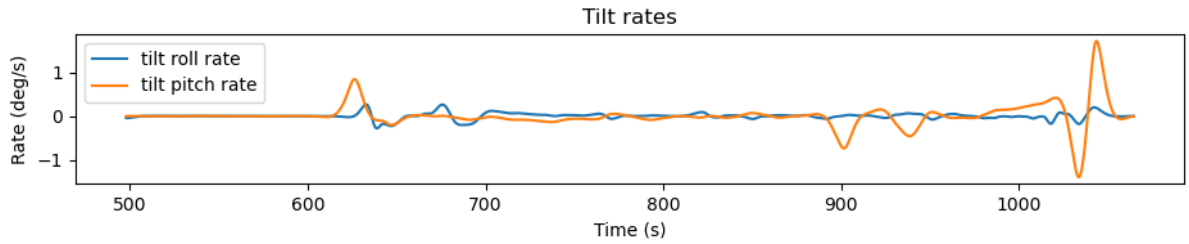

Figure 5.12: Tilt rates obtained from the tilt coordination channel

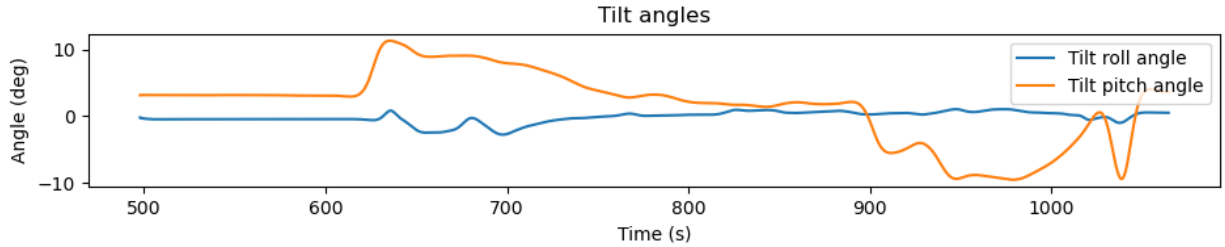

Figure 5.13 Tilt angles obtained from the tilt coordination channel

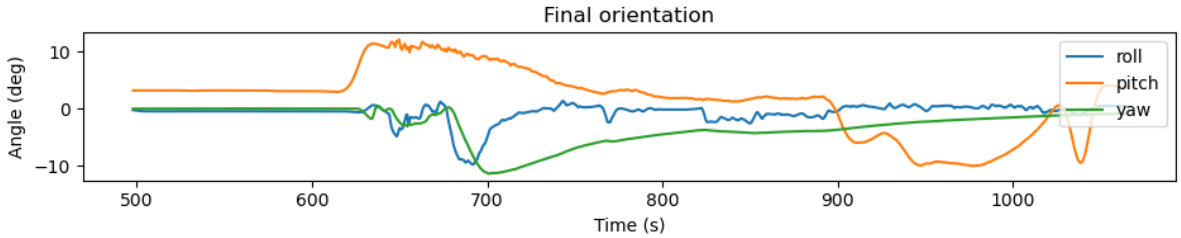

Figure 5.14: The final orientation of the IPAnema 3 platform

The customized CWA was successfully implemented and evaluated using a representative flight trajectory. The results demonstrated that the CWA effectively separated LF and HF motion components through filtering, while the washout damping played a crucial role in stabilizing both translational and rotational outputs. Force feasibility was achieved through manual tuning of the

damping parameters. The tilt coordination channel proved essential in replicating sustained linear accelerations through orientation angles. Overall, the results confirm that the CWA is functioning as intended, providing a strong foundation for optimisation.

5.2 Evaluation of Cost Function Optimisation

With the functionality of the customised CWA validated through its results, the next step involved improving the performance of the CWA by optimising the washout damping parameters for its translational channel. This section presents the evaluation of this optimisation process, and results of cost function minimization while maintaining force distribution feasibility on the IPAnema 3 platform.

5.2.1 Force Factors as Cost Metric

- **Input and Output Forces**

A key objective in motion cueing is to ensure that the forces experienced on the motion platform closely replicate those of the simulated environment. To evaluate this, force comparisons were performed between the raw forces derived from the flight simulation and the forces obtained after processing through the CWA.

Figure 5.15 presents plots comparing the raw forces with the post-motion-cueing forces along the X, Y, and Z axes for the flight trajectory for manually tuned washout parameters. As observed, differences between the two sets of forces are clearly visible, especially during periods of high dynamic activity. These deviations are mainly attributed to the effects of filtering, damping, and signal scaling within the washout process. The huge spike at the end corresponds to a hard landing and braking. Despite these discrepancies, the trends in the post-cueing force signals largely follow those of the raw forces, indicating that the general dynamics are preserved.

