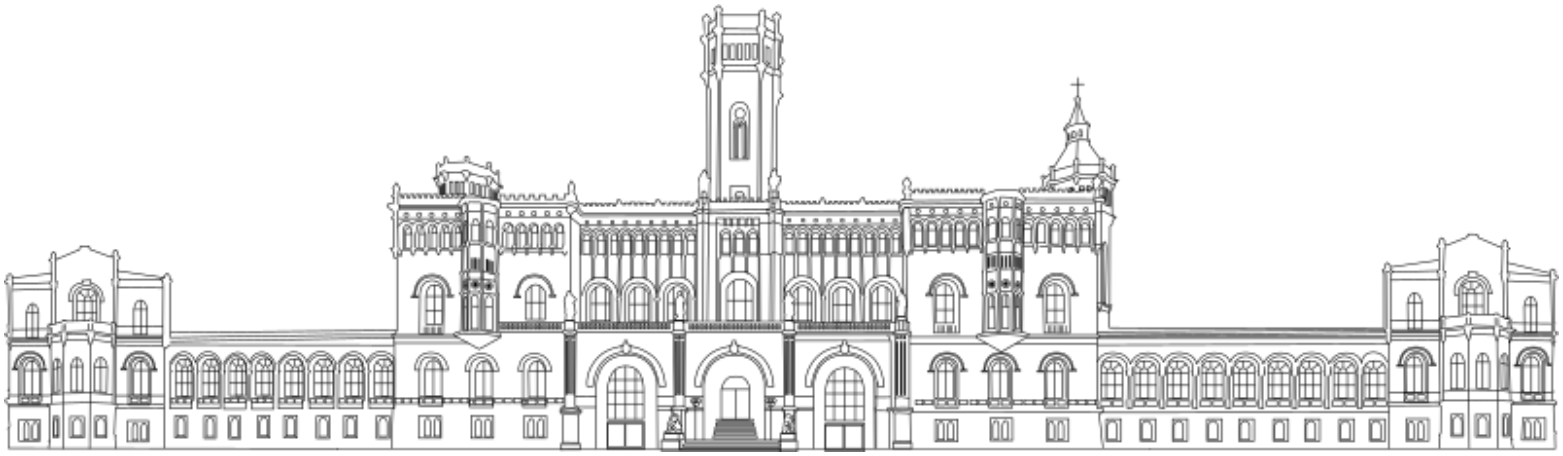
ikg

Institut für Mechatronische Systeme
imes Leibniz Universität Hannover

Leibniz Universität Hannover

Spatiotemporal Calibration between a Helmet Mapping System and the HoloLens Augmented Reality System



Master's Thesis June / 2021

Özgün Karataş
Matriculation Number 10025506

Hanover, 24. June 2021

First Examiner	Prof. Dr.-Ing. Marc Wielitzka
Second Examiner	Prof. Dr.-Ing. Monika Sester
Supervisor	Vinu Kamalasanan, M.Sc.

I, Özgün Karataş, hereby affirm that the present thesis was written independently, that no references and aids other than those indicated were used, that all passages of the thesis which were taken over literally or analogously from other sources are marked as such and that the thesis has not yet been presented to any examination board in the same or similar form.

Hannover, 24. June 2021

(Özgün Karataş)

Task Description of the Master's Thesis

Özgün Karataş, Matriculation Number 10025506

Introduction:

Mobile mapping systems are used to map indoor environments by utilizing LiDAR sensors. Such systems are held in the hand or worn on the body supported by a backpack for data acquisition. When a helmet-mounted LiDAR with IMUs (HMS) is associated with an AR device like the HoloLens, the sensor scan data can be visualized for understanding the mapping quality of the HMS in real time. To achieve the objective of visualizing 3D data acquired with the HMS on the HoloLens camera, precise synchronization is needed between the two systems.

Objective:

The objective of this thesis is to achieve time synchronization and rigid body transformation between a helmet-mounted mapping system (HMS) equipped with a Velodyne LiDAR sensor and an Xsens IMU module, and an Augmented Reality (AR) system. The AR system explored in this thesis encompasses the HoloLens from Microsoft. While the Xsens IMU module captures data synchronized to an external timeline, the HoloLens captures data in batches and is not synchronized to an external timeline. Therefore, a spatiotemporal calibration between these two sensors is aimed. Due to the non-linear and highly dynamic nature of human movement, the data acquisition will first be done on a rigid system to minimize disturbances and mismatches. After achieving success on the rigid system, pedestrian motion will be tested to find patterns for data synchronization regarding human movement.

For this thesis, the following tasks are to be fulfilled:

- literature research regarding spatiotemporal synchronization and SLAM using the HoloLens,
- implementation of a data acquisition infrastructure for the HMS and the HoloLens sensors considering requirements of real-time data processing using ROS,
- estimation of the frequency and error properties of the IMU modules for the HMS and the HoloLens,
- design of a rigid body system for data acquisition and benchmarking,
- implementation of an algorithm for the time synchronization based on the previous data acquisition,
- data acquisition with the HMS and the HoloLens in an indoor environment with pedestrian motion,
- creating patterns for data synchronization with pedestrian motion,
- time synchronized data acquisition with a tracking apparatus in an indoor environment,
- analysis of the results and conclusive remarks regarding synchronization of these heterogenous systems.

The time period for this thesis is 900 hours.
Handout date of the assignment: 19.11.2020
Deadline date of the thesis: 24.06.2021

Supervisor: Vinu Kamalasanan

Student: Özgün Karataş

Abstract

This thesis presents a stand-alone data acquisition and calibration framework between the Microsoft HoloLens and a Helmet Mapping system (HMS) housing an Xsens IMU, which aims to solve the spatiotemporal calibration by utilizing the angular velocity values to perform temporal calibration and by employing a complementary filter to perform extrinsic rotation calibration. Both the HoloLens and the HMS are worn by a subject at the same time throughout the duration of the data acquisition. Viable pedestrian motions are derived and motion patterns with insufficient excitation are discarded. A series of human and rigid body experiments are carried out with a calibration board to verify the accuracy and integrity of the system with a state-of-the-art calibration framework. It is shown that the pre-calculated distance between the sensors can be used to test whether the calibration framework is able to handle the so-called lever-arm effect which arises when acceleration values of multiple points on a rigid body are to be merged. The results obtained from experiments and comparisons are discussed regarding real-time capabilities and further development ideas for the framework. The thesis aims to create a unified framework to aid the HoloLens with an industrial grade IMU in augmented reality applications.

Keywords: spatiotemporal calibration, fusion of heterogeneous systems, augmented reality, multi-IMU calibration, visual-inertial odometry.

Contents

List of Figures

List of Tables

List of Abbreviations

AR	Augmented Reality
CAD	Computer Aided Design
CPU	Central Processing Unit
DCM	Direction Cosine Matrix
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
FPS	Frames Per Second
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HL	HoloLens
HMD	Helmet Mounted Display
HMS	Helmet Mapping System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
INS	Integrated navigation system
KF	Kalman Filter
MFI	Multisensor Fusion and Integration
MSCKF	Multi-state Constraint Kalman Filter
ROS	Robot Operating System
RTOS	Real Time Operating System
SLAM	Simultaenous Localization and Mapping
UKF	Unscented Kalman Filter
UWP	Universal Windows Platform
VR	Virtual Reality

1 INTRODUCTION

Augmented and Virtual Reality (AR and VR respectively) are an intertwined couple of emerging technology which have been popular for the last few decades [**?**]. The rise in computer power both in industrial scale and also in the consumer market has given the technology giants like Microsoft and Oculus the technical prowess to create immersive headsets. In the case of augmented reality, the immersion is created in holograms, which are projected from the headset onto a certain distance away from the wearer's eyes and these created holograms orient themselves according to the geometry of the surroundings. The real world is still intact and it is in a sense "extended" with the holograms [**?**].

On the other hand, virtual reality creates a heightened layer of immersion, where the wearer is separated entirely from their surroundings and all contact is lost with the real world. Here, movement of the head is directly transferred to the movement in the virtual reality, meaning that looking around would produce the same effects inside the VR as well. These two technologies have been widely used in the entertainment industry due to their allure and novelty, but using them in research areas has been increasing as well.

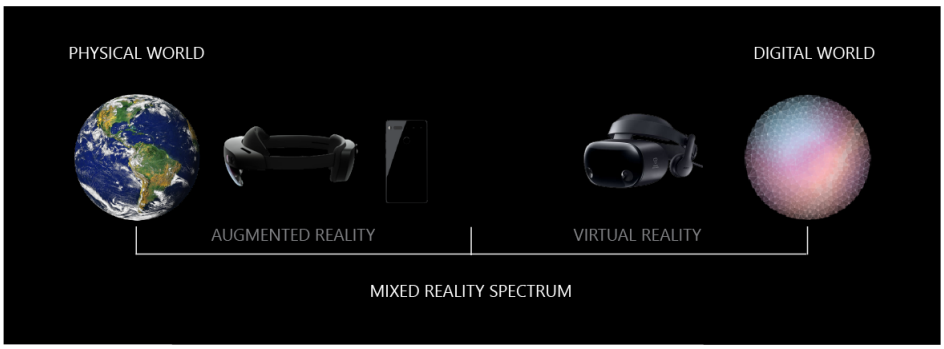


Figure 1.1: The spectrum of Virtual and Augmented Reality with the HoloLens on the left and some VR goggles on the right.[**?**]

Microsoft's HoloLens is a prime example of this, where developers can often create applications for entertainment purposes but it also provides a *Research Mode* extension, where the developer has hands-on low-level access to the sensors on the device. This extension attempts to bring low-cost consumer grade sensors to demanding fields, where contribution to AR and VR research could be furthered.

Helmet mapping systems (HMS) are also a new area of research which have their inspirational roots in helmet mounted displays (HMD) that are widely used in military aircraft pilots which have nearly immaculate accuracy and a low error tolerance.[**?**] Such systems are now being combined with a range of different sensors and worn by humans frequently for pedestrian and indoor applications.

Figure 1.2: The F35 HMD.[**?**]

In applications where movement is of key importance, sensors such as integrated measurement units (IMU) play a crucial role. The position, velocity and acceleration of bodies are derived from their motion through space and time and in order to describe this motion. Their *all-in-one* approach to many motion problems is their greatest source of selection and simultaneously their biggest criticized point, due to the inherent uncertainties of sensor measurements that come with it. [**?**]

Figure 1.3: A subject wearing the HMS with the HoloLens.

The performance of such sensors is not a problem resulting from the physics that define them, but rather is dependent on the strenuous and expensive process of manufacturing sensors which are bound to have material imperfections, assembly mismatches and discretization errors when they are transferred from the analog world to the digital world [**?**, p. 71,156]. These errors do not present themselves as worrying in consumer applications but rather in industrial fields where smaller errors will often lead to catastrophic failure of systems [**?**]. In such cases, IMUs in the order of hundreds of thousands of dollars are used and these errors are minimized below a certain tolerance. Same cannot be guaranteed for everyday usage, where these sensors would cost 3 to 4 dollars.

For applications happening outside, it is common to use the GPS signals obtained from at least 4 of the orbiting 32 satellites to have a positioning accuracy up to several orders of centimeters [**?**] to aid the sensors used to describe other behaviors of the rigid body that they are attached to. Aircraft and automobiles are the leading examples of this harmonious work, or in other, rather popular terms, of this sensor *fusion*. [**?**] In indoor applications, these signals are blocked

by not only surrounding buildings but they are also subject to quality diminishing effects such as multipath [**?**]. Owing to this problem, they are usually not utilized in such applications and other forms of sensor fusion are aimed.

Exteroceptive sensors, which derive meaning from surveying the surroundings and extract information from it, are the common replacement for the GPS, where they would similarly provide navigational information in the from of position updates [**?**, **?**]. Cameras and LiDAR are two main examples for this sensor family, due to their relatively low cost and high performance output.

In the absence of an aiding sensor that provides reliable information to the system, the successful combination of the sensors proves to be challenging, due to a list of limitations including:

- Plant-model mismatches
- Noisy sensor data
- Irregular sampling rates
- Dissimilar data acquisition platforms
- Nonlinearity of motion

Thus, owing to both economical and practical reasons, aiding sensors are always favored.

1.1 Motivation

In this thesis, the Microsoft Hololens 2 (abbreviated as HL) and the industrial-grade XSens Mti-G IMU are used together to extend the AR application possibilities of the HL. The embedded IMU of the HL is a low-quality consumer grade sensor which produces sensor readings that are highly dependent on the CPU load and how many sensors are accessed simultaneously. On the other hand, Xsens MTi-G is an industrial level sensor with factory calibration, dedicated low-level drivers, adjustable configuration, hardware triggering and GPS functionality. The Xsens IMU is rigidly attached to a wearable helmet. Both the HoloLens and the helmet will be worn simultaenously.

By subjecting both sensors to a number of tests, their noise and performance profile will be recorded. Then, by leveraging the same movement patterns that the wearer will undergo during various experiments, spatial and temporal calibration of the sensors will be achieved. It is the aim of the thesis to aid the HL with the Xsens IMU so that the AR applications developed on the HL can utilize the full range of an industrial grade sensor, while still being accessible to developers of all expertise. Establishing a data acquisition and spatiotemporal calibration framework for these sensors will also be beneficial should the need arise to fuse multiple sensors on the helmet, such as a LiDAR.

1.2 Problem Description

The spatiotemporal calibration between the HL and the HMS which houses the Xsens IMU is the focal point of this thesis. The main application for the combination of these sensors can be categorized in pedestrian motion, indoor localization, and indoor mapping. In order to solve this calibration problem, the task must be divided into constituent parts, due to the dissimilarities between the sensors. A unified data acquisition framework will be developed to capture both sensor streams simultaneously. Since the noise and performance profile of the Xsens IMU is known and the sensor stream is accessed with the ROS framework, capturing and benchmarking this data will rely on already established and performant platforms.

On the other hand, capturing data from the HL IMU will be done in a holographic Windows app, which publishes the data with no network connection and a clock that is not synchronized with an external source. Therefore, the temporal offset between the data streams must be systematically calculated so that each time the devices are started, the time difference between them is known. While the unified data acquisition framework will be developed on a Python environment after the raw data is captured from Xsens by utilizing the ROS framework on an Ubuntu computer and from the HoloLens by a native DirectX11 holographic app, the temporal calibration will require the gyroscope data, leveraging the fact that rigidly attached points will exhibit the same angular velocity.

Moreover, the axes of the HL IMU are not factory calibrated and unreliable results when compound motion are present. In cases where the HL IMU fails to capture the real motion, Xsens IMU will augment these intervals. As mentioned before, same movement patterns will be leveraged for this task. In addition, the acceleration values obtained from the IMUs will be compared with each other to verify the distance between the sensors. For this, rigid body kinematics will be utilized.

After the development of the unified framework, an experiment with a mannequin head will be carried out under a controlled environment and a further experiment will be done with a human subject and its results will be compared with a state-of-the-art gold standard calibration framework and the obtained results will be discussed.