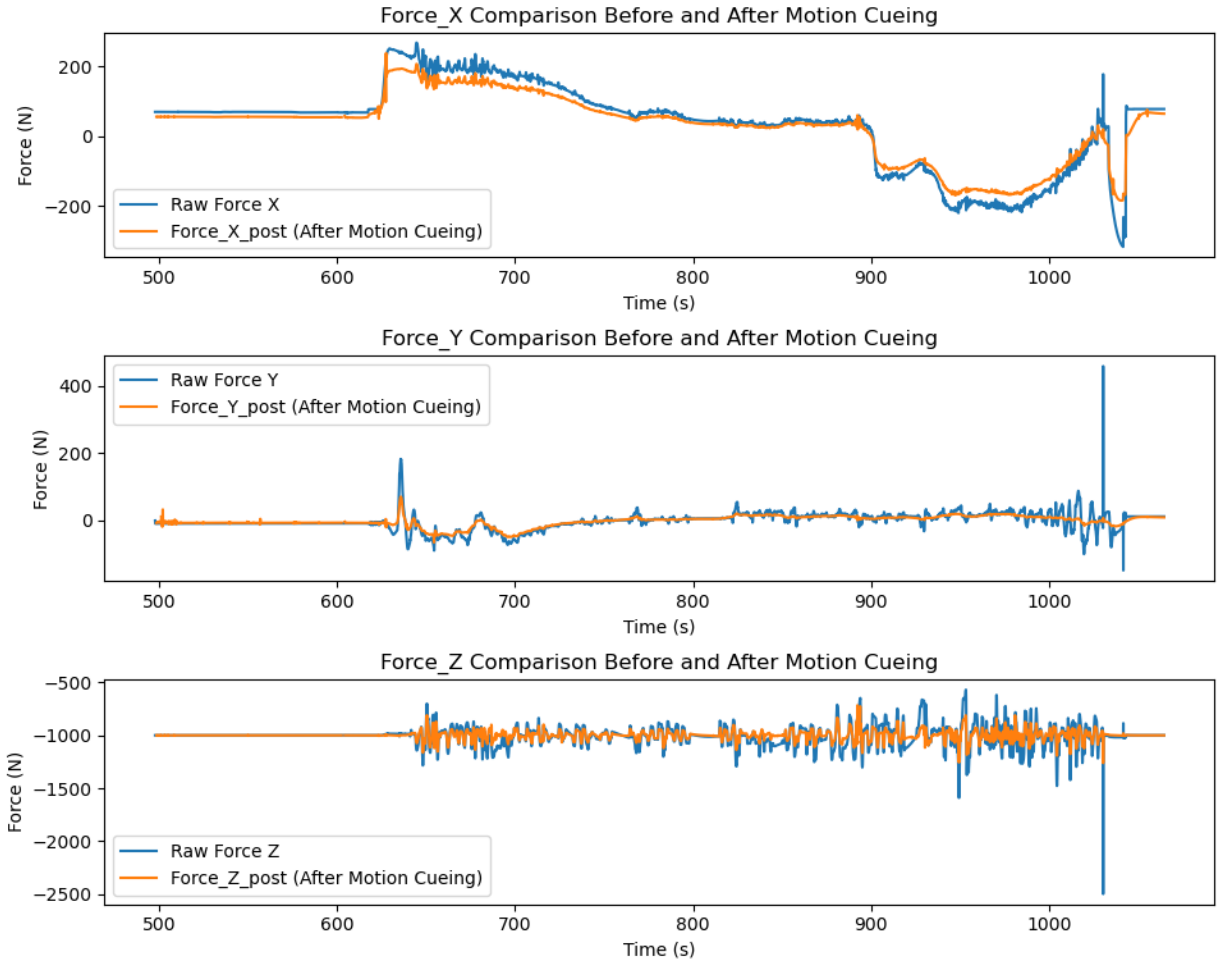

Figure 5.15: Comparison of input forces to the forces acting on the platform

- **Outlier Mitigation through Normalisation**

In the cost function design, the Force Factor was computed using the RAE method. Initially, infinite values were neglected during the computation. However, this approach alone did not eliminate outliers effectively. As shown in the plots in Figure 5.16, when the Force Factor was calculated without any denominator offset, the results contained extreme outliers, particularly in the Y axis. These spikes heavily distorted the mean Force Factor, which in turn affects the cost function optimisation process.

To address this, a small constant offset (5 N) was introduced in the denominator during Force Factor computation. This significantly suppressed large outliers, as shown in the plots in Figure 5.17. This modification resulted in a more stable calculation of the mean force factor, allowing the optimisation algorithm to perform more reliably.

Notably, the Y-axis exhibited higher variance in both versions due to its higher washout damping, leading to amplified motion cues. Conversely, the Z-axis naturally presented lower Force Factor values due to gravity's dominance along that direction, especially in level flight phases.

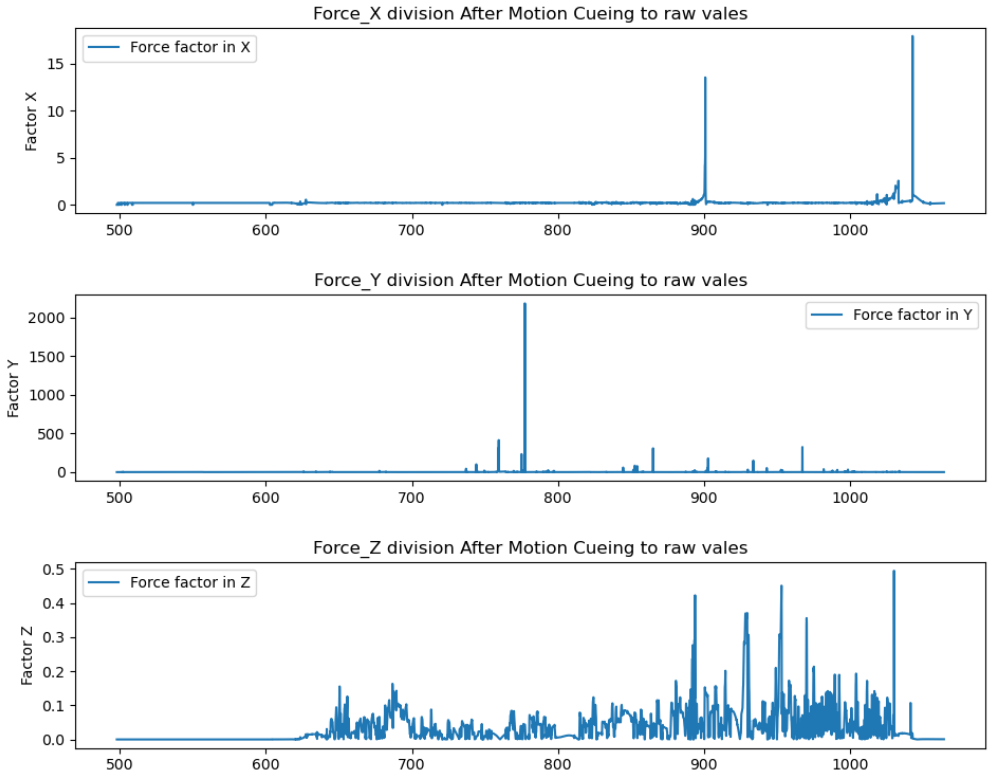

Figure 5.16: Force Factors without Offset

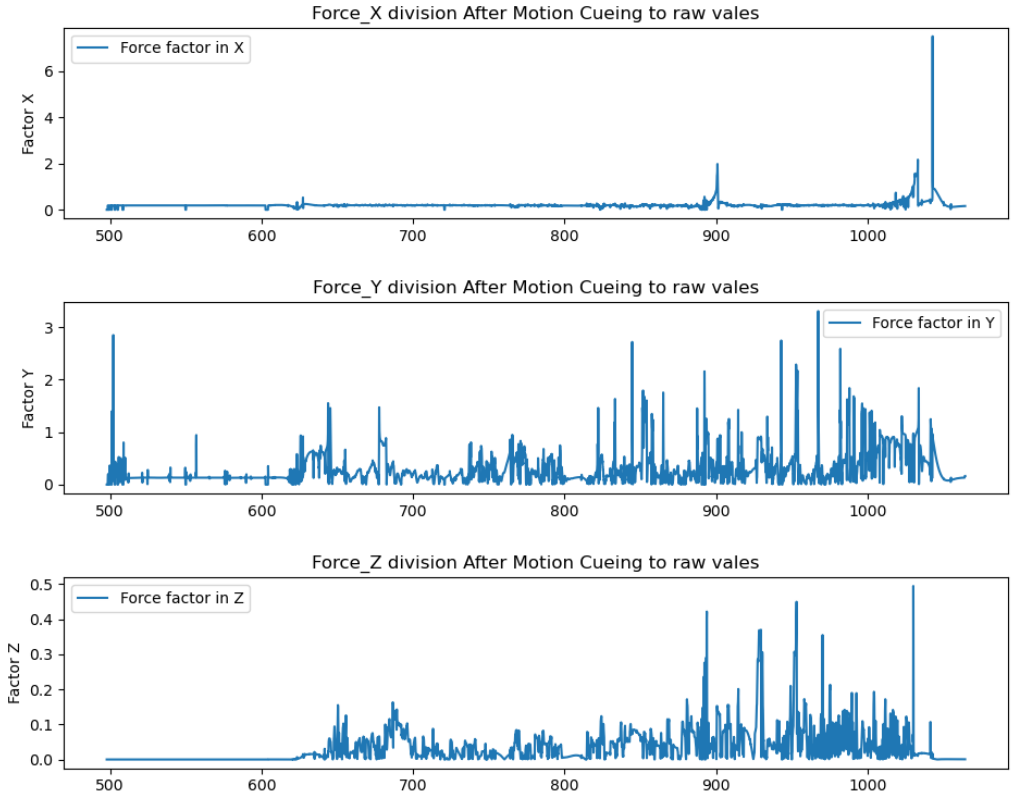

Figure 5.17: Force Factors with Offset

5.2.2 Evaluation of Cost Function Optimisation Results

Cost function optimisation was implemented to minimize the mean force factor across the X, Y, and Z axes using the L-BFGS-B method. The obtained average force factors with manual tuning, average force factors after the optimisation and washout damping factor after optimisation are mentioned in Table 6.

Table 6: Force factors and optimised washout damping factors

Axis	Average force factors (manual tuning)	Average force factors (post optimisation)	Optimised washout damping factor
X	0.2101	0.2271	0.00470148
Y	0.3002	0.2971	0.00470148
Z	0.0345	0.0505	0.00470153

- **Trade-Offs in Multi-Axis Optimisation**

From the results, we observe that the Y-axis force factor decreased slightly after optimisation. However, this improvement came at the cost of increased force factors in the X and Z axes. While the changes in X and Z may appear counterproductive, this behaviour is a result of the interdependency between axes because of platform dynamics and force distribution model.

As the platform is cable-driven, every change in translational position along one axis inevitably affects the tensions across all cables. During the optimisation, when the algorithm attempted to reduce damping in Y to improve cue replication, the resulting shift in Y-position also influenced X and Z forces. The motion platform adjusted its X and Z components slightly to maintain an overall force-feasible configuration, which caused marginal deviations in those force factors.

This outcome illustrates a typical case of multi-objective optimisation trade-offs, where enhancing performance in one domain may slightly compromise others due to shared physical constraints and coupled system behaviour [62]. In this case, the optimisation prioritised reducing Y-axis error, likely because it initially exhibited the largest force factor error and provided the most room for improvement within the feasible operational limits of the platform.

- **Influence of Manual Tuning**

The manually tuned parameters had already been refined that involved iteratively adjusting parameters and evaluating force feasibility on specific poses. As a result, the manually tuned settings already represented a reasonably good compromise between motion cue quality and force feasibility. Thus, there was limited room for dramatic improvement using optimisation.

- **Stability of Force Distribution Post-Optimisation**

A prominent visual outcome of the optimisation process was the noticeable improvement in the smoothness of force distribution across all eight cables. Figure 5.18 illustrates the cable forces before optimisation, while Figure 5.19 presents the results after optimisation. Although all force values remained within the acceptable operating range of the IPAnema 3 system (100 N to 3000 N), the post-optimisation plots exhibited fewer sharp spikes and fewer abrupt changes, therefore less stress on the robot and its mechanical parts. The transitions between force values appeared more gradual and consistent. This indicates that, despite only modest improvements in the numerical force factor metrics, the optimisation led to enhanced overall quality and stability in the platform's physical actuation. This suggests that, despite an only small change in the force factor metrics, the overall quality and stability of the force distribution benefited from the optimisation process.