1.3 Contribution

Not necessarily in the order of appearance but in the order of specificity, the following contributions are made by this thesis.

- An overall comparison of the state-of-the-art spatiotemporal calibration frameworks and their application possibilities.
- Development of a native holographic DirectX11 application for the data capture of the Research Mode sensors of the HoloLens.
- A data acquisition framework for both the HL and the Xsens.
- Temporal calibration of sensor streams .
- Noise and performance profiling for the IMUs and the HoloLens camera.
- Derivation of viable pedestrian motion patterns for use with these sensors.
- Calculation of sensor orientation via a complementary filter implementation.
- Benchmarking the spatiotemporal calibration framework with a state-of-the-art established gold standard and a rigid body experiment.
- Conclusive remarks on the generalization of merging multiple sensors of varying characteristic properties in a unified framework.

2 STATE OF THE ART

2.1 Clock and Time Synchronization

Computers, satellites and many major systems have their own clock, which is an oscillator, usually a quartz crystal, that produces square waves [**?**]. Clocks in different hardware configurations may be specifically designed to oscillate at different frequencies, which is set by their manufacturers. More often than not clocks are set to the same frequency on different hardware configurations in order to promote a unified framework in between. However, due to manufacturing imperfections, clocks may have differing frequencies even though they are the same model of the same manufacturer. Additionally, two clocks may oscillate at non-uniform frequencies, which produces a clock drift between them[**?**]. Clock skew is the phenomenon where the clock signal arrives at a component at a time point which is different than expected, thereby causing either a negative or a positive skew. And lastly, a clock offset is the delay between two clock signals.

The data originating from the sensors must go through the internal hardware, get timestamped from its internal clock (if the sensor has an internal clock), travel through either a wireless medium or on a cable, arrive at the data acquisition environment (usually a computer), get through the operating system delays, get timestamped by the operating system and then be shown to the user. On top of the clock synchronization problem, users must calculate these delays as well. There are some mitigating methods to combat these delays, like using a real time operating system (RTOS) and assuming a deterministic delay value. As the system becomes more complex, such measures must be taken to ensure the accuracy of the data, but in systems where commercial grade sensors and hardware are being used, these are rare to find.

With the advancement of technology, the tolerance for the inaccuracy in timekeeping has narrowed to the order milliseconds for usual daily activities, microseconds for critical applications such as stock exchange and booking of flight tickets [**?**, **?**]. Clock synchronization has become ubiquitous within the internet with established gold standards such as Network Time Protocol (NTP) discussed in the reference [**?**]. These gold standards have been in power for more than 30 years with proven results. The algorithms work in a network of computers, where each computer provides their timestamps in packets when they communicate with each other. In the distributed network of computers, one computer assumes the role of master with a source clock, while the other computers reference their time according to the master.

These protocols require that the communication occur in a bi-directional fashion, therefore the application areas of such established standards do not usually include consumer grade devices, where the sensors publish their data to a platform with either timestamps created by the operating system of this platform, or by timestamps created by the internal clock of these sensors, where this procedure is unidirectional. Such timestamping is bound to encounter delays such as:

- Operating system delays
- Transmission delays (the time it takes for electrons to travel through the cable or for EM waves to excite an arbitrary receiver)
- Clock skew and clock drift in between multiple timestamping sources
- Manual start-up delays

These delays may be critical to some applications, however they are usually ignored or assumed as a deterministic constant [**?**]. In high-grade sensors, a feature called *hardware triggering* is found, where a dedicated signal can be sent to trigger the data stream capture [**?**]. This also aids in the calculation of temporal offset in between multiple sensors. Although such a feature is present in Xsens MTi-G, the hardware architecture of the HL does not allow such an external signal to be interpreted in any way. In addition, at the time of writing this thesis, the HL does not have any networking capabilities that may bring an array of sensors together on a unified platform. [**?**]

Additionally, passive synchronization algorithms exist where an external event which is observed by multiple sensors is interpreted to create a waypoint for the sensors to base their timestamps off of. Such a method can be found in the following references [**?**, **?**, **?**]. These algorithms also assume that the sampling rate of the timestamps are in a small error tolerance, meaning that their sampling can be assumed non-changing. However, as it will be shown later, such an assumption will be impossible to make with the HL.

The term clock synchronization pertains to a broader range on the subject, accounting for clock skew and clock drift whereas time synchronization, temporal alignment and temporal calibration are the terms used to signify design patterns that take care of clock offset [**?**]. In the scope of this thesis, due to the hardware limitations, clock synchronization is not aimed but rather the calculation of the temporal offset in between two sensors takes priority.

2.2 Multisensor Fusion and Integration

The sensors, which capture the data produced within a multidisciplinary system are unique and require a form of augmentation to be processed together. Since sensors provide the system with information regarding the environment and inner system state, this information will be used for the propagation of the state into the future. Incorrect data capture may lead to faulty behavior, it may make a stable system unstable or even cause a catastrophic failure. In order to achieve this, various research has been done on the field of multi-sensor fusion and integration (abbreviated as MFI) [**?**].

MFI is not only important when it comes to processing unique and dissimilar data, but also to achieve inference of future data which is otherwise not possible had the sensors been processed individually. Another reason why fusion is of high priority is the fact that sensors have inherent uncertainties. These uncertainties range from systematic errors to random errors, but they all pose a significant threat to the integrity of the obtained data. However, these errors can be modeled by utilizing probabilistic modeling [**?**, **?**]. Therefore, fusing multiple data in order to eliminate their uncertainty is also a reason why MFI is an integral part of any multidisciplinary system.

There are numerous methods available for data fusion with probabilistic modeling, but they are all primarily an implementation of the Bayes' theorem. This theorem is a way to calculate the probability of an event, given that a prior knowledge of conditions regarding this event is available.[**?**] By following this rule, the data obtained from the sensors can be used to infer the system state and the environment in which the system is currently placed. A prominent and widely applied implementation of the Bayes' theorem is the Kalman Filter, which estimates the system state by calculating a statistical model of the time evolution of the system state. This approach has been used since the 1960s from its inception, and continuous augmentation and modifications are undertaken to make it an even better option for multisensor fusion methods. [**?**]

2.3 Calibration Frameworks

In the areas of autonomous driving, augmented reality and unnamed aerial vehicles, the fusion of cameras and inertial measurement units are usually favored [**?**]. These sensors complement each other in a way that they cover for each other's shortcomings.Motion can be accurately extracted from cameras at relatively slow speeds but they suffer from motion blur when fast movements are encountered. Additionally, their frames per second vary from model to model, meaning that in low-budget cameras the users are also sacrificing a lot of image quality. On the other hand, inertial measurement units (IMU) can pick up movements at high speeds. However, they suffer from high-frequency noise and low-frequency drift [**?**], which result in the accumulation of unbounded errors when numeric integration is applied. This relationship has led to many systems include them as a staple combination, albeit still leaving some room for other sensors. A popular sensor to use in the areas mentioned above is the LiDAR sensor, having the capability to produce point cloud data and at a fast sampling rate. These three sensors are arguably some of the widely used sensors in contemporary research.

Sensor latency often poses detrimental drawbacks to the accuracy of fusion algorithms, therefore forcing researchers to fulfill strict assumptions while designing the algorithms, like assuming known and fixed time delays between data streams [**?**, **?**, **?**]. Additionally, timestamps obtained from the sensors are usually subject to clocks with varying frequency, clock jitter and clock skew and transmission delays incurred on the way to the host computer [**?**].

Counteracting these delays falls under the category of clock synchronization where network time protocols and hardware synchronizations are used to achieve strict temporal synchronization between sensors. This demands that the sensors be networked with each other, meaning that they send and receive data and they must additionally fulfill some energy requirements, which are usually not available in consumer-grade systems due to their low-cost and mobile design. Moreover, hardware synchronization is indeed present in tactical-grade sensors, but they are expensive and not preferred in the majority of robotics applications [**?**, **?**, **?**].

On the other hand, spatiotemporal calibration deals with aligning timestamps from various datastreams[**?**]. This is usually favored in the majority of robotics applications due to the sensors not having hardware synchronization capabilities and them being not networked with each other, thereby eliminating the possibility of using time protocols.

There are popular sensor combinations, which aid each other in their ability to both extract meaningful information from the environment (LiDAR, camera) and infer the system state (GNSS, IMUs). These combinations are fused together on the base of probabilistic oder deterministic modeling. The algorithms developed for such combinations are usually exclusive for that set-up and it is cumbersome to apply the same approach to a different sensor combination. Therefore, a unified and generalized approach to combining multiple sensors in one framework is of high priority [**?**, **?**]. Developing spatiotemporal calibration algorithms can be summarized in three distinct categories [**?**].

- Closed-form solutions by reducing system complexity
- Correlation-based approaches
- Filter-based approaches
- Optimization-based approaches

Before discussing the related research in this area, let us a give a brief summary of the categories mentioned above. By reducing the system complexity, a sensor is rigidly fixed, therefore leading to the elimination of one or more degrees of freedom. Thus, certain noise factors are completely eliminated. In correlation-based approaches, signals are brought on top of each other and the objective is to maximize the cross-correlation between them. Since it is known that the sensors are undergoing the same movement, their data must correlate with each other.

The Kalman filter is a popular estimation method in the category of filter-based approaches where the data obtained from the sensors are used to determine the next state of the system [**?**]. Moreover, particle filters also play a significant role in this category but they are impractical in terms of computational cost when 6 degrees of freedom are to be considered [**?**]. Lastly, a cost function is derived which describes the error between the predicted and the observed state of the system and tries to minimize this error [**?**].

For closed-form solutions, an IMU is usually the center of attention, where it is placed on a rotation table an assumed to be fixed in space and the spatial alignment between the IMU and a visual sensor is easily derived from this set-up [**?**]. However, this closed-form solutions assume that the sensors are strictly synchronized, thereby rendering the temporal calibration impossible. Additionally, the calibration set-up for such approaches are cumbersome and error-prone, making them not favored in combined sensor suites.

Elmar et al. [**?**] have examined the cross correlation of the motion of an IMU and a camera to determine the temporal offset between these two sensors. This is a good method when the sensors are already synchronized, therefore they recommend the results of the work to be used for the initialization for optimization-based approaches. Also, they apply one dimensional

cross-correlation, which neglects the remaining two axes of the sensors, which contributes to a reduction in accuracy. In their earlier work [**?**], they outlined the discrete motion model obtained from the IMU and transferred it to the continuous-time domain with the help of B-splines while still keeping the original nonlinear format of the equations. After this, they applied a batch-optimization based solution for the problem of IMU-camera registration. However, their approach was limited to offline calibration.

Many variants have been proposed, in which an extended Kalman filter (EKF), an unscented Kalman filter (UKF) or their modified versions are utilized [**?**, **?**, **?**, **?**, **?**]. In these approaches, an IMU and a camera have been used for calibration, where a state vector has position, orientation, linear and angular velocity, bias of gyroscopes, bias of accelerometers, translation between the IMU and the camera and rotation between the IMU and the camera. Kelly et al. [**?**] have used a UKF to determine the transform between the sensors.

Kelly et al. [**?**] devised an algorithm where the curves of the camera and the IMU have been aligned in a three-dimensional orientation space by treating the calibration task as a registration problem. Originating from registration, they named their algorithm Time Delay Iterative Closest Point(TD-ICP) which handled the registration by employing a total least squares cost function. Furgale et al. [**?**] wanted to jointly estimate the temporal offset and the spatial displacement simultaneously and based on their earlier work [**?**]. They used B-splines to model the trajectories of the sensors in the continuous-time domain and the continuous-time batch optimization approach. They also used a camera and an IMU as their sensor suite. Their work has led to the open-source calibration toolbox by the name of Kalibr, which provides not only the calibration between an IMU and a camera, but also with the extension of Rehder et al. [**?**] they offer a solution for a multiple-IMU sensor suite. At the time of their writing, there was little work focusing on the fusion of multiple IMUs in one framework, therefore their toolbox was and still is a staple benchmarking software when it comes to calibrating multiple sensors.

Qin et al. [**?**] and Li et al. [**?**] have also treated the time offset as an additional state inside the state vector. Li et al. have used a variation of the EKF called multi-state constraint Kalman filter (MSCKF) which considered the geometrical constraints of the features in the environment between consecutive frames, whereas Qin et al. [**?**] formulated the problem inside a cost function where the IMU propagation has been modeled and this cost function was to be minimized. Both of these approaches have similarly used an IMU and a camera as their sensor suite.

Qui et al. [**?**] are influenced by the cross-correlation approach, but they take it a step further an calculate the three-dimensional motion correlation of the raw three-dimensional motion data obtained from the sensor. They focus on the ego motion, which is ubiquitous for all sensors. A central IMU is considered as a reference for all other target sensors and they are thus calibrated with each other accordingly. Additionally, their work is not limited to a specific sensor set-up and can be scaled up with multiple heterogenous and unique sensors. In their algorithm, temporal and rotational calibration is achieved.

With the light of the discussed related works, a table representing their advantages and disadvantages is presented below, where the proposed framework developed within the scope of this thesis is also compared with the state-of-the-art solutions.

Table 2.1: Comparison of state-of-the-art calibration frameworks

Framework	**Sensors**	**Method**	**Time Domain**	$\boldsymbol{R}$	T	**Camera tion**	**Augmenta-**
Kalibr [?]	IMU + Camera	Optim. Problem	Continuous	✓	✓	Board	
Kalibr [?]	multi-IMU	Optim. Problem	Continuous	✓	X	Board	
Elmar et al. [?]	IMU + Camera	1D cross-correlation	Discrete	✓	X	Board	
VINS [?]	IMU + Camera	Optim. Problem	Continuous	✓	✓	Natural	
Li et al. [?]	IMU + Camera	MSCKF	Continuous	✓	✓	Natural	
Qiu et al.[?]	IMU + Camera	3D trace-correlation	Discrete	✓	X	Natural	
Proposed	multi-IMU	3D cross-correlation	Discrete	✓	X	none	

In the preceding table, the methods contain optimization problems, a reduced-system 1D cross correlation (two axes of the gyroscope are not taken into account) a multi-state constraint Kalman filter and two cross-correlation implementations. The time domains signify that the problems are either solved with discrete samples or the data streams are transformed into differentiable splines and solved accordingly. $\boldsymbol{R}$ indicates the solution for the extrinsic rotation whereas $\boldsymbol{T}$ indicates the distance between the sensors. Since all except the proposed framework extract ego motion from cameras, the extraction method is gven on thr augmentation column with either a calibration board with a known pattern or natural landmarks.

In order to calculate the distance between multiple sensors, IMUs are not a great candidate due to their inherent noisy characteristics when numeric integration is executed to obtain position and velocity values from acceleration data. Thus, the solutions described in the table 2.1 *always* employ a sensor which has high accuracy in this regard, such as a camera which extracts ego-motion from a calibration board or natural landmarks. In cases where the motion is linear and not compound, such as driving a car on a straight line, IMUs can be used to deal with the translational motion to a certain degree, however this assumption is violated when a person, such as in the case of this work, is wearing the sensors and undergoing compound pedestrian motion and the traveled distances are insignificant.