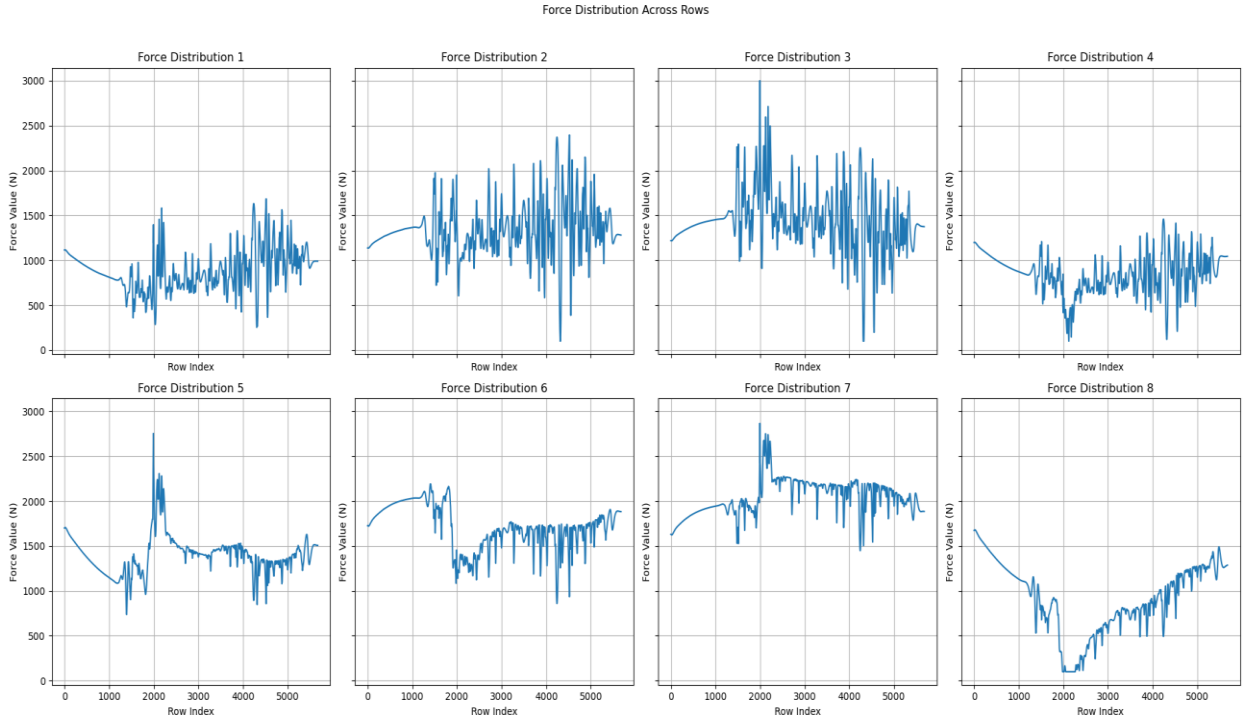

Figure 5.18: Force distribution before optimisation

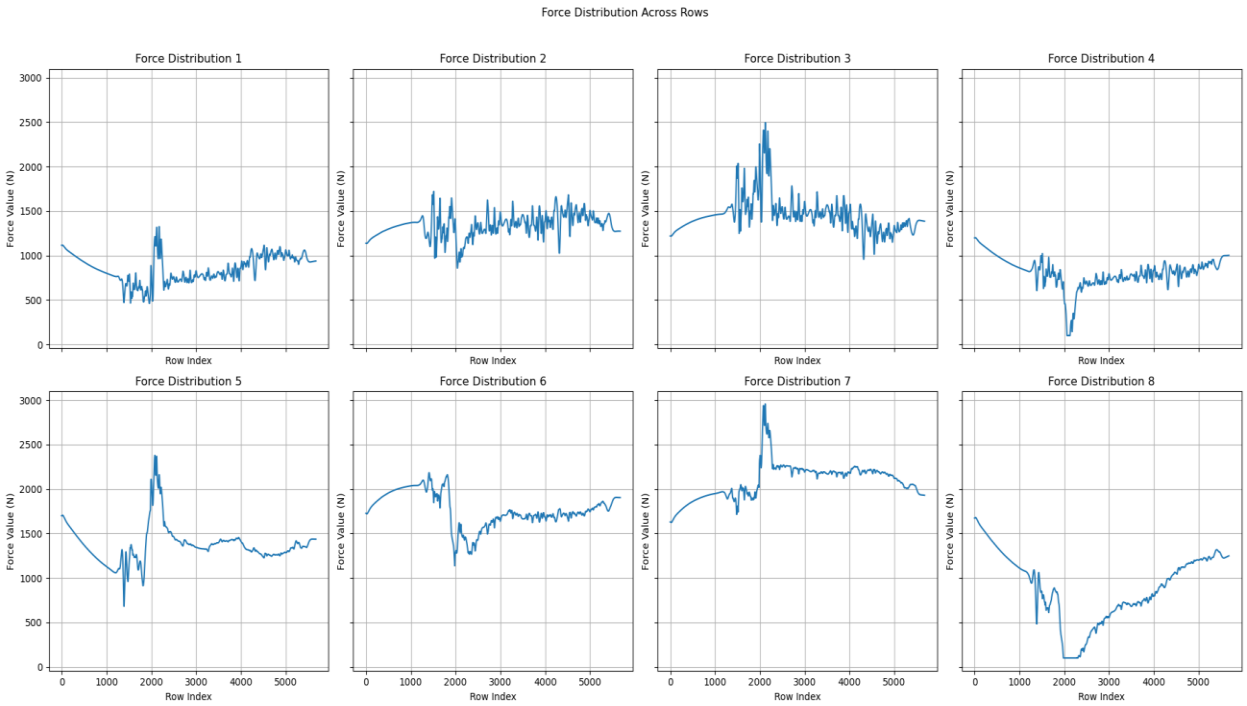

Figure 5.19: Force distribution after optimisation

5.3 Testing on Different Trajectory

Further testing was conducted on another trajectory, 'Trajectory 2' with demanding motion profiles involving sharp turns, rapid climbs and descents, and segments with high rotational rates. The scaling factors and cut-off frequencies were kept the same as before. The final parameter values for this trajectory are given in Table 7.

Table 7: Final parameter values for Trajectory 2

Parameters	Values
Manual washout damping factor [X, Y, Z]	[0.001, 0.005, 0.005]
Manual washout damping factor [roll, pitch, yaw]	[0.005, 0.007, 0.018]
Manually tuned average force factor [X, Y, Z]	[0.2176, 0.5563, 0.2850]
Optimised average force factor [X, Y, Z]	[0.2351, 0.5527, 0.2826]
Optimised washout damping factor [X, Y, Z]	[0.00482051, 0.00482051, 0.00482056]

In this more dynamic trajectory, the platform experienced greater deviations from its neutral position especially along the Z axis and in yaw. As a result, the washout damping values for Z and yaw increased compared to the earlier trajectory.

The force factors in the Y and Z directions were significantly higher during the dynamic trajectory due to a combination of factors. Firstly, in highly dynamic manoeuvres, the platform must respond to larger and more abrupt accelerations, especially in Y and Z axes. In Y, lateral forces during turns can be harder to reproduce accurately when the platform's range of motion is limited. In Z, the large vertical accelerations during manoeuvres (such as climbs or drops) lead to higher discrepancies between the simulated and generated forces, especially when the washout is pulling the platform back to neutral. Together, these factors contribute to increased force factor error in Y and Z during high dynamic conditions of the flight trajectory.

5.4 Testing on a High Payload

The next test was conducted using the same smooth trajectory as the first test but with a significantly higher payload. A wrench value of -2400 N was applied in the Z direction, corresponding to a payload of approximately 244.6 kg, close to the IPAnema 3 platform's maximum load capacity of 250 kg. The final parameter values for high payload are given in Table 8.

Table 8: Final parameter values for a high payload value

Parameters	Values
Manual washout damping factor [X, Y, Z]	[0.002, 0.006, 0.002]
Manual washout damping factor [roll, pitch, yaw]	[0.006, 0.008, 0.02]
Manually tuned average force factor [X, Y, Z]	[0.2300, 0.4010, 0.0410]
Optimised average force factor [X, Y, Z]	[0.2361, 0.3514, 0.0459]
Optimised washout damping factor [X, Y, Z]	[0.00307043, 0.00307043, 0.00307049]

In this case, the manually set washout damping values for both translational and rotational channels were initially higher than those used in the lower payload scenario (force Z = -1000N) to get force feasible solution. This conservative damping reduces the effective workspace of the platform.

The force factor in the Y direction decreased more notably after optimisation compared to the manual tuning. This suggests that the manually chosen Y-axis damping was suboptimal for the high payload case. Meanwhile, the X and Z damping values slightly increased post the optimisation, likely as a compensatory adjustment to maintain force feasibility which consistent with the earlier observation from the lower-payload scenario.

6 CONCLUSION

This final chapter provides a brief summary of the work accomplished in this thesis and outlines potential directions for future research. The main objectives, methodologies, and results are recapped. Following this, suggestions for future work are discussed based on the insights and limitations identified during the research.

6.1 Summary

This thesis focused on the development, implementation, and evaluation of a CWA tailored for a CDPR platform, specifically the IPAnema 3. The objective was to replicate flight simulation motion cues effectively, while ensuring physical feasibility within the mechanical limits of the platform through cable force distribution constraints

The work began with a comprehensive literature review on motion cueing strategies and current state-of-the-art techniques. This helped identify a research gap in applying washout algorithms specifically within the context of CDPR platforms, thereby establishing the foundation and motivation for this thesis.

The CWA was successfully implemented with cascaded filtering for both translational and rotational inputs, along with appropriate scaling and washout damping. At first, the damping parameters were manually tuned using an iterative approach, where each pose was checked for force feasibility using force distribution results. To improve this process, a cost function was introduced that measured the difference between the input and post motion cueing forces. A modified RAE method was used with a stabilising offset to mitigate the influence of small denominator values and outliers.

A cost-function-based optimisation procedure was then implemented using the L-BFGS-B method which is used to minimise the cost function. Only the washout damping parameters in the translational channel were optimised, while all other parameters were pre-defined. This is because the damping factors directly influence the system's dynamic response and the decay of motion over time, making them more critical for refining the washout behaviour compared to scaling or cut-off frequencies. Limiting optimisation to translational damping helped reduce computational complexity and allowed better control over force feasibility during early-stage development and validation. Crucially, this optimisation was constrained by force feasibility checks where only the damping values that resulted in valid cable force distributions across all poses were considered.

While some force error factors showed moderate improvements, in certain cases, the error changed slightly due to the interdependent dynamics of the IPAnema 3 platform. Nevertheless, the post-optimisation force distribution plots exhibited smoother cable force profiles with fewer abrupt spikes, indicating improved physical stability. The optimisation approach was further tested on

scenarios such as a highly dynamic flight trajectory and a heavy payload condition, demonstrating the robustness and adaptability of the method (see A1 for comparison).

Overall, combining cost-based optimisation with force feasibility checks provided a systematic and effective method to tune the CWA for the IPAnema 3 platform.

6.2 Future Work

This section outlines possible directions for extending the work presented in this thesis. While the current implementation of CWA has shown good results, there are several ways it can be further improved and adapted. These suggestions aim to enhance the system's performance, make it more flexible, and prepare it for real world testing on the IPAnema 3 platform and similar cable-driven systems. The work lays a solid foundation for the motion simulation at Fraunhofer IPA. The future work could involve:

- **Real-World Implementation on Hardware**
  One of the most immediate future steps would be to implement the developed algorithm on the actual IPAnema 3 hardware. This would validate the optimisation framework under real-world mechanical conditions.

- **Extending Optimisation to Rotational Parameters**
  In this thesis, optimisation was limited to washout damping values in translational channel. Future work could expand this to include rotational channels, which may lead to better overall cue reproduction and improve dynamic behaviour during aggressive flight manoeuvres. Additionally, incorporating checks for cable–cable collisions under high rotational movements could be beneficial. Since rotational angles can be higher near the centre of the workspace compared to the edges, adapting orientation constraints or implementing workspace-aware collision detection could prevent unsafe configurations.

- **Axis-Wise Independent Optimisation**
  To avoid the trade-offs observed when optimising all translational damping parameters together, future efforts could explore optimisation strategies that tune X, Y, and Z parameters independently while keeping the force feasibility. This could help reduce unintended inter-axis compensation and improve performance in specific directions.

- **Adaptive Platform Orientation and Origin Shifting**
  The current setup uses a fixed platform orientation and origin point. Future work could explore adjusting the platform's orientation. For instance, switching the X and Y axes or swapping the roll and pitch directions based on the dominant translation or rotation in the trajectory. This would help make better use of the available workspace. Additionally,

dynamically shifting the platform's origin based on the trajectory could further maximise workspace efficiency.

- **Online Optimisation and Real-Time Cueing**
  This work focuses on pre-recorded trajectories. A potential extension is to adapt the framework for real-time trajectory input, allowing on-the-fly optimisation or motion cueing. This would be particularly valuable for pilot-in-the-loop simulations or unpredictable flight conditions.

- **Integration of Pilot Feedback and Visual Cues**
  Human perception plays a crucial role in the effectiveness of motion simulators. Future studies could incorporate pilot feedback, physiological responses, and the integration of visual cues—whether through VR headsets or screen-based environments—into the cost function. This would enable the system to optimise not just for physical feasibility, but also for perceived realism and motion comfort, making the simulation more immersive and effective.

- **Generalisation to Other CDPR Systems**
  While the work is built around IPAnema 3, the methodology could be adapted to other CDPR platforms. Exploring its scalability and effectiveness on different CDPR designs and applications could broaden its impact.

A APPENDIX

A1 Data Logging using XML file.

The code snippet of the XML file for logging the data for the translational accelerations generated by FlightGear is given in the Figure A.1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PropertyList>
    <logging>
        <log>
            <filename>C:\Users\mrf-ar\AppData\Roaming\flightgear.org\Export\log_parameters_mrf_non_crash.csv</filename>
            <enabled>true</enabled>
            <interval-ms>100</interval-ms>
            <delimiter>,</delimiter>
            <entry>
                <enabled>true</enabled>
                <title>Acceleration_X</title>
                <property>/accelerations/pilot/x-accel-fps_sec</property>
            </entry>
            <entry>
                <enabled>true</enabled>
                <title>Acceleration_Y</title>
                <property>/accelerations/pilot/y-accel-fps_sec</property>
            </entry>
            <entry>
                <enabled>true</enabled>
                <title>Acceleration_Z</title>
                <property>/accelerations/pilot/z-accel-fps_sec</property>
            </entry>
```

Figure A.1: XML file code snippet for logging translational accelerations

A2 CWA for X acceleration and Pitch rate

The code snippet for CWA in python for the inputs - X axis acceleration and pitch rate is seen in the Figure A.2.

```python
# --- X Axis Washout with Early Damping ---
# Scale input acceleration for high-pass filtering
scaled_ax_hp = accel_data['x'] * self.scale_x_hp
# Apply 1st and 2nd order high-pass Butterworth filters
highpass_accel_x = butter_filter(scaled_ax_hp, self.hp_acc_x_1, self.fs, order=1, filter_type='high')
highpass_accel_x_2 = butter_filter(highpass_accel_x, self.hp_acc_x_2, self.fs, order=2,
filter_type='high')
# Apply washout damping to the filtered acceleration
damped_accel_x = highpass_accel_x_2 * np.exp(-self.washout_damping_x *
np.arange(len(highpass_accel_x_2)) / self.fs)
# Integrate to get velocity and then position
vel_x = it.cumulative_trapezoid(damped_accel_x, dx=1 / self.fs, initial=0)
pos_x = it.cumulative_trapezoid(vel_x, dx=1 / self.fs, initial=0)

# --- Pitch Angle Calculation (Rotation + Tilt Coordination) ---
# Scale pitch rate and apply high-pass filters
scaled_pitch = rotational_data['pitch'] * self.scale_b_hp
highpass_pitch = butter_filter(scaled_pitch, self.hp_pitch_1, self.fs, order=1, filter_type='high')
highpass_pitch_2 = butter_filter(highpass_pitch, self.hp_pitch_2, self.fs, order=2, filter_type='high')
# Integrate to get angle and apply washout damping
washout_pitch = it.cumulative_trapezoid(highpass_pitch_2, dx=1 / self.fs, initial=0)
pitch_angle = washout_pitch * np.exp(-self.washout_damping_b * np.arange(len(washout_pitch)) / self.fs)
# Tilt coordination from X-axis acceleration
scaled_ax_lp = accel_data['x'] * self.scale_x_lp
tilt_x = scaled_ax_lp / -Acc_g
lowpass_accel_x = butter_filter(tilt_x, self.lp_acc_x, self.fs, self.order, 'low')
tilt_x_angle = np.arcsin(np.clip(lowpass_accel_x, -1, 1))
tilt_x_angle_deg = np.degrees(tilt_x_angle)
# Combine both components
total_pitch = pitch_angle + tilt_x_angle_deg
```

Figure A.2: Python code for CWA

A3 Validation of Force Distribution

The python code snippet for validating the force distribution across the cables for a given pose is seen in the Figure A.3.