In the proposed framework, the extrinsic rotation between the sensors can be calculated with information reliant on the IMUs only, however the distance between the sensors is pre-calculated and *validated* with rigid body kinematics equations. The results obtained from the framework developed in this thesis will be compared with [**?**] (from now on, referred to as Kalibr) for verification purposes, since Kalibr also calculates the extrinsic rotation in a similar manner. Kalibr employs a continuous-time optimization problem in which it transforms the discretely sampled IMU streams to B-splines and extracts ego motion from the camera via means of an April Tag calibration board with computer vision algorithms. Kalibr has been praised as a highly accurate framework, provided that sensors that are to be tested obey the framework's rules. However, the computational demand of this framework cannot be satisfied with consumer-grade CPUs such as that of HoloLens'.

The framework from [**?**] is suitable for real time capabilities but the code is not open source so there were no verification possibilities. Even though VINS [**?**] is on par with Kalibr's accuracy, it does not have a multi-IMU calibration step. Due to all of these reasons Kalibr was chosen as the golden standard for verification. For this verification, a human experiment with compound motion and a rigid body experiment with known angular velocity input will be carried out.

3 FUNDAMENTALS

In this section, the methodology underlying the spatiotemporal calibration framework will be explained in three parts. The first part explains the formalisms that describe rotation in three-dimensional space. The second part will discuss the fundamentals of digital signal processing pertaining to this work. Thirdly, the fundamentals of IMU and navigation technology will be explained.

3.1 Rotation in Three-Dimensional Space

Everything is relative to a reference point in our universe and so is movement. Observers at different points in space will observe motion according to their reference frames. In order to establish a common ground in which motion can be universally true is a necessary task if one wishes to fully describe the kinematics of bodies in motion [**?**].

Especially in the fields of robotics, aerospace and navigation, rotations play a significant role in describing the orientation of end effectors, the missiles traveling through the air and the orbiting of satellites around Earth. In addition to rotations, reasoning about the translational motion that a body may exhibit is also of importance. Depending on the application, the term *attitude* is used for orientation as well, especially in aerospace [**?**].

Various ways to describe rotations in space have been developed over the last century with the advent of the two world wars, where accurate navigation for aircraft was of high importance. Coincidentally, this also led to the technological explosion of inertial navigation technology and inertial measurement units [**?**].

Only the relevant rotation formalisms will be explained, although many more types of formalisms are present. This subsection encompasses the following:

- Euler angles and rotation matrices
- Direction cosine matrices
- Quaternions

3.1.1 Euler Angles and Rotation Matrices

The most intuitive way of understanding rotations in three-dimensional space can be accomplished by using the euler angles, which have been introduced by Leonhard Euler near the end of the 18th century [**?**]. Many variations have been introduced over the years which have improved Euler's proposition, but the theory remains the same. Euler angles can describe the orientation of a rigid body with respect to a reference frame and they can also describe the orientation of a frame to a reference frame.

The angles represent the rotation about an axis, in conventional cases, these axes will be defined as the x, y, z axes in an euclidian space. By successively turning about the axes, the final orientation of the rigid body will be obtained. The order of rotations around axes is also important, because different rotation sequences are not guaranteed to give the same orientation results.

To rotate a vector in three-dimensional space, three basic rotation matrices, each corresponding to an axis, can be arbitrarily combined with each other. These basic matrices are given as follows.

$$\begin{aligned}
\boldsymbol{R}_x(\alpha) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}(\alpha) & -\mathrm{s}(\alpha) \\ 0 & \mathrm{s}(\alpha) & \mathrm{c}(\alpha) \end{pmatrix} \\
\boldsymbol{R}_y(\beta) &= \begin{pmatrix} \mathrm{c}(\beta) & 0 & \mathrm{s}(\beta) \\ 0 & 1 & 0 \\ -\mathrm{s}(\beta) & 0 & \mathrm{c}(\beta) \end{pmatrix} \\
\boldsymbol{R}_z(\gamma) &= \begin{pmatrix} \mathrm{c}(\gamma) & -\mathrm{s}(\gamma) & 0 \\ \mathrm{s}(\gamma) & \mathrm{c}(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{3.1}$$

Where the cosine and sine functions are abbreviated as $\mathrm{c}(x) = \cos(x)$ and $\mathrm{s}(x) = \sin(x)$. Here, α rotates around the x-axis, β rotates around the y-axis and γ rotates around the z-axis. When compound rotation is to be undertaken, these matrices can be multiplied with one another to obtain a unique rotation matrix.

In aerospace applications, it is common to use the zyx convention, which unfolds as follows [**?**].

- Rotate around x-axis with α
- Rotate around the newly formed y-axis, y', with β
- Rotate around the newly formed z-axis, z', with γ

This compound movement can also be expressed in matrix multiplication.

$$\boldsymbol{R}_{\mathrm{EUL}}(\gamma,\beta,\alpha) = \boldsymbol{R}_z(\gamma)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha) = \begin{pmatrix} \mathrm{c}_\alpha\mathrm{c}_\beta & \mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{s}_\gamma - \mathrm{s}_\alpha\mathrm{c}_\gamma & \mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\gamma + \mathrm{s}_\alpha\mathrm{s}_\gamma \\ \mathrm{c}_\beta\mathrm{s}_\beta & \mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{s}_\gamma + \mathrm{c}_\alpha\mathrm{c}_\gamma & \mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\gamma - \mathrm{c}_\alpha\mathrm{s}_\gamma \\ -\mathrm{s}_\beta & \mathrm{c}_\beta\mathrm{s}_\gamma & \mathrm{c}_\beta\mathrm{c}_\gamma \end{pmatrix} \tag{3.2}$$

In this matrix multiplication, it is important that the sequence be done from right to left because rotation matrices are not commutative, meaning $\boldsymbol{R}_x\boldsymbol{R}_y \neq \boldsymbol{R}_y\boldsymbol{R}_x$.

Rotation matrices are a member of the special orthogonal group $\mathrm{SO}(3)$ where they perform rotations about the origin of a rigid body or a coordinate frame in three-dimensional euclidian space. They are also orthogonal and their determinant is always equal to 1 [**?**, **?**, **?**]. Rotation matrices are also *rotation operators*, meaning that they can also transform vector quantities, provided that it is a column vector. This is particularly useful when vector quantities such as acceleration and velocity need representation on multiple coordinate frames [**?**, p. 31]. An example is provided below.

$$\begin{gathered} \boldsymbol{w} = \boldsymbol{R}\boldsymbol{v} \\ \text{where} \\ \boldsymbol{v} = [v_1, v_2, v_3]^{\mathrm{T}} \end{gathered} \tag{3.3}$$

Moreover, a rotation matrix taking the quantities from frame a to frame b is denoted as ${}^{\mathrm{b}}\boldsymbol{R}_{\mathrm{a}}$.

Although rotation matrices are quite intuitive, they are usually disfavored due to their overweighing disadvantages. For certain angle configurations, a singularity will be produced in which the one degree of freedom will be lost. Another point is that many more euler angles conventions exist which confuse the users and make it hard to establish a standard rule among engineers [**?**].

3.1.2 Direction Cosine Matrices

Direction cosine matrices (abbreviated as DCM) are essentially rotation matrices but their euler angle convention is stripped away, in which they are only described as a matrix which performs rotation operation on its operand, be it a vector or a coordinate frame. In order to avoid confusion, they are usually denoted by $\boldsymbol{C}_{\mathrm{b}}^{\mathrm{a}}$ in which the DCM takes quantities from frame b to the frame a. An example which describes a trivial DCM can be found in the following equation.

$$\boldsymbol{C}_{\mathrm{b}}^{\mathrm{a}} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \tag{3.4}$$

Each element in this matrix is identical to a final rotation matrix which would have been produced with any euler convention, therefore it is safe to assume that DCMs and rotation matrices convey the same information, and thus can be used interchangeably.

3.1.3 Quaternions

Quaternions are an extension to complex numbers first introduced by William Hamilton in 1843 [**?**]. Rather than a consequtive sequence of rotation about axes, quaternions describe rotation about an arbitrary axis which pierces the rigid body or the coordinate frame, which would undergo this rotation. After augmentation, quaternions can be used exactly like DCMs, where they are multiplied with vector quantities.

Quaternion algebra is essentially the same with the complex numbers algebra that one is used to calculate. Regular complex numbers are described in the form of $z = a + ib$ where $i^2 = -1$. Regular complex numbers are also referred to as *hyper-complex numbers of rank 2* where the rank hints at the count of numbers used to describe the complex number. Similarly quaternions are referred to as *hyper-complex numbers of rank 4* due to the fact that a triplet of imaginary numbers and a scalar are required to fully describe the number. [**?**]

In the following equations, quaternion algebra primers will be shared and their further use in this work will also be discussed. Note that not all of the derivation steps of the algebraic equations are fully given and some steps are omitted for the sake of brevity. Further details may be found in the references [**?**, **?**, **?**].

A quaternion is described as follows:

$$\boldsymbol{q} = q_0 + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k} \tag{3.5}$$

where $q_i \in \mathbb{R}$ and $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$. Here the elements that comprise the quaternion are real numbers and the multiplication that describes the relationship between the triplets is

the quaternion product satisfying the quaternion ring multiplication. All quaternions in this thesis are assumed to be unit quaternions unless stated otherwise, satisfying the condition that the absolute value of the quaternion is always 1.

i, j, k denote the standard orthonormal basis for $\mathbb{R}^3$ satisfying the following conditions: $i = (1, 0, 0), j = (0, 1, 0), k = (0, 0, 1)$

Quaternions can also be described in the four dimensional number space with the following expression

$$q = (q_0 + q_1 + q_2 + q_3) \in \mathbb{R}^4 \tag{3.6}$$

It is often practical to separate a quaternion by its scalar and the triplets, where the scalar is left as q_0 and the triplets are grouped together under a vector $\boldsymbol{q} = q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}$. This sums up the quaternion as:

$$q = q_0 + \boldsymbol{q} \tag{3.7}$$

With quaternions, addition and subtraction are straightforward operations where each element affects its corresponding element, meaning that scalars will be added/subtracted with each other, and the triplets will be added/subtracted by their corresponding orthonormal basis.
Suppose that $p = p_0 + p_1\mathrm{i} + p_2\mathrm{j} + p_3\mathrm{k}$ and $q = q_0 + q_1\mathrm{i} + q_2\mathrm{j} + q_3\mathrm{k}$ are two quaternions and we would like to multiply them.

$$\begin{aligned} pq = p_0q_0 - (p_1q_1 + p_2q_2 + p_3q_3) + p_0(q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \\ +q_0(p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) \\ +(p_2q_3 - p_3q_2)\mathbf{i} + (p_3q_1 - p_1q_3)\mathbf{j} \\ +(p_1q_2 - p_2q_1)\mathbf{k} \end{aligned} \tag{3.8}$$

Quaternion multiplications can also be written in matrix form, where calculations look more familiar to the eye.

Suppose that r is a quaternion originating from the multiplication of p and q where they obey the 4-element notation with 1 scalar and 1 triplet described previously. The multiplication in matrix notation would unfold as follows:

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & -p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} \tag{3.9}$$

Taking the complex conjugate of a quaternion is the same as the complex numbers, where the imaginary part is reversed. This means that the triplet will be reversed in their sign. On the

other hand, taking the inverse of a quaternion will also result in conjugation.

$$\begin{aligned} q &= q_0 + \boldsymbol{q} \\ q^* &= q_0 - \boldsymbol{q} \\ q^{-1} &= q^* \end{aligned} \tag{3.10}$$

Taking the norm of a quaternions is described as follows.

$$N(q) = \sqrt{q^* q} \tag{3.11}$$

Quaternions, just like DCMs (remember that DCMs are unique and correspond to the final rotation matrix originating from the sequentially applied euler angles about consecutive axes) are a tool to describe vectors in one reference frame with respect to another. Depending on the accuracy and computer cost demands of the application, one or the other may be chosen. It was mentioned in 3.1.1 that these tools are called *rotational operators*. While DCMs can be directly pre-multiplied with a vector that we wish to convert from one frame to another by $w = \mathbf{A}v$, the same does not apply to quaternions, at least not without further augmentation.

The first augmentation will be to treat the vectors as the equivalent to the triplet of the quaternion and as the scalar part, a zero will be placed. The vector is now a quaternion and may undergo quaternion multiplication. Now suppose that we would like to convert this quaternion on the body frame of a vehicle to an arbitrary reference frame. The conversion of the vector on the body frame into the quaternion is expressed as $\mathbf{r}^{\mathrm{b}'} = 0 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ where the vector components on the x,y,z axes are transferred to the i, j, k orthonormal basis.

Now assume that this quaternion is to be expressed with respect to the reference frame, the reference frame must also house the augmented quaternion, obeying the same conversion described above, and it is expressed as $\mathbf{r}^{\mathrm{n}'}$. The quaternions may now undergo multiplication.

$$\mathbf{r}^{\mathrm{n}'} = \mathbf{q}\mathbf{r}^{\mathrm{b}'}\mathbf{q}^* \tag{3.12}$$

Here, $\mathbf{q}$ and its transpose represent the quaternion that rotates the vector in three-dimensional space with an angle about an arbitrary axis which pierces the origin of the frame. Moreover, quaternions may also be expressed in matrix multiplication as follows.

$$\mathbf{r}^{\mathrm{n}'} = \boldsymbol{C}'\mathbf{r}^{\mathrm{b}'} \tag{3.13}$$

where

$$\boldsymbol{C}' = \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{C} \end{pmatrix} \tag{3.14}$$

$$\mathbf{r}^{\mathrm{b}'} = \begin{pmatrix} 0 \\ \mathbf{r}^{\mathrm{b}} \end{pmatrix} \tag{3.15}$$

Here $\boldsymbol{C}$ represents the direction cosine matrix that rotates quantities from the body frame to the reference frame. It can be directly calculated from the terms of the quaternion with the following matrix expression.

$$\boldsymbol{C} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{3.16}$$

$$\mathbf{r}^{\mathrm{n}} = \boldsymbol{C}\mathbf{r}^{\mathrm{b}} \tag{3.17}$$

With the established matrix multiplication, quaternions can now be used to describe orientations of rigid bodies and transfer vectors from one frame to another.