```
input_file = 'C:\\Users\\mrf-ar\\motioncueing\\processed_motion_data.csv'  # CSV file containing motion
cues
output_file = 'C:\\Users\\mrf-ar\\motioncueing\\Force distribution\\force_distribution.csv'  # Output
file to store force values
# Set up geometry and workspace for IPAnema 3
setGeometry(Irobot, Ikin, Iws)
calcWorkspaceforce(Iws)

with open(input_file, mode='r') as infile, open(output_file, mode='w', newline='') as outfile:
    reader = csv.DictReader(infile)
    writer = csv.writer(outfile)
    for row in reader:
        # Extract position and orientation from each row
        x = float(row['x_position'])
        y = float(row['y_position'])
        z = float(row['z_position'])
        roll = float(row['roll_angle'])
        pitch = float(row['pitch_angle'])
        yaw = float(row['yaw_angle'])
        # Calculate cable force distribution for the current pose
        force_distribution = ForceDis(Irobot, x, y, z, roll, pitch, yaw)
        # Check if the pose is feasible by verifying if ForceDis returns valid (non-zero) forces
        if force_distribution:
            writer.writerow(force_distribution)  # Save valid force values to output file
        else:
            writer.writerow([0] * 8)  # Save all-zero row if the pose is infeasible
            print(f"The pose x={x}, y={y}, z={z}, roll={roll}, pitch={pitch}, yaw={yaw} is not
feasible.")  # Log infeasible pose
```

Figure A.3: Python code for force distribution validation

A4 Cost Function Calculation

The python code snippet for calculating the cost function in X which is the average force factor error is seen in the Figure A.4.

```
# --- Force Calculation Before and After Motion Cueing (X-axis) ---

# Raw force in X before motion cueing
force_x = accel_data['x'] * Pilot_mass

# Velocity and acceleration after motion cueing
v_x = np.gradient(processed_movement['x'], data['Time'])
accel_x_post = np.gradient(v_x, data['Time'])

# Final acceleration after adding tilt coordination
final_accel_x = accel_x_post + (-1 * processed_movement['lowpass_accel_x'] * Acc_g)

# Force after motion cueing
force_x_post = final_accel_x * Pilot_mass

# --- Force Error Factor (X-axis) ---

# Relative error between raw and post-cueing force
force_x_factor = np.abs(force_x_post - force_x) / (np.abs(force_x) + 5)

# Clean up NaNs or infinities
force_x_factor = np.where(np.isfinite(force_x_factor), force_x_factor, 0)
force_x_factor = np.nan_to_num(force_x_factor)

# Compute average force factor (used as cost metric)
average_force_x_factor = np.mean(force_x_factor)
print(f"Average Force_X_Factor: {average_force_x_factor}")
```

Figure A.4: Average force factor calculation for X axis in python

A5 Cost Function Optimisation

An example logic for the part of the python code showing how the optimisation is taking place with the use of L-BFGS-B method for minimisation of cost function is shown in Figure A.5.

```python
from scipy.optimize import minimize

def cost_fn(params):
    wx, wy, wz = params

    # Run Classical Washout with updated damping parameters
    washout = ClassicalWashout(
        fs, hp_acc_x_1, hp_acc_y_1, hp_acc_z_1, hp_acc_x_2, hp_acc_y_2, hp_acc_z_2,
        lp_acc_x, lp_acc_y, lp_acc_z, hp_roll_1, hp_pitch_1, hp_yaw_1,
        hp_roll_2, hp_pitch_2, hp_yaw_2, scale_x_hp, scale_y_hp, scale_z_hp,
        scale_x_lp, scale_y_lp, scale_a_hp, scale_b_hp, scale_c_hp,
        wx, wy, wz, washout_damping_a, washout_damping_b, washout_damping_c
    )
    result = washout.process(accel_data, rotational_data)
    poses = extract_positions(result)

    # Penalize if force distribution is not feasible
    if not is_force_feasible(Irobot, poses):
        return 1e6

    # Calculate post-motion cueing forces
    v_x = np.gradient(result["x"], data["Time"])
    accel_x_post = np.gradient(v_x, data["Time"])
    final_accel_x = accel_x_post + (-1 * result["lowpass_accel_x"] * Acc_g)
    force_x_post = final_accel_x * Pilot_mass
    force_x_ref = accel_data["x"] * Pilot_mass

    # Cost = Normalized absolute error between desired and simulated forces
    fx = np.abs(force_x_post - force_x_ref) / (np.abs(force_x_ref) + 5)
    return np.nanmean(fx)  # (Repeat similarly for y and z if needed)

# Initial guess and bounds for damping parameters
initial = [0.01, 0.01, 0.01]
bounds = [(1e-5, 0.01), (1e-5, 0.01), (1e-5, 0.01)]

# Run L-BFGS-B minimization
result = minimize(cost_fn, initial, bounds=bounds, method="L-BFGS-B")

# Optimized washout damping values
print("Optimized washout damping (x, y, z):", result.x)
```

Figure A.5: Logic for Cost function optimisation in python

A6 Comparison of Final Parameters

Table A-1 presents the comparison of the parameters for different flight simulation scenarios.

Table A-1: Comparison of final parameters

Trajectory type	**Manual damping [X, Y, Z]**	**Optimised damping [X, Y, Z]**	**Manual damping [X, Y, Z]**	**Manual average force factor [X, Y, Z]**	**Optimised average force factor [X, Y, Z]**
Trajectory 1 with Z = -1000N	[0.001, 0.005, 0.001]	[0.00470148, 0.00470148, 0.00470153]	[0.005, 0.007, 0.01]	[0.2101, 0.3002, 0.0345]	[0.2271, 0.2971, 0.0505]
Trajectory 2 with Z = -1000N	[0.001, 0.005, 0.001]	[0.00482051, 0.00482051, 0.00482056]	[0.005, 0.007, 0.018]	[0.2176, 0.5563, 0.2850]	[0.2351, 0.5527, 0.2826]
Trajectory 1 With Z = -2400N	[0.002, 0.006, 0.002]	[0.00307043, 0.00307043, 0.00307049]	[0.006, 0.008, 0.02]	[0.2300, 0.4010, 0.0410]	[0.2361, 0.3514, 0.0459]

BIBLIOGRAPHY

[1] Fraunhofer IPA, Institute Profile. [Online]. Available: https://www.ipa.fraunhofer.de/en/about-us/institute-profile.html, 27.02.2025

[2] Robot and Assistive Systems at Fraunhofer IPA, Stuttgart. [Online]. Available: https://dih-hero.eu/robot-and-assistive-systems-at-fraunhofer-ipa/, 27.02.2025

[3] A. H. J. Jamson, *"Motion Cueing in Driving Simulators for Research Applications"*, Ph.D. dissertation, Univ. of Leeds, 2010. [Online]. Available: https://core.ac.uk/download/pdf/1145908.pdf

[4] B. D. C. Augusto and R. J. L. Loureiro, *"Motion Cueing in the Chalmers Driving Simulator: A Model Predictive Control Approach"*, M.Sc. thesis, Chalmers Univ. of Technol., Göteborg, Sweden, 2009. [Online]. Available: https://odr.chalmers.se/bitstream/20.500.12380/98871/1/98871.pdf

[5] J. Freeman, G. Watson, Y. Papelis, T. Lin, A. Tayyab, R. Romano, and J. Kuhl, "The Iowa Driving Simulator: An Implementation and Application Overview", *SAE Technical Paper 950174*, 1995, DOI: 10.4271/950174.

[6] H. J. Teufel, H.-G. Nusseck, K. A. Beykirch, J. S. Butler, M. Kerger, and H. H. Bülthoff, "MPI Motion Simulator: Development and Analysis of a Novel Motion Simulator", *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Hilton Head, South Carolina, August 2007, DOI: 10.2514/6.2007-6476.

[7] T. Bellmann, J. Heindl, M. Hellerer, R. Kuchar, K. Sharma, and G. Hirzinger, "The DLR Robot Motion Simulator Part I: Design and Setup," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4694–4701. DOI: 10.1109/ICRA.2011.5979913.

[8] F. Nieuwenhuizen and H. H. Bülthoff, "The MPI CyberMotion Simulator: A Novel Research Platform to Investigate Human Control Behavior," *Journal of Computing Science and Engineering*, vol. 7, no. 2, pp. 122–131, 2013. DOI: 10.5626/JCSE.2013.7.2.122.

[9] Pandilov, Z., & Dukovski, V. (2014). Comparison of the Characteristics Between Serial and Parallel Robots. *Acta Technica Corviniensis – Bulletin of Engineering*, 7(1), 143–150. [Online]. Available: *https://acta.fih.upt.ro/pdf/2014-1/ACTA-2014-1-19.pdf*

[10] P. Miermeister, M. Lächele, R. Boss, C. Masone, M. F. Huber, A. Schlaefer, A. Pott, and H. H. Bülthoff, "The CableRobot Simulator: Large Scale Motion Platform Based on Cable Robot Technology," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, 2016, pp. 3024–3029. DOI: 10.1109/IROS.2016.7759468.

[11] Tang, X. (2014). "An Overview of the Development for Cable-Driven Parallel Manipulator". *Advances in Mechanical Engineering*, 6, Article ID 823028. DOI: 10.1155/2014/823028.

[12] Qian, S., Zi, B., Shang, W.-W., & Xu, Q.-S. (2018). "A Review on Cable-driven Parallel Robots". *Chinese Journal of Mechanical Engineering*, 31, Article 66. DOI: 10.1186/s10033-018-0267-9

[13] Kuhl, J. G., Evans, D., Papelis, Y., Romano, R., & Watson, G. (1995). "The Iowa Driving Simulator: An Immersive Research Environment". *IEEE Computer*, 28(7), 35–41. DOI: 10.1109/2.391039.

[14] The National Advanced Driving Simulator. [Online]. Available: https://dsri.uiowa.edu/nads-1#&gid=1&pid=1, 27.02.2025

[15] Fraunhofer IPA, Cable-Driven Parallel Robots. [Online]. Available: https://www.ipa.fraunhofer.de/en/expertise/robot-and-assistive-systems/intralogistics-and-material-flow/cable-driven-parallel-robot.html, 27.02.2025

[16] Brandt, T., Dichgans, J., & Koenig, E. (1973). „Differential effects of central versus peripheral vision on egocentric and exocentric motion perception". *Experimental Brain Research*, 16(5), 476–491. DOI: 10.1007/BF00234474

[17] Young, L. R., Dichgans, J., Murphy, R., & Brandt, T. (1973). "Interaction of optokinetic and vestibular stimuli in motion perception". *Acta Oto-Laryngologica*, 76(sup315), 24–31 DOI: 10.3109/00016487309121479

[18] Vestibular Disorder Association, Ear anatomy. [Online]. Available: https://vestibular.org/article/what-is-vestibular/the-human-balance-system/ear-anatomy/, 03.03.2025

[19] Dreamstine, Human Vestibular System, [Online]. Available: https://www.dreamstime.com/human-vestibular-system-highlighting-its-structure-components-educational-purposes-diagram-hand-drawn-schematic-raster-image319897280, 03.03.2025

[20] Fischer, M. (2007). "A survey of state-of-the-art motion platform technology and motion cueing algorithms". In Proceedings of the 2nd Motion Simulator Conference, Braunschweig, Germany. [Online]. Available: https://www.researchgate.net/publication/224987062_A_Survey_of_State-of-the-art_Motion_Platform_Technology_and_Motion_Cueing_Algorithms

[21] Casas, S., Olanda, R., & Dey, N. (2017). "Motion cueing algorithms: A review. Algorithms, evaluation and tuning". *International Journal of Virtual and Augmented Reality*, 1(1), 90–106. DOI: 10.4018/IJVAR.2017010107

[22] Reymond, G., & Kemeny, A. (2000). "Motion cueing in the Renault driving simulator. *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility*, 34(4), 249–259". DOI: 10.1076/vesd.34.4.249.2059

[23] Reid, L. D., & Nahon, M. A. (1985*). Flight Simulation Motion-Base Drive Algorithms: Part 1 - Developing and Testing the Equations.* University of Toronto. [Online]. Available: https://repository.tudelft.nl/record/uuid:45b071c0-0568-4e8f-948f-dfa52d350665

[24] Colombet, F., Dagdelen, M., Reymond, G., Père, C., Merienne, F., & Kemeny, A. (2008). "Motion cueing: What's the impact on the driver's behaviour?", *In Proceedings of the Driving Simulation Conference* (pp. 1–8). Monaco. [Online]. Available: https://www.researchgate.net/publication/237305826_Motion_Cueing_what's_the_impact_on_the_driver's_behaviour

[25] Schmidt, S. F., & Conrad, B. (1969). „The Calculation of Motion Drive Signals for Piloted Flight Simulators. Palo Alto, CA, USA: NASA". [Online]. Available: https://ntrs.nasa.gov/citations/19700001603

[26] Schmidt, S. F., & Conrad, B. (1970). „Motion Drive Signals for Piloted Flight Simulators". Palo Alto, CA, USA: NASA. [Online]. Available: https://ntrs.nasa.gov/citations/19700017803

[27] Reid, L. D., & Nahon, M. A. (1986). „Flight Simulation Motion-Base Drive Algorithms: Part 2 - Selecting the System Parameters". University of Toronto. [Online]. Available: https://repository.tudelft.nl/record/uuid:4faf3129-88c9-4117-82e9-f9819601dafd