All of the rotation formalisms can be converted to each other. It was mentioned that the 3.1.1 zyx convention was the most commonly used angle sequence in aerospace applications due to the convenience of rotating about the axes consequtively. It is possible to extract these euler angles from the DCM, as well as from quaternions. The three-way relationship can be expressed in the following equation.

$$\begin{aligned} \boldsymbol{R}_{\mathrm{EUL}}(\gamma, \beta, \alpha) &= \boldsymbol{R}_z(\gamma)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha) \\ &= \begin{pmatrix} \mathrm{c}_\alpha\mathrm{c}_\beta & \mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{s}_\gamma - \mathrm{s}_\alpha\mathrm{c}_\gamma & \mathrm{c}_\alpha\mathrm{s}_\beta\mathrm{c}_\gamma + \mathrm{s}_\alpha\mathrm{s}_\gamma \\ \mathrm{c}_\beta\mathrm{s}_\beta & \mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{s}_\gamma + \mathrm{c}_\alpha\mathrm{c}_\gamma & \mathrm{s}_\alpha\mathrm{s}_\beta\mathrm{c}_\gamma - \mathrm{c}_\alpha\mathrm{s}_\gamma \\ -\mathrm{s}_\beta & \mathrm{c}_\beta\mathrm{s}_\gamma & \mathrm{c}_\beta\mathrm{c}_\gamma \end{pmatrix} \\ &= \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \\ &= \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \end{aligned} \tag{3.18}$$

Each term belonging to a specific rotation formalism can be extracted from the 3.18 equation described above. Now it is fully possible to transfer a vector from one frame to another by

using three formalisms which can also be used interchangeably.

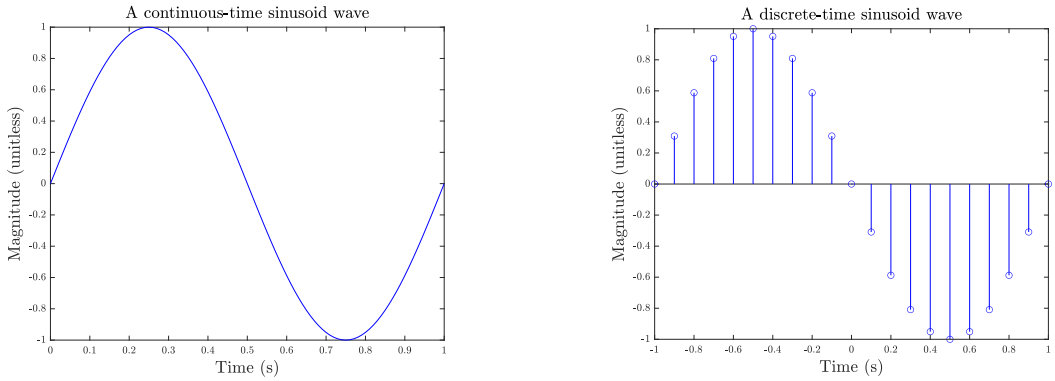
In this work, both of the formalisms are used extensively to describe the rotation and orientation of rigid bodies with respect to a reference coordinate frame, which will be assumed as an *inertial frame*, which is non-rotating and non-translating. The orientations of the HMS and the HL are described in DCMs and quaternions, which take acceleration and angular velocity vectors from the body frame of these devices to the inertial frame. Frames and their usages will also be explained in the section 3.3.4.

3.2 Digital Signal Processing

3.2.1 Basics

All phenomena happen in continuous-time but in the digital world they are transformed to the discrete-time domain and analyzed accordingly. Digital signal processing covers the tools and methods that are utilized to analyze such phenomena, in which the information obtained from the observation of these phenomena are displayed as a *sequence of samples*, also called as *signals*. The infinite nature of continuous time is decimated into finite and countable samples, which represent the original continuous-time signal. This procedure is named as *sampling* and the samples represent a slice of the original continuous-time information at the particular point in time where they are sampled.

Much of the work regarding digital signal processing has been attributed to the following references [**?**, **?**, **?**] and this section inspired by these references. Extensive details regarding signal processing can be found in the section *Sampling of Continuous-Time Signals* of [**?**], and *Basic Principles of Sampling and Sampling Rate Conversion* and *Design of Digital Filters for Decimation and Interpolation* of [**?**].



(a) Continuous-time

(b) Discrete-time

Figure 3.1: An arbitrary sinusoid wave in continuous and discrete time.

In the figure 3.1 we see an analog sine wave on the left which is displayed over 1 second and its discretely sampled version on the right. The signal has been sampled at a frequency of 10 Hertz meaning that 10 samples are taken each 1 second. The higher the sampling frequency, the nearer the approximation to the real continuous-time signal that we aim to replicate in discrete-time. A similar, real-world analogy can be made to solidify the concept of sampling of continuous-time signals. Suppose that a car is accelerating from 0 km/h to 90 km/h on a

highway. Each time the driver looks at the odometer, his eye *samples* the speed of the car at that particular point in time.

Assume that an arbitrary analog signal is represented as $x_{\mathrm{a}}(t)$, its discrete counterpart would unfold as follows.

$$\begin{gathered} \boldsymbol{x}[n] = x_{\mathrm{a}}(nT) \\ \text{where} \\ -\infty < n < \infty \end{gathered} \tag{3.19}$$

Here, the analog signal is sampled n times and the n$^{\text{th}}$ sample of the analog signal is represented as $\boldsymbol{x}[n]$. T signifies the sampling period, whereas $f = 1/T$ is the sampling frequency. The discrete-time signal is a sequence, meaning that all of n values are integers.

The conversion from the real world (also called as *analog*) to the digital world is done by devices which are called *analog to digital converters* (abbreviated as ADC). Since every device is prone to imperfections and hardware limitations, ADCs will not convert the signals completely accurately, but with the current technology that we have, the conversion results can be approximated as real representations of signals. [**?**]

Such signals, which are defined on the time domain, can be transformed into signals, which are defined on the frequency domain. Should the transform occur with analog signals, *Fourier transform* is used for conversion, whereas discrete signals require *discrete-time Fourier transform* for conversion. Since this work deals with discrete signals exclusively, the latter will be used for signal processing. By analyzing the frequency makeup of the signal, the sine and cosine components which are found in the signal can be deduced since all signals can be represented as a combination of sines and cosines.

Suppose that an arbitrary sensor stream has been captured and the contents of the stream are to be analyzed. The sequence of the stream samples can also be termed as a *vector* of data where $\boldsymbol{x}[n] = [x_1, x_2, \ldots x_{\mathrm{n}}]^{\mathrm{T}}$ signifies the vector of stream samples. This data vector will be broken up into its constituent sine and cosine parts. A corresponding Fourier-transformed data vector $\tilde{\boldsymbol{x}}[n] = [\tilde{x}_1, \tilde{x}_2, \ldots x_{\mathrm{n}}]^{\mathrm{T}}$ with frequency components will be established.

$$\begin{gathered} \tilde{x}_{\mathrm{k}} = \sum_{j=0}^{n-1} x_{\mathrm{j}} e^{\frac{-\mathrm{i}2\pi\mathrm{jk}}{\mathrm{n}}} \\ \text{where} \\ \omega_{\mathrm{n}} = e^{\frac{-2\mathrm{i}\pi}{\mathrm{n}}} \end{gathered} \tag{3.20}$$

Here, the k[th] Fourier coefficient is obtained by taking the sum of all j data points at the j[th] frequency multiplied with the k[th] frequency over the total number of samples. In addition, $\omega_{\mathrm{n}} = e^{\frac{-2\mathrm{i}\pi}{\mathrm{n}}}$ is the *fundamental frequency* of this particular data vector. Note that all elements of the data vector are an integer multiple of this fundamental frequency. In order to obtain the original data vector from the Fourier-transformed data vector containing frequency coefficients, each k[th] coefficient must be multiplied with the corresponding frequency and then added up with the rest of the terms [**?**]. The relationship between these vectors are defined in the following equation.

$$\begin{pmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega_{\mathrm{n}} & \omega_{\mathrm{n}}^2 & \ldots & \omega_{\mathrm{n}}^{n-1} \\ 1 & \omega_{\mathrm{n}}^2 & \omega_{\mathrm{n}}^4 & \ldots & \omega_{\mathrm{n}}^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_{\mathrm{n}}^{n-1} & \omega_{\mathrm{n}}^{2(n-1)} & \ldots & \omega_{\mathrm{n}}^{(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{3.21}$$

The sampling rates of signals can also be changed, provided that some intermittent transformation take the discrete samples to the continuous-time signal. A *bandlimited* signal will be constructed from these samples. If the original continuous-time signal has been sampled with a sufficiently high enough sampling rate, the signal will be reconstructed without any loss.

3.2.2 Change of Sampling Rates

Suppose that the sampling period of the signal in the equation 3.19 is to be changed and let this be denoted by T' and the changed signal be denoted by $\boldsymbol{x'}[n] = x_{\mathrm{a}}(nT')$. A straightforward approach would be to reconstruct the original signal back to $x_{\mathrm{a}}(nT)$ and sample this continuous-time signal again with T' sampling period. On the other hand, discrete-time operators can be employed to handle the sample rate conversion without reconstructing the signal as well.

Assume that this signal is to be *decimated* i.e. downsampled with an integer factor of M.

$$\begin{aligned} \boldsymbol{x}_{\mathrm{d}}[n] &= \boldsymbol{x}[nM] \\ &= x_{\mathrm{c}}(nMT) \end{aligned} \tag{3.22}$$

The new signal $\boldsymbol{x}_{\mathrm{d}}[n]$ is now decimated by a factor of M. Analogously, the same procedure can be executed if one wishes to *upsample* a signal, meaning that a higher sampling rate is desired. Suppose that an upsampling with an integer factor of L is aimed. The newly created signal is denoted as $\boldsymbol{x}_{\mathrm{u}}[n]$.

$$\begin{aligned} \boldsymbol{x}_{\mathrm{u}}[n] &= \boldsymbol{x}[\frac{n}{L}] \\ &= x_{\mathrm{c}}(\frac{nT}{L}) \end{aligned} \tag{3.23}$$

Upsampling a signal is also called as *interpolation* because new data points are extracted from an already present signal and missing information is created. If the procedure were to be done with non-integer factors, extra steps are to be taken, namely the intermittent interpolation and decimation to obtain an overall integer value.

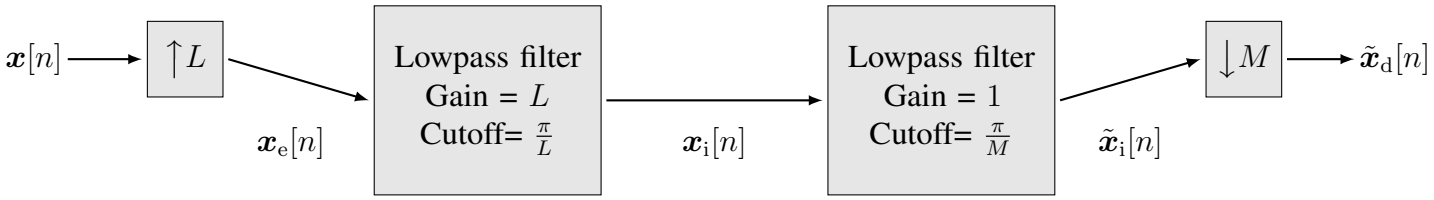


Figure 3.2: A non-integer factor resampling procedure.

In the preceding image, an arbitrary data vector $\boldsymbol{x}[n]$ is upsampled by a non-integer factor of $\frac{M}{L}$. Here, the term *filter* is introduced, which will be explained in the following subsection.

3.2.3 Digital Filters

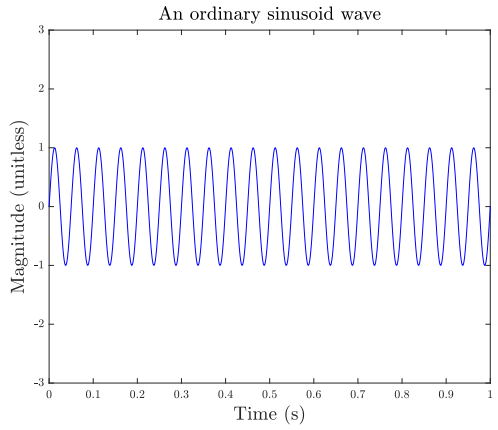
Digital filters are realized in software in which they perform operations on the aforementioned discrete-time signals. Many forms of filters exist but an exhaustive explanation of all of them would exceed the scope of this work, hence only the relevant filters and their usage will be explained. A list of the common filters explored in this work can be found below.

- Anti-aliasing filters.
- High-pass filters.
- Low-pass filters.

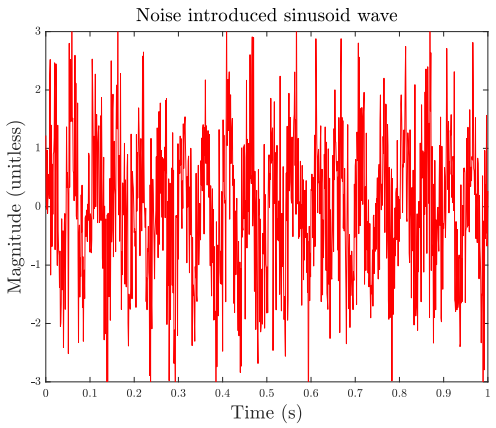
It was mentioned in 3.2.1 that signals could be transformed from the time domain into the frequency domain. After this transformation is finalized, filters operate on the frequency of the signal in that they allow some frequencies to pass and some frequencies to be attenuated. The limit at which frequencies are allowed to pass is termed as the *cutoff frequency*.

Anti-aliasing filters and low-pass filters could be categorized in the same class, in that they allow frequencies to pass below the selected cutoff frequency, whereas high-pass filters do the exact opposite, allowing higher frequencies to pass. Such filters are especially useful when dealing with sensor streams that are subject to external stochastic noise. An example can be found in the following image which displays how signals can be *denoised* by performing Fourier transforms on them and analyzing the frequency makeup of the signals.

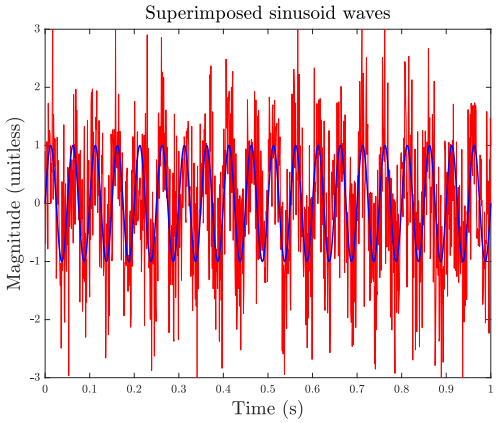
Let $x = \sin(2\pi 20t)$ be an ordinary sine wave with a frequency of 20 Hertz and an amplitude of 1. Let us corrupt this sine wave signal with a noisy signal of gaussian distribution with a mean of 1. All signals are now plotted against time in the following image.



(a) Ordinary wave.



(b) Noisy wave



(c) Superimposed waves

Figure 3.3: An ordinary noise introduced sine wave.

Applying the Fast Fourier Transform (a computationally efficient algorithm that executes the discrete-time Fourier transform) on the noisy signal results in the data vector $\tilde{\boldsymbol{x}}[n]$ which houses the coefficients of frequencies described in the equation 3.21. In order to visualize the frequency content of the signal, a *power spectrum density* analysis must be carried out on the obtained data vector. Such analysis describes how much of each frequency component is present inside the signal.

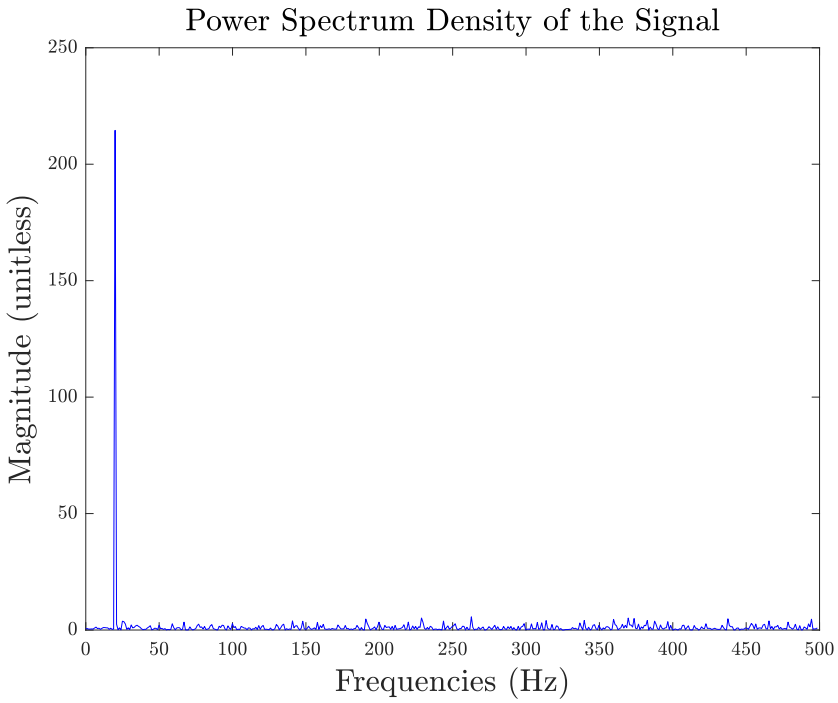


Figure 3.4: Power Spectrum Density of the noisy sinusoid wave.

The preceding image shows that, even though the noisy signal looked nothing like an ordinary sinusoid wave, there is a signal underneath with a frequency of 20 Hertz clouded with other signals with various frequencies scattered around, albeit not coming remotely close to the power of the 20 Hz frequency signal. A cutoff limit can be estimated in which we decide to let frequencies about a certain magnitude pass, and the other frequencies be eliminated. Consecutively, an inverse Fourier transform can be applied to transform the signal from the frequency-domain to the time-domain. This is the essence of digital filtering.

3.2.4 Cross Correlation

Suppose that two sensor streams produce the following data vectors $\boldsymbol{x}_{\mathrm{a}}[n]$ and $\boldsymbol{x}_{\mathrm{b}}[n]$ and the similarity between these streams are to be analyzed. By means of cross-correlation, which measures the similarity between these signals by treating the difference in similarity as the displacement of signals relative to each other [**?**, p. 68]. This is illustrated in the following equation.

$$\boldsymbol{R}_{\mathrm{x_a x_b}} = \sum_{n=-\infty}^{\infty} \boldsymbol{x}_{\mathrm{a}}[n]\boldsymbol{x}_{\mathrm{b}}^{*}[n-m] \tag{3.24}$$

Here $\boldsymbol{x}_{\mathrm{b}}^{*}$ signifies the complex conjugation of the discrete-time signal and $\boldsymbol{R}$ is the cross-correlation coefficient. This operation can be intuitively thought of as *sliding* one sensor stream over the other to see where they match the most. Had the sensor streams been perfectly identical and only exhibited a temporal offset in between them, the cross-correlation function would have a peak at this exact temporal offset. If we wanted to find this peak automatically at each time a cross-correlation function is executed, $\arg\max$ of the cross-correlation can be taken.

Moreover, since the preceding equation describes signals with infinite length, it cannot be used in real life applications. The same equation expressed in finite length is constructed as follows.

$$\begin{gathered}\boldsymbol{R}_{\mathrm{x_a x_b}} = \sum_{n=-N}^{N} \boldsymbol{x}_{\mathrm{a}}[n]\boldsymbol{x}_{\mathrm{b}}^{*}[n-m] \\ \text{where} \\ N = \text{num. of samples}\end{gathered} \tag{3.25}$$

3.3 IMU Technology

In this section, the usage of IMUs in navigation applications, rigid body kinematics from the sensor readings of IMU devices, definitions of frames and calibration of IMUs will be discussed.

Using IMUs to aid navigation is a prime focus point in an *integrated navigation system* (abbreviated as INS) where the main task of navigation is done by other sensors such as a GPS receiver, a UWB receiver or a sensor that captures visuals such as a camera.[**?**, **?**] Typically, these sensors have high accuracy but come with the burden of low sampling rates, hence the disadvantage of not being able to describe the intermittent navigation steps in between two updates. To cope with this problem, IMUs with high sampling rates are coupled with the aforementioned sensors to provide an accurate and fast navigation solution. [**?**]

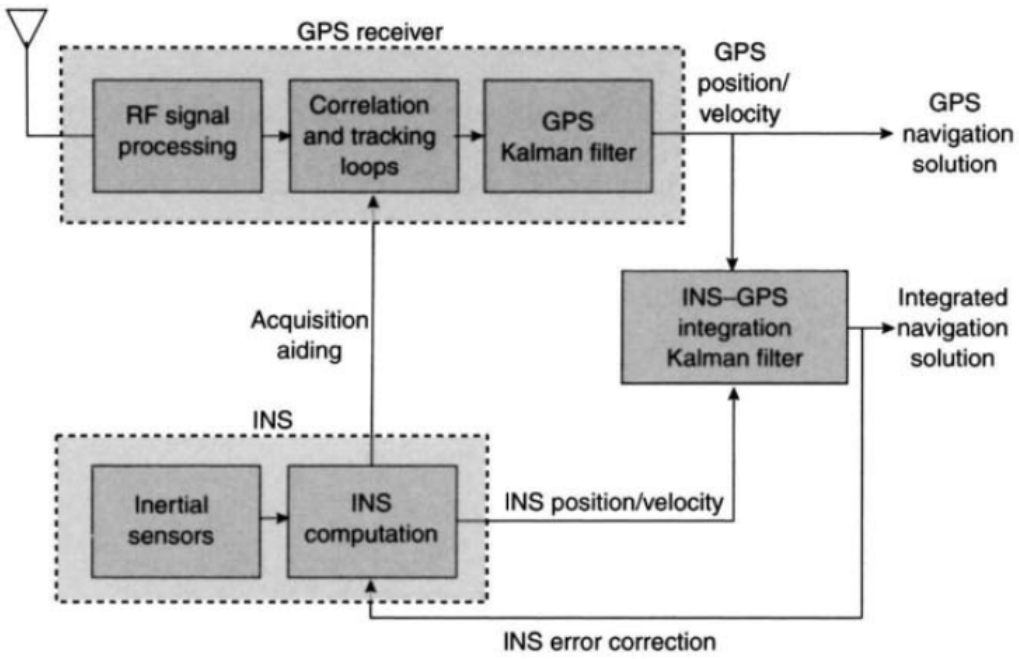


Figure 3.5: A loosely coupled INS-GPS integration architecture. [**?**, p. 412]

Depending on the application, a vast variety of IMUs can be selected. In areas such as submarine and missile navigation, IMUs must exhibit high levels of accuracy, but this is usually never the case for consumer-grade applications such as automobiles and robots [**?**, **?**]. In these cases, IMUs cannot be solely relied upon due to their noisy and biased output and must be fused with other sensors to create a fully functioning navigation solution. Latest research shows that

LiDAR and cameras provide good results in the fusion between IMUs for such consumer-grade applications [**?**, **?**]. In this work, such aiding sensors are not used, but rather two IMUs which varying characteristics are brought together. Therefore, the inner workings of an IMU will be focus of this section and other aiding sensors will be left out.

To determine the position and velocity of a rigid body moving through space with respect to a reference coordinate frame, some essential quantities must be calculated, otherwise the navigation solution will be incomplete.

- A clearly defined reference coordinate frame.
- Measurement of specific forces.
- Description of the gravitational field that the rigid body is subject to.
- Numeric integration of specific forces for velocity and position calculations.

3.3.1 Reference Frames

The motion of rigid bodies is defined relative to a frame of reference. In order to reason about our surroundings, a clearly defined coordinate frame must be given so that all motion can be understood *relative* to this particular frame. In navigation applications, the accurate definition of reference frames is an important task and some standards have already been established in the past century. Navigating on Earth must take into account the Earth's shape and dynamically changing gravitational field and these factors especially play a role in aerospace applications.

Not all reference frames are required for the application at hand, and in this work, not all of them are used. Due to the negligible movement around the curvature of the Earth and the negligible change of the gravitational field, only three reference frames are required to fully describe the motion in three-dimensional space [**?**].

- **Inertial Frame (*i frame, x,y,z axes*)** : This frame is accepted as a constant universally true frame in which all acceleration is referred to in classical physics. It is non-rotating relative to stars and its origin is coincident with the core of Earth.
- **Geographic Frame (*n frame, N,E,D axes*)**: This frame is an application specific local *navigation* frame where the axes are conventionally aligned with the North, East and Down directions. This frame lives on the surface of the Earth where D points toward the core of the Earth, aligned with the gravity vector, N points to the direction which is perpendicular to the D direction and E completes the right-handed orthogonal set. This can be intuitively, albeit informally, thought of as N pointing in front, E pointing to the right and D pointing towards the feet of the subject who is walking on Earth.
- **Body Frame (*b frame, r,p,y axes*)**: As the names of the axes would suggest, this frame coincides with the conventional roll, pitch, yaw of an arbitrary vehicle. The vehicle might travel on the surface, in air or it may orbit the Earth. within the scope of this work, the body frame coincides with the axes of the IMUs of both the HoloLens and the HMS. The IMU readings are *sensed* in the body frame, but must be *resolved* in the reference frame.

Other frames such as Earth frame, geocentric frame and tangent frame also exist but as mentioned, are irrelevant in the scope of this work and are therefore left out. Additionally, all of these frames can be converted to each other by defining the relationships between them. For example, a vector quantity living in the body frame can be expressed in the geographic frame by multiplying this vector with the DCM that carries the body frame to the geographic frame.

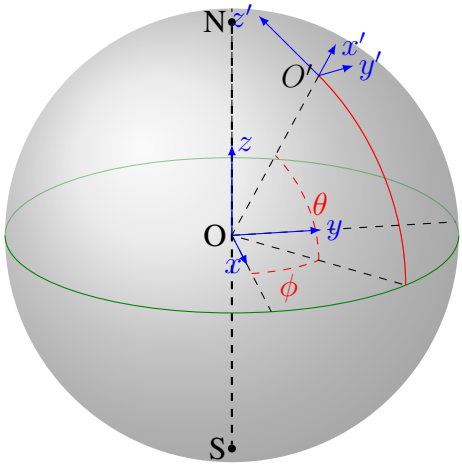


Figure 3.6: The inertial and the navigation frame on a representative Earth sphere.

In the preceding image, the inertial frame and the geographic frame (which will be used as the *navigation frame* from now on) can be seen on a sphere, which represents Earth. The frame coincident with the center of Earth is a non-translating, non-rotating inertial frame, whereas the navigation frame lives on the surface of Earth. A body frame, which might be attached to a vehicle, is commonly referred to the navigation frame first. If the vehicle is traveling immense distances in which changes in the shape of Earth and the gravitational field cannot be neglected, the navigation frame must also be referred to the inertial frame. In the scope of this work, such changes are insignificant and thus all references will be made with respect to the navigation frame and it will be accepted as a non-rotating, non-translating reference frame as well.

3.3.2 Rigid Body Kinematics

Accelerometers inside IMUs do not actually measure *acceleration* but rather they measure *specific force*. A proof mass with a known weight is suspended inside a casing with strings, and the tension on this string is measured as a result. The units of specific force are still ms^{-2} and they still obey Newton's second law of motion [**?**, p. 154].

$$\begin{aligned} F &= m\boldsymbol{a} \\ F &= m\boldsymbol{f} + m\boldsymbol{g} \end{aligned} \tag{3.26}$$

where m is the mass of the proof mass, $\boldsymbol{a}$ is acceleration, $\boldsymbol{f}$ is specific force and $\boldsymbol{g}$ is gravity and the specific force and the gravity can be merged to obtain the acceleration of an arbitrary rigid body. Thus, IMUs will measure the *every* acceleration exerted on its proof mass and in cases where it is required, gravitational acceleration will be compensated for. On the other hand, gyroscopes can directly measure angular velocity and its unit is $rads^{-1}$ or s^{-1}.

Setting up correct equations of motion for rigid bodies traveling in three-dimensional space is a key step in which rules of rigid body kinematics must be stringently followed. What may be true to an observer standing stationary on a reference frame may prove to be false to an observer who stands on a frame that is moving with respect to time. The rotation and translational motion of a rigid body determine how they are perceived in multiple reference frames [**?**, **?**]. *Angular velocity* is directly linked with the rotation, whereas *acceleration* determines the translational motion.

Angular velocity is denoted by the term $\boldsymbol{\omega}$ and acceleration by the term $\boldsymbol{a}$. Note that these quantities are column vectors, housing the scalar values in x,y and z axes in euclidean space.

$$\begin{aligned} \boldsymbol{\omega} &= [\omega_{\mathrm{x}}, \omega_{\mathrm{y}}, \omega_{\mathrm{z}}]^{\mathrm{T}} \\ \boldsymbol{a} &= [a_{\mathrm{x}}, a_{\mathrm{y}}, a_{\mathrm{z}}]^{\mathrm{T}} \end{aligned} \tag{3.27}$$

All points on a rigid body will exhibit the same angular velocity because as the object rotates, all points will sweep the same angle. However, the same claim cannot be made for acceleration. Additionally, if the object is undergoing rotation and translation at the same time, additional terms such as coriolis and centrifugal acceleration will be introduced to the system of equations. These terms are fictitious forces, but they are necessary to describe the full motion of points on a rigid body. These forces become apparent in cases like aircraft manoeuvres and even in merry-go-rounds. Such forces cannot be felt on the surface of Earth due to the insignificant difference between how fast Earth and the observer rotate.

Some primers to deal with the kinematics of points which are undergoing rotation and translation relative to a reference frame are discussed in the following equations. Suppose that point a and point b are on a rigid body, undergoing compound motion. The acceleration relationship of these points would unfold as follows.

$$\boldsymbol{a}_b = \boldsymbol{a}_\mathrm{a} + \boldsymbol{\alpha}_\mathrm{a} \times \boldsymbol{r}_\mathrm{b} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times r_\mathrm{b}) + 2\boldsymbol{\omega} \times \boldsymbol{v}_\mathrm{rel} + \boldsymbol{a}_\mathrm{rel} \tag{3.28}$$

Here, the acceleration of point *b* will be obtained. $\boldsymbol{a}$ represents acceleration with subscripts denoting their corresponding points, $\boldsymbol{\alpha}$ denotes angular acceleration, $\boldsymbol{r}$ denotes the position vector with its subscript denoting that the position vector points to that point, $\boldsymbol{\omega}$ denotes the angular velocity of the rigid body, which is the same for both points. $\boldsymbol{v}_\mathrm{rel}$ and $\boldsymbol{a}_\mathrm{rel}$ represent relative velocity and acceleration of point b with respect to point a, respectively. Note that cross-products between vectors are also present.
If the points on the rigid body do not move relative to each other, the final two terms indicating relative motion will be dropped out, leaving us with a simplified equation.

$$\boldsymbol{a}_b = \boldsymbol{a}_\mathrm{a} + \boldsymbol{\alpha}_\mathrm{a} \times \boldsymbol{r}_\mathrm{b} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times r_\mathrm{b}) \tag{3.29}$$

Angular velocity vectors have the property that they may also be represented with a *skew symmetric matrix* which allows them to undergo regular matrix multiplication, rather than cross-product operation. A skew symmetric angular velocity matrix is made up of its constituent parts and is denoted by $\boldsymbol{\Omega}$. Assume that an arbitrary angular velocity vector $\boldsymbol{\omega}$ is to undergo a cross-product operation with the vector $\boldsymbol{k}$. This relationship may be expressed as follows.

$$\boldsymbol{\omega} \times \boldsymbol{k} = [\boldsymbol{\omega}]_\mathrm{x} \boldsymbol{k} \tag{3.30a}$$

$$= \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} \tag{3.30b}$$

The skew-symmetric matrix of an arbitrary angular velocity vector is denoted as $[\boldsymbol{\omega}]_\mathrm{x}$. The same can also be done for angular acceleration. With this augmentation, the acceleration of point *b* expressed from point *a* will be

$$\boldsymbol{a}_b = \boldsymbol{a}_\mathrm{a} + [\boldsymbol{\alpha}]_\mathrm{x} \boldsymbol{r}_\mathrm{b} + [\boldsymbol{\omega}]_\mathrm{x} ([\boldsymbol{\omega}]_\mathrm{x} \boldsymbol{r}_\mathrm{h}) \tag{3.31}$$

The equation 3.31 can also be used to transfer acceleration vectors from one frame to another. In the scope of this work, since the inertial frame is assumed to be the only relevant frame,

all acceleration and angular velocity values are *sensed* on the body frame, with respect to the inertial frame. This means that the acceleration and angular velocity readings obtained from the IMUs are *sensed* on the body frame of these devices, as seen by an observer on the inertial frame.

HoloLens and HMS are assumed to be rigidly attached to each other where the magnitude of the position vector in between them never changes. If we would like to transfer the acceleration vectors of both of the devices to a reference frame, the position vector must be calculated in terms of the axes of this frame. However, this is not done in the scope of this work, rather the acceleration of one device is compared to the acceleration of another, hence the devices' body frames are assumed to be the reference frame to which the vector transfer will occur.

3.3.3 Calibration of IMUs

Theoretically, all of the equations of motion described previously, with the introduction of numeric integration to obtain linear velocity and position, should suffice to fully describe the motion of rigid bodies. However, this is almost never the case because sensors will always undergo imperfect manufacturing processes, thereby leading to incorrect sensor readings, which in turn will give incorrect specific force and angular velocity readings. With calibration, these errors are mitigated to a degree. The sources of these errors are vast but they are commonly categorized as follows [**?**, **?**, **?**].

- *Bias*: The constant output obtained from sensors in the abscence of external stimulation
- *Scale factor error*: the error in the ratio that relates sensor output to sensor input
- *Cross-coupling error*: also called as misalignment errors, signifying the erroneous output obtained from non-orthogonal axes
- *Drift*: a derived error type, describing the unboundedly growing error in numeric integration when accelerometer readings are integrated to obtain velocity and position.

Gyroscopes and accelerometers are both affected from such errors, albeit to different extents. The *true* sensor readings are thus clouded with incorrect readings and a generalization of how the readings are affected can be seen in the following equations.

$$\tilde{\omega}_{\mathrm{x}} = (1 + \boldsymbol{S}_{\mathrm{x}})\omega_{\mathrm{x}} + \boldsymbol{M}_{\mathrm{y}}\omega_{\mathrm{y}} + \boldsymbol{M}_{\mathrm{z}}\omega_{\mathrm{z}} + \boldsymbol{B}_{\mathrm{f}} + n_{\mathrm{x}} \tag{3.32a}$$

$$\tilde{a}_{\mathrm{x}} = (1 + \boldsymbol{S}_{\mathrm{x}})a_{\mathrm{x}} + \boldsymbol{M}_{\mathrm{y}}a_{\mathrm{y}} + \boldsymbol{M}_{\mathrm{z}}a_{\mathrm{z}} + \boldsymbol{B}_{\mathrm{f}} + n_{\mathrm{x}} \tag{3.32b}$$

Assume that the angular velocity ω_{x} about the x-axis and the acceleration a_{x} about the x-axis are to be measured. $\sim$ signifies the *measured* quantity. $\boldsymbol{S}_{\mathrm{x}}$ are the scale-factors for the x-axis,

$\boldsymbol{M}_{\mathrm{y}}$ and $\boldsymbol{M}_{\mathrm{z}}$ are the cross-coupling matrices for the y-axis and z-axis respectively, $\boldsymbol{B}_{\mathrm{f}}$ is the bias factor and n_{x} is the random noise bias. Note that the same procedure can be done for the y-axis and z-axis as well.
This equation 3.32 can also be thought of as having two distinct parts, where the first part comprises the deterministic errors inside the device and the second part pertains to the stochastic processes. Deterministic errors can be compensated for by subjecting the IMUs to extensive testing before they are released to the market. On the other hand, stochastic processes cannot be quantified by tests and must be tackled with other methods, specifically with filtering [**?**].
High-grade IMUs such as the Xsens Mti-G come with a datasheet with the necessary calibration information but further tests must be done to fully integrate such IMUs into navigation solutions. Benchmarking the performance of IMUs can be done by the so-called Allan Variance test [**?**] in which the IMUs are left rested on a stable surface for a few hours and their static noise profile is extracted. From this test, the bias factor and random noise parameters of the IMUs can be obtained. This test has been carried out for the Xsens and the HoloLens as well.

3.3.4 Orientation Determination

The measured acceleration and angular velocity quantities can be reformulated into *probabilistic* models, so that their true and noise parts can be observed individually. Let us augment the equation 3.32 into a probabilistic model. Assume that, with a slight abuse of notation, all of the quantities calculated here are on the body frame and they are calculated with respect to time, hence these sub and superscripts are omitted for the sake of an easier reading experience.

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \delta_{\omega} + n_{\omega} \tag{3.33a}$$

$$\tilde{\boldsymbol{a}} = \boldsymbol{a} + \delta_{\mathrm{a}} + n_{\mathrm{a}} \tag{3.33b}$$

$$n_{\omega} \sim \mathcal{N}(0, \Sigma_{\omega}) \tag{3.33c}$$

$$n_{\mathrm{a}} \sim \mathcal{N}(0, \Sigma_{\mathrm{a}}) \tag{3.33d}$$

$$\Sigma_{\omega} = \begin{pmatrix} \sigma^2_{\omega,\mathrm{x}} & 0 & 0 \\ 0 & \sigma^2_{\omega,\mathrm{y}} & 0 \\ 0 & 0 & \sigma^2_{\omega,\mathrm{z}} \end{pmatrix} \tag{3.33e}$$

$$\Sigma_{\mathrm{a}} = \begin{pmatrix} \sigma^2_{\mathrm{a,x}} & 0 & 0 \\ 0 & \sigma^2_{\mathrm{a,y}} & 0 \\ 0 & 0 & \sigma^2_{\mathrm{a,z}} \end{pmatrix} \tag{3.33f}$$

Here, δ_{ω} and δ_{a} represent the biases of the angular velocity and the acceleration respectively. n_{ω} and n_{a} are the random noise values, which are Gaussian distributed. Assuming that the

individual axes of the sensors are properly calibrated, the covariance matrices can be expressed as Σ_ω and Σ_a for angular velocity and acceleration, respectively. Additionally, it is assumed that the axes of the gyroscope and the accelerometer are coincident and no extrinsic rotation need be applied to axes.

These two probabilistic models will be used to find the orientation of the rigid body that the IMU is attached to, an in this case, this will be the orientation of the Hololens and the Xsens. Three distinct solution approaches are given to determine orientation [**?**, **?**, **?**].

- Solve an optimization problem in which an objective function that minimizes the error in orientation calculations is minimized. All variables must be present to start such an optimization problem, hence this is also called as *batch optimization*, because variables are processed in batches.
- Use extended Kalman filters (abbreviated as EKF). Kalman filters combine two independent estimates to form a weighted mean value. Extending this approach to non-linear systems form EKFs.
- Use complementary filters which rely on the noise profile of the IMUs to create an orientation estimation.

Solving an optimization problem and applying EKF both require probabilstic modeling of the acceleration and angular velocity values, whereas complementary filtering solely relies of raw sensor readings. These filters exploit inherent sensor characteristics, such as the long-term stability of accelerometer readings and the short-term stability of gyroscope readings. In contrast, they compensate for the drift encountered by numerically integrating the angular velocity values and the noisy output of the accelerometer. Essentially, two filtering techniques known as high-pass filtering and low-pass filtering will be applied to the raw sensor readings.

4 CALIBRATION FRAMEWORK

In the previous section, some fundamental primers regarding rotation in three-dimensional space, digital signal processing pertaining to cross-correlation and the principles of IMU were discussed. With those in mind, the calibration framework can be now be constructed.

This section is divided into multiple parts. First, a hardware initialization step will be discussed detailing how the raw IMU information originating from the Xsens IMU and the HL IMU are processed. Secondly, the estimation of the temporal offset between the sensors will be detailed. Consecutively, an implementation of a complementary filter inspired from [**?**] will be used to estimate the orientation of the sensors with respect to Earth's gravity and from this orientation the extrinsic rotation between the sensors will be extracted.

4.1 Hardware Initialization

The Xsens IMU is an industry-grade IMU with dedicated low-level drivers for both the Windows and the Linux environment. Since using the ROS software suite is a more convenient option that writing dynamic libraries to control the sensor on Windows, the raw information from the Xsens IMU is extracted on an Ubuntu computer with a Ryzen 5 2500U CPU and 8GB of DDR4 RAM running the ROS Melodic distribution. The configuration for the Xsens IMU is found in the following table.

Figure 4.1: The Xsens IMU

Frame	**Theoretical Sampling Rate**	**Average Sampling Rate**
ENU	100 Hz	82.9Hz

Table 4.1: Xsens Configuration.

In contrast to the Xsens IMU, applications for the HoloLens can only be developed on a universal Windows platform (UWP) where DirectX11 architecture and the proprietary COM (component object model) [**?**] architecture must be used. The Research Mode of the HoloLens allows developers to access the sensors and there is a generic template app that records the streams which can be found under the following link.[1] This template app does not record the IMUs, therefore a new app based on that one has been developed, which can simultaneously record IMU, RGB camera and other camera types (depth and point cloud). The modified app can be found under the following link.[2]

Since the Research Mode of the HL is still in its experimental stage, accessing more than one sensor simultaneously causes CPU overhead, which results in the erratic reduction of the sampling rate of all sensors. Normally, the advertised sampling rate of the HL IMU would be 12 Hz for the accelerometer, 20 Hz for the gyroscope and 5 Hz for the magnetometer and 30 FPS for the RGB camera. Additionally, no documentation of the magnetometer exists as of yet, therefore it will be excluded from further usage within the scope of this work.

Figure 4.2: The HoloLens worn by a subject.

[1] https://github.com/microsoft/HoloLens2ForCV/tree/main/Samples/StreamRecorder
[2] https://github.com/ozgunkaratas/HoloLens2ForCV/tree/main/Samples/StreamRecorder

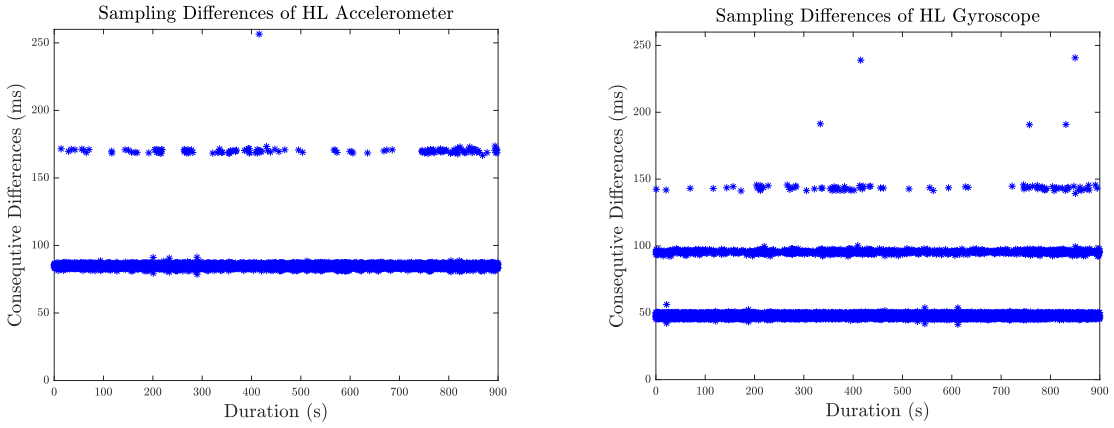
After stationary tests have been done on the HL with three sensor set-ups, the first being the IMU only, the second being the camera only and the last one using both of the sensors at the same time, some preliminary sampling rates have been obtained, which verifies that the sampling rate indeed varies with each experiment, depending on current battery percentage, temperature of the device and how long the device has been open for. The obtained sampling rate cannot be replicated, (unlike the Xsens which always publishes its data with respect to its internal clock with equidistant intervals) meaning that they have to be calculated dynamically for each experiment.

Layout and Duration	5 minutes	10 minutes	15 minutes
IMU	Acc:11.55 Hz Gyr:18.24 Hz	Acc:11.58 Hz Gyr:18.46 Hz	Acc:11.36 Hz Gyr:17.58 Hz
Camera	29.32 fps	27.80 fps	27.61 fps
IMU + Camera	Acc:10.80 Hz Gyr:14.21 Hz Cam:22.55 fps	Acc:10.92 Hz Gyr: 14.29 Hz Cam:22.41 fps	Acc:11.16 Hz Gyr:14.92 Hz Cam:23.51 fps

Table 4.2: Sampling Rates of HL Sensors

Nine experiments in total have been carried out to determine the sensor behavior, where the HL was left stationary on a table. It was estimated that the sampling rate of accelerometer fluctuates between 10 Hz and 12 Hz, whereas the gyroscope fluctuates between 13 to 15 Hz and the camera fluctuates between 19 and 23 Hz. Since this fluctuation cannot be pinpointed, it is dynamically calculated for each time the calibration framework is executed.

Another important factor pertaining to the sensor behavior of the HL is that the sensors *never* publish their data with equidistant timestamps. The sampling rates discussed in 4.2 are *average* rates over the full duration of the experiment. This problem is further exacerbated when multiple sensors must be accessed simultaneously. In the following images, the time differences between consecutive timestamps are plotted for the 15-minute-experiment, visualizing the sampling behavior.



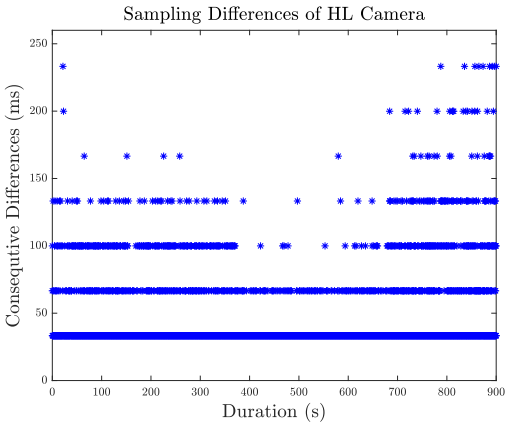


Figure 4.3: The sampling time differences of HL sensors in stand-alone mode.

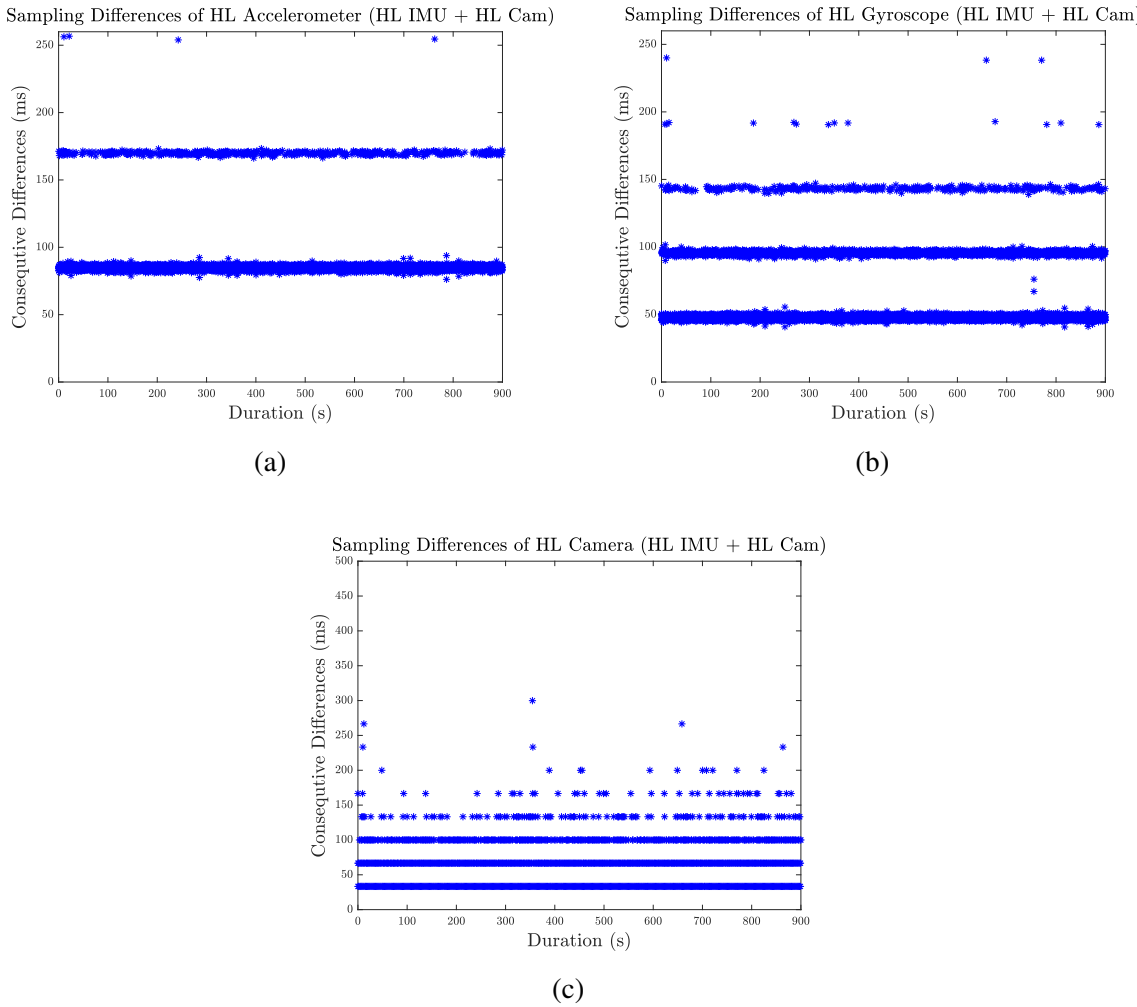


Figure 4.4: The sampling time differences of HL sensors in simultaneous operation.

In both images, it is seen that timestamping for all of the sensors is *not* equidistant. Some outliers amounting to as high as 2 seconds between consecutive timestamps have been omitted to show a clearer image. Such delay in a critical application has the potential to cause catastrophic failure. In order to combat this, linear interpolation of known data points is a viable option to reformulate the sensor streams with stable timestamps and data readings which would have corresponded to that timestamp had the sensor been able to publish its data at a stable rate. Linear interpolation is extensively used to handle raw data obtained from the HL in the scope of this work. This is also a mandatory step for the verification of the framework with the state-of-the-art solution, Kalibr.

It was mentioned in 3.3.3 that an Allan Variance test is a common tool to find the characteristic properties of IMUs. This test has been applied to both of the sensors to determine their properties. Such a test delivers the *random walk* and *white noise* parameters of an IMU, which signifies the drift occurring due to numeric integration of raw acceleration and gyroscope data and the random noisy output, respectively.

These two parameters are dependent on each other, in that the numeric integration of white noise results in the random walk. Random walk for both the accelerometer and the gyroscope grows unboundedly with time [**?**]. This test has been done on the MATLAB environment with the sensors left on an even table for about 2 hours. The script used to produce the results can be accessed under the following link. [3]

[3] https://github.com/rpng/kalibr_allan

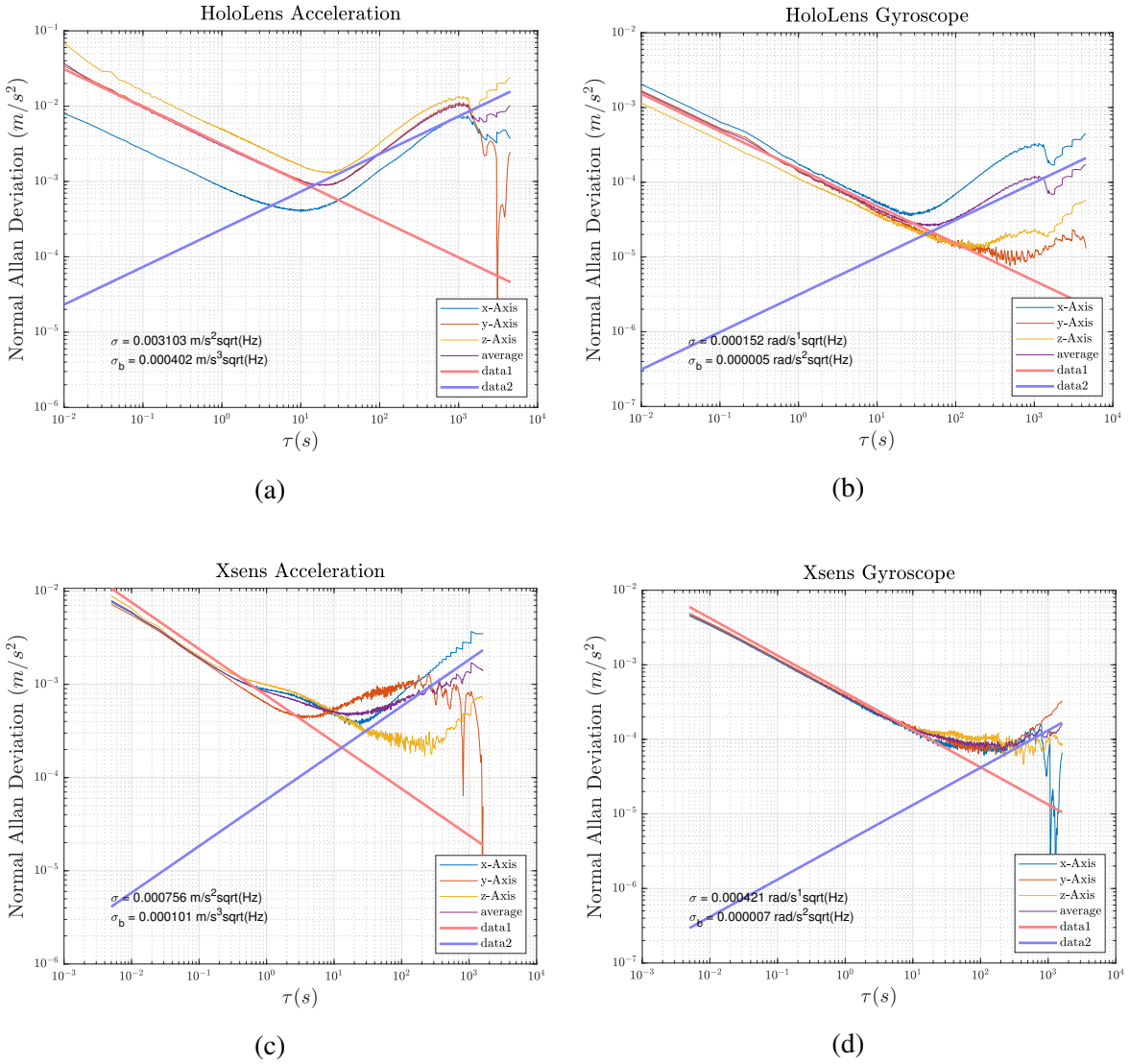


Figure 4.5: Allan Variance Results for the HL and the Xsens.

The corresponding results which are extracted from the Allan Variance test can be found in the following table.

	HL	**XSENS**
Accelerometer White Noise	$0.00610736 \frac{m}{s^2\sqrt{Hz}}$	$0.00075587 \frac{m}{s^2\sqrt{Hz}}$
Accelerometer Random Walk	$0.00055500 \frac{m}{s^3\sqrt{Hz}}$	$0.00010065 \frac{m}{s^3\sqrt{Hz}}$
Gyroscope White Noise	$0.00654623 \frac{rad}{s\sqrt{Hz}}$	$0.00042143 \frac{rad}{s\sqrt{Hz}}$
Gyroscope Random Walk	$0.00000649 \frac{rad}{s^2\sqrt{Hz}}$	$0.00000722 \frac{rad}{s^2\sqrt{Hz}}$

Table 4.3: Allan Variance Results for the IMUs

These values can be used in the design of both digital filters and Kalman filter variants in which cutoff-frequencies and probabilistic model parameters can be determined.

4.2 Data Acquisition Framework

Since the sensors cannot be connected to a single platform in which all of their data can be processed at the same time, a unified data acquisition framework must be developed. The Xsens IMU possesses an internal clock which starts at zero each time the device is powered up and synchronizes itself with the UTC time if it obtains a fix from satellites with its GPS receiver. Therefore, there are two options to timestamp the data originating from this sensor. Firstly, an unparsed data string can be sent from the device to the platform over a dedicated ROS topic, which is internally timestamped with Xsens' clock. Secondly, IMU values such as orientation, acceleration etc. can be directly published to corresponding ROS topics, which are timestamps with the ROS clock. Initiating a ROS node directly powers up the sensor.

After running some experiments, it was observed that the timestamping with the ROS clock introduces extra delays into the system originating from various sources, therefore the unparsed data string is used for raw data extraction. The string can then be parsed with regular expressions.

The HoloLens architecture is in contrast vastly different. As soon as the HoloLens starts recording, it starts writing the camera and the IMU data onto its storage with Windows timestamps. After the recording finishes, a few minutes are needed so that the HL can finalize saving the data. After this, the data must be extracted from the HL storage. From then on, the data is ready for post-processing.

In the following diagram, the data acquisition framework is shown. Note that, starting both sensors at the same time is impossible because a button on the holographic app and a button on the laptop keyboard must be pressed at the same time to start recording, No matter how accurate, this is prone to human errors. Additionally, after start-up there is an *unquantifiable* delay between pressing the button and the sensors publishing their data to their respective platforms. Moreover, the HL clock exhibits a non-negligible clock drift, which can only be remedied by connecting the HL with the internet on regular intervals. At the time of writing this thesis, the HL suffered problems connecting with the internet, therefore other methods to remedy this problem have been devised, namely a coarse alignment scheme in which the experiment periods were used to determine the temporal offset between two sensor streams.

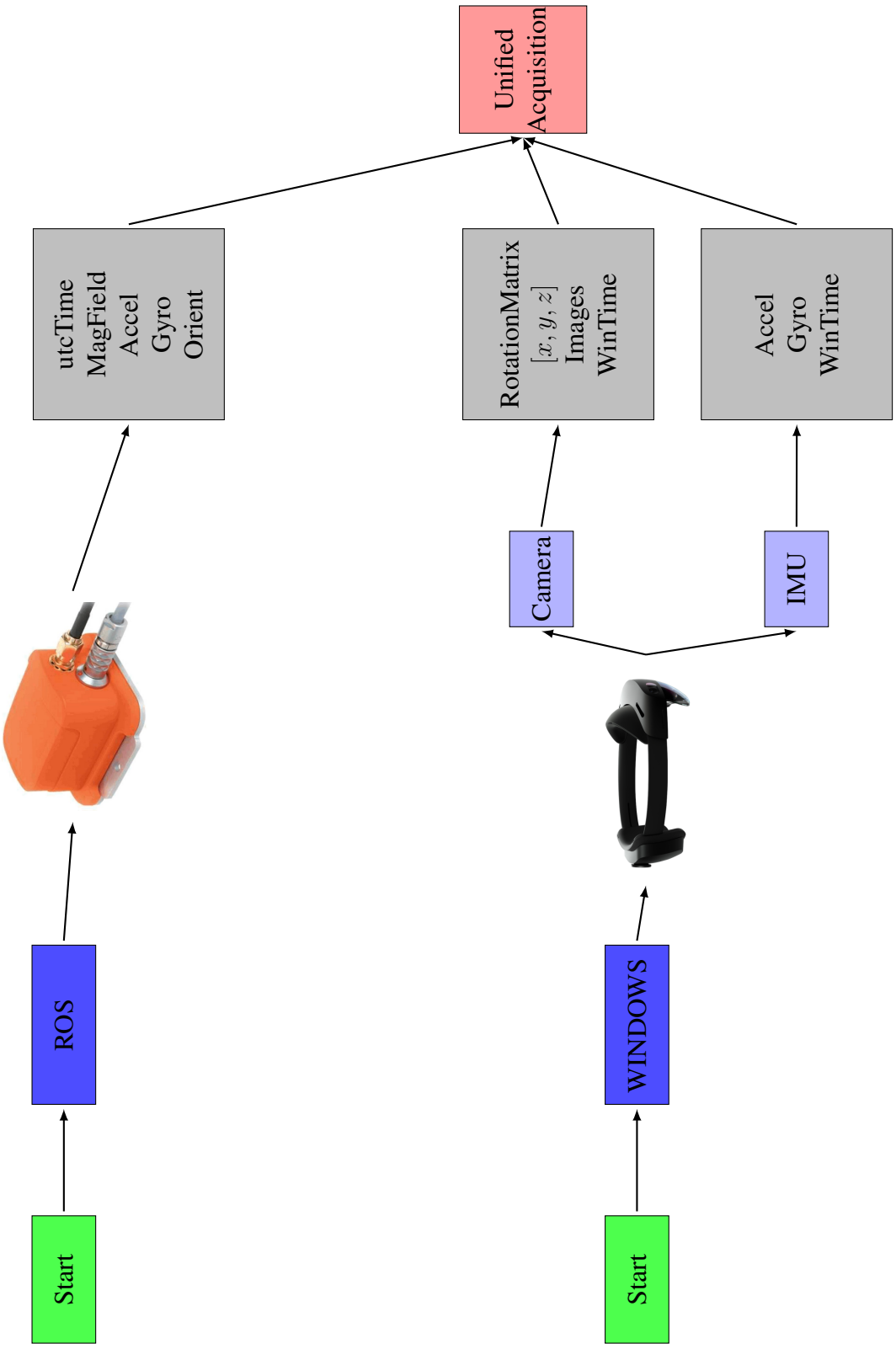


Figure 4.6: The Unified Data Acquisition Framework.

Gray blocks signify the data obtained after the experiment is finished. Note that the HL can stream both camera and IMU data. Each time an app is started which utilizes the HL camera, the it establishes a *scene* regarding the surroundings of the wearer and positions itself with respect to this scene. It takes into account how tall the wearer is and where the walls are located. The internal scene understanding algorithm of the HL then supplies the camera with positional information. Additionally, a rotation matrix which gives the orientation of the camera with respect to the scene is also shared within the stream recorder app. Thus, the pose of the HL with respect to the scene can be deduced.

Moreover, the HL IMU publishes its data erratically due to the CPU overhead mentioned previously. In order to extract coherent information from the HL IMU, not only does the accelerometer require linear interpolation but it also needs *upsampling* to match the gyroscope sampling rate since there is an internal sampling rate mismatch between accelerometer and gyroscope readings.

Now that a framework is established which collects raw data from both of the sensors, they can be used for further post-processing.

4.2.1 Coarse Alignment

In the previous section, it was mentioned that due to the clock differences between the dissimilar platforms, a coarse alignment was necessary to bring the sensor streams together. If both of the sensors were running on ROS, it would be possible to get the first timestamp of the streams to obtain a deterministic offset between the start of the streams. Since the drift of the HL also plays a role in skewing this difference significantly as time goes on, raw streams cannot be processed without augmentation.

In the following algorithm, the method of coarse alignment is shown, in which the experiment periods are explicitly compared with each other. The excess amount of time (and the corresponding samples) is trimmed off and the first timestamp of the XSens IMU is assigned to the first timestamp of HL IMU. Another viable option would be to directly compare the creation times of the rosbag from the Ubuntu computer and the IMU readings from the HL storage, but this was ruled out because the HL clock drift was making this approach inapplicable.

Algorithm 1: Coarse Alignment

Data: Gyr. and acc. streams, Stream timelines, Sampling rates
Result: Coarse aligned streams

begin
  Find last timestamp
  $lastStamp_{ROS} = utcTime[-1]$
  $lastStamp_{HL} = winTime[-1]$
  $initDiff = lastStamp_{HL} - lastStamp_{ROS}$
  if $initDiff > 0$ **then**
    Xsens ran for more time
    $longerStreamTime = initDiff$
    $longerSamplingRate = xsens$
  end if
  else if $initDiff < 0$ **then**
    HoloLens ran for more time
    $longerStreamTime = initDiff$
    $longerSamplingRate = HL$
  end if
  $excessSamples = longerStreamTime * longerSamplingRate$
  $del\,(excessSamples)\,from\,longerStream$
  return *coarseAlignedStreams*
end

This algorithm assumes that the provided sensor streams have already been linearly interpolated and that the accelerometer has been upsampled to match the gyroscope's sampling rate. In linear interpolation, the known timestamps are used as *query points* (also known as *anchor points*) to create new data in between these timestamps The total sample count of the gyroscope is used for the accelerometer upsampling. Upsampling a discrete-time signal has been discussed in the subsection 3.2.2. The same method is used here as well.

4.2.2 Temporal and Extrinsic Rotation Calibration

Initiation

By exploiting the fact that two points attached to a rigid body will exhibit the same angular velocity, we can use the gyroscope data from the sensors to achieve temporal calibration. Note that this method is extensively used in *all* of the frameworks discussed in the section 2.3. Each time the HMS and the HL are worn, their orientation with respect to the ground and also with respect to each other will be different. However, after the devices are worn and tightly fixed on the head, they can be assumed as a single rigid body undergoing motion. In the following image, the misalignment of the sensor axes with respect to each other is illustrated. Note that this misalignment does not correspond to the misalignment discussed in the section 3.3.3 which described the misalignment from the orthonormal basis.

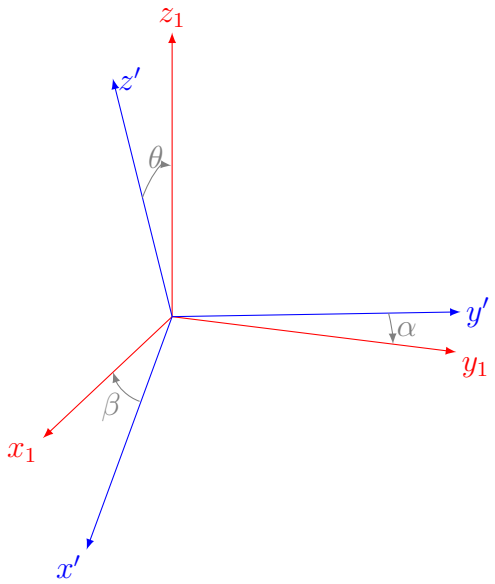


Figure 4.7: Representation of axis misalignment between the HL and the Xsens.

Here, suppose that the individual axes are misaligned by arbitrary angles. In order to obtain the rotation matrix which rotates the axes from one frame to another, the orientation of both of the sensors must be known with respect to a known reference frame. Since the IMU values measure the true acceleration and angular velocity as discussed in the section 3.3, the raw measurements can be used to estimate the orientation of the devices with respect to the inertial frame. It was also mentioned in the section 3.3.1 that the navigation frame can be substituted with the inertial frame since the sensors do not travel significant distances over the surface of Earth.

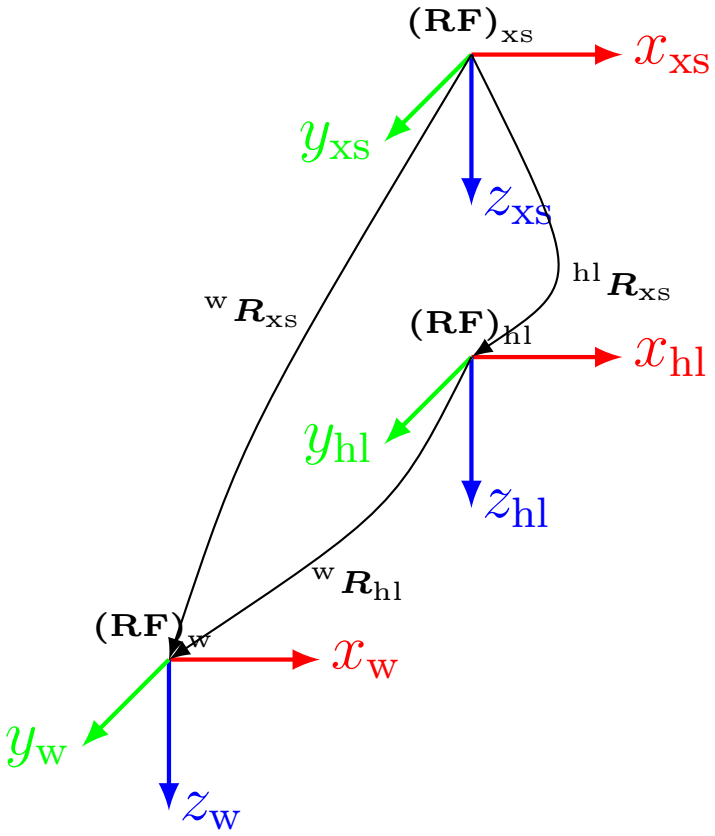


Figure 4.8: Reference frames in an arbitrary experiment scenario.

In the preceding image, the reference frames of the navigation frame, the HL frame and the HMS frames are given in an arbitrary experiment scenario where $(\mathbf{RF})_{\mathrm{w}}$ describes the *NED* frame mentioned in the section 3.3.1 and for the sake of simplicity, it is also termed as *world frame* hence the subscript *w*. $(\mathbf{RF})_{\mathrm{hl}}$ represents the reference frame corresponding to the HL gyroscope axes and $(\mathbf{RF})_{\mathrm{xs}}$ represents the reference frame corresponding to Xsens gyroscope axes. Three distinct rotation matrices can be observed from the image, transferring vectors from one frame to another. By chaining these rotation matrices where an intermittent reference to the world frame is carried out, the extrinsic rotation between the Xsens and the HL can be achieved.

$$^{\mathrm{hl}}\boldsymbol{R}_{\mathrm{xs}} = \left[^{\mathrm{w}}\boldsymbol{R}_{\mathrm{hl}}\right]^{-1\,\mathrm{w}}\boldsymbol{R}_{\mathrm{xs}} \tag{4.1}$$

When the gyroscope axes are aligned, comparing them to each other should be straightforward by means of cross-correlation, because they will measure the same angular velocity.

After starting the sensor streams on both of the devices, the wearer can stay still for a few seconds so that the orientation of the devices keep still as well. From both of these orientations, the extrinsic matrix $^{\mathrm{hl}}\boldsymbol{R}_{\mathrm{xs}}$ will be initialized, which aligns the gyroscope axes of the sensors. In order to carry out this task, an implementation of a complementary filter is used. [**?**]

Complementary Filtering

Complementary filters are an alternative to extended Kalman filters when orientation is to be extracted from IMU readings due to their easy yet effective set-up. Two main complementary filter implementations [**?**], [**?**] are extensively used in both consumer-grade applications and also with IMUs that are medium to high quality although the theory dates back . In addition, both of these complementary filter implementations can be found inside a software stack under the following link[4], indicating that these are also an established standard in ROS applications.

Within the frame of this work, a complementary filter inspired from the reference [**?**] has been implemented with slight modifications to the original source code to handle HoloLens and Xsens data simultaneously, by creating a `class` which handles any raw IMU readings, provided that the noise profile of the IMUs are given as well. Thus, this makes the complementary filter a viable alternative to Kalman filter variants not only due to the real-time applicability but also because the performance does not suffer when a KF variant is abandoned. Additionally, the complementary filter proposed in this work does not need magnetometer readings to estimate orientation, which is better suited for the HL, since as previously discussed, magnetometer data cannot be interpreted due to a lack of documentation. Moreover, KF variants are not

[4] http://wiki.ros.org/imu_tools

favored in applications that suffer from computation limitations such as drones and consumer-grade applications[**?**, **?**, **?**] and with regards to the HoloLens' erratic performance metrics, the complementary filter proves to be the suitable choice.

By utilizing the noise profile of the sensors, a series of low-pass and high-pass filter implementations can be applied to the raw readings to establish a refined orientation estimation, unlike KF variants which require a probabilistic model of the sensors. A *gain*, which can be adaptive or constant, is introduced to the system which constantly corrects the raw readings to obtain a better orientation estimation.

As explained in the subsection 3.1.3, quaternions are used to rotate vectors from one reference frame to another and in this case, the reference frames are the body and the world frame, respectively. The following equation shows the vector transform regarding the accelerometer readings.

$$[0\,0\,1]^{\mathrm{T}} = \boldsymbol{R}(q_{\mathrm{acc}})[a_{\mathrm{x}}\,a_{\mathrm{y}}\,a_{\mathrm{z}}]^{\mathrm{T}} \tag{4.2}$$

In the preceding equation, we see that the rotation matrix which has been converted from the quaternions (see 3.18) transforms the raw acceleration readings into the normalized gravity vector, which points downwards, signifying the z-axis. A similar equation can be constructed for the magnetometer readings, albeit this is not done within the scope of this work and the reduction in accuracy is negligible. This orientation obtained from the accelerometer readings is assumed as a predictive orientation, where the gyroscope readings will be used in a succeeding step to correct it.

Regarding the reference [**?**, p. 42] and [**?**], the quaternion derivative can be calculated as follows.

$${}^{\mathrm{a}}_{\mathrm{b}}\dot{\boldsymbol{q}}_{\omega,t_k} = -\frac{1}{2}{}^{\mathrm{a}}\boldsymbol{\omega}_{q,t_k} \otimes {}^{\mathrm{a}}_{\mathrm{b}}\boldsymbol{q}_{\omega,t_{k-1}} \tag{4.3}$$

Here, $\otimes$ signifies that the operation is quaternion multiplication. The quaternions take vectors from the frame a to the frame b. ω indicates the gyroscope readings and the subscripts indicate readings at the k$^{\text{th}}$ time instance.

In order to merge the quaternions obtained from the preceding equations, the following equation is constructed.

$${}^{\mathrm{a}}_{\mathrm{b}}\boldsymbol{q} = {}^{\mathrm{a}}_{\mathrm{b}}\boldsymbol{q}_{\omega} \otimes \widehat{\Delta\boldsymbol{q}}_{acc} \tag{4.4}$$

In the preceding equation, the delta quaternion is obtained by LERP operations and intermittent comparison with Earth's gravity vector. The reader is directed to the reference [**?**] for intermittent calculation steps.

As seen in the figure 4.8, two rotation matrices that take the vectors from Xsens' body frame and HoloLens' body frame are used for orientation estimation with respect to the world frame. The quaternions calculated from the preceding functions are transformed into rotation matrices. Right after the data streams start on the devices, the wearer can create a stable interval where he stands still for a few seconds and in this interval, an accurate orientation estimate with respect to the world frame can be achieved. This will be then used to rotate both the gyroscope and accelerometer raw reading vectors and align them with each other. ${}^{\mathrm{hl}}\boldsymbol{R}_{\mathrm{xs}}$ described in the same image signifies the extrinsic rotation matrix between the body frames of the IMUs.

The following algorithms gives a brief look as to how the stable region extrinsic rotation alignment works.

Algorithm 2: Stable Region Extrinsic Rotation Alignment

Data: Gyr. and acc. streams and sample counts, Sampling rates
Result: Extrinsic rotation matrix, rotation-aligned streams

begin
  Instantiate sensor filter objects
  $hl\_Filter = compFilter.init(noiseparameters, samplerates)$
  $xsens\_Filter = compFilter.init(noiseparameters, samplerates)$
  Get quats and convert to rotation matrix
  $q\_0, q\_1, q\_2, q\_3 = hl\_Filter.execute()$
  $hl\_StackedMatrix