[28] de Winter, R., van Stein, N., Dijkman, M., & Bäck, T. (2019). „Designing Ships Using Constrained Multi-Objective Efficient Global Optimization". *Lecture Notes in Computer Science, 11353,* 230–245. Springer. DOI: 1007/978-3-030-13709-0_16

[29] Micomonaco, M. (2019). Large angle washout algorithms for flight simulators (Master's thesis, Carleton University). [Online]. Available: https://carleton.ca/johnhayes/wp-content/uploads/MikaylaMicomonacoThesisCompressedOptJH.pdf

[30] Groen, E. L., & Bles, W. (2004). How to use body tilt for the simulation of linear self-motion. *Journal of Vestibular Research*, 14(5), 375–385. DOI: 10.3233/VES-2004-14503

[31] M. Wentink, E. L. Groen, and P. Feenstra, "New technologies & applications in the Desdemona simulator," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit*, San Francisco, CA, USA, 2005. [Online]. Available: https://publications.tno.nl/publication/34610719/olTtmW/wentink-2008-new.pdf.

[32] D. Ariel and R. Sivan, "False cue reduction in moving flight simulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, no. 4, pp. 665-671, 1984. DOI: 10.1109/TSMC.1984.6313342.

[33] Parrish, R. V., Dieudonne, J. E., & Martin, D. J. Jr. (1975), "Coordinated Adaptive Washout for Motion Simulators". *Journal of Aircraft*, 12(1), 44–50. DOI:10.2514/3.59800

[34] M. A. Nahon, L. D. Reid, and J. Kirdeikis, "Adaptive Simulator Motion Software with Supervisory Control," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 2, pp. 376-383, 1992. DOI: 10.2514/3.20846

[35] R. Sivan, J. Ish-Shalom, and J.-K. Huang, "An Optimal Control Approach to the Design of Moving Flight Simulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 12, no. 6, pp. 818–827, 1982. DOI: 10.1109/TSMC.1982.4308915

[36] M. Kurosaki, "Optimal Washout for Control of a Moving Base Simulator," presented at the *7th IFAC World Congress*, Helsinki, Finland, June 12–16, 1978. DOI: 10.1016/S1474-6670(17)66089-0

[37] Parallel robot IRB 360 FlexPicker® series: [Online]. Available: https://www.directindustry.com/prod/abb-robotics/product-30265-169123.html, 05.03.2025

[38] T. Paty, P. Cardou, and S. Krut, "Cable-Driven Parallel Robot Modelling Considering Pulley Kinematics and Cable Elasticity," *Mechanism and Machine Theory*, vol. 158, p. 104211, 2021. DOI: 10.1016/j.mechmachtheory.2020.104211

[39] Martins, J. R. R. A., & Ning, A. (2021). "Engineering Design Optimization. Cambridge University Press". [Online]. Available: https://flowlab.groups.et.byu.net/mdobook.pdf

[40] Pandey, A. (2020). "Defining Cost Function for Optimization of Machine Learning Models". *International Journal of Research in Engineering, Science and Management,* 3(2), 186–189. [Online]. Available: https://www.ijresm.com/Vol.3_2020/Vol3_Iss2_February20/IJRESM_V3_I2_47.pdf

[41] Analytics Vidhya, "Cost Function is No Rocket Science!". [Online]. Available: https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/#h-why-to-use-a-cost-function

[42] Metaschool, "What is a Cost Function in Machine Learning?". [Online]. Available: https://metaschool.so/articles/cost-function#What_is_a_Cost_Function_in_Machine_Learning

[43] Datacamp, "Normalization in Machine Learning". [Online]. Available: https://www.datacamp.com/tutorial/normalization-in-machine-learning

[44] Witkin, A., & Baraff, D. (1997). Physically Based Modeling: Principles and Practice. SIGGRAPH '97 Course Notes. [Online]. Available: https://www.cs.cmu.edu/~baraff/sigcourse/

[45] Conrad, B., Schmidt, S. F., & Douvillier, J. G. (1973). "Washout Circuit Design for Multi-Degrees-of-Freedom Moving Base Simulators". *Proceedings of the American Institute of Aeronautics and Astronautics Visual and Motion Simulation Conference.* DOI: 10.2514/6.1973-929

[46] Reid, L. D., & Nahon, M. A. (1988). Response of airline pilots to variations in flight simulator motion algorithms. *Journal of Aircraft*, 25(7), 639–646. DOI: 10.2514/3.45635

[47] Nahon, M. A., & Reid, L. D. (1990). Simulator motion-drive algorithms: A designer's perspective. *Journal of Guidance, Control, and Dynamics*, 13(2), 356–362. DOI: 10.2514/3.20557

[48] Grant, P.R., & Reid, L.D. (1997). “Motion washout filter tuning: Rules and requirements”. *Journal of Aircraft*, 34(2), 145–151. DOI: 10.2514/2.2158

[49] Advani, S., Hosman, R., & Haeck, N. (2002). “Integrated design of a motion cueing algorithm and motion-base mechanism for a Wright Flyer simulator”. *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit.* DOI: 10.2514/6.2002-4690

[50] Hosman, R., Advani, S., & Haeck, N. (2002). “Integrated Design of Flight Simulator Motion Cueing Systems”. *Paper presented at the Royal Aeronautical Society Conference on Flight Simulation*, London, UK. DOI: 10.1017/S000192400000049X

[51] Katliar, M., Fischer, J., Frison, G., Diehl, M., & Bülthoff, H. H. (2017). Nonlinear model predictive control of a cable-robot-based motion simulator. *IFAC-PapersOnLine, 50*(1), 9833-9839. DOI: 10.1016/j.ifacol.2017.08.901

[52] Cyberneum, CableRobot Simulator. [Online]. Available: https://www.cyberneum.de/CableRobotSimulator, 11.03.2025

[53] Kraus, W., Spiller, A., & Pott, A. (2016). “Energy efficiency of cable-driven parallel robots”. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1568–1573). IEEE. DOI: 10.1109/ICRA.2016.7487087

[54] Fraunhofer IPA, Product Sheet “Cable-driven Parallel Robots for Handling Goods of all Sizes”. [Online]. Available: https://www.ipa.fraunhofer.de/en/expertise/robot-and-assistive-systems/intralogistics-and-material-flow/cable-driven-parallel-robot.html, 12.03.2025

[55] Liao, C. S., Huang, C. F., & Chieng, W. H. (2004). “A novel washout filter design for a six degree-of-freedom motion simulator”. *JSME International Journal Series C, Mechanical Systems, Machine Elements and Manufacturing, 47*(2), 626–636. DOI: 10.1299/jsmec.47.626

[56] FlightGear. [Online]. Available: https://www.flightgear.org/, 12.03.2025

[57] Electrical 4 U, “Butterworth Filter: What is it? (Design & Applications)”. [Online]. Available. https://www.electrical4u.com/butterworth-filter/, 13.03.2025

[58] Pott, A. (2019). “WireX – An Open Source Initiative: Scientific Software for Analysis and Design of Cable-driven Parallel Robots”. *Presented at the Fourth International*

Conference on Cable-Driven Parallel Robots, Krakow, Poland. DOI: 10.13140/RG.2.2.25754.70088.

[59] Fraunhofer Gitlab, WireX Repository. [Online]. Available: https://gitlab.cc-asp.fraunhofer.de/wek/wirex.

[60] Martin, C., Fabritius, M., Stoll, J.T., & Pott, A. (2021). "Accuracy Improvement for CDPRs Based on Direct Cable Length Measurement Sensors. In M. Gouttefarde, T. Bruckmann, & A. Pott (Eds.)", *Cable-Driven Parallel Robots* (pp. 348–359). Springer. DOI: 10.1007/978-3-030-75789-2_28

[61] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995. DOI: 10.1137/0916069

[62] P. K. Shukla, C. Hirsch, and H. Schmeck, "Towards a Deeper Understanding of Trade-offs Using Multi-objective Evolutionary Algorithms," in *Applications of Evolutionary Computation. EvoApplications 2012*, C. Di Chio *et al.*, Eds., *Lecture Notes in Computer Science*, vol. 7248. Berlin, Heidelberg: Springer, 2012, pp. 396–405. DOI: 10.1007/978-3-642-29178-4_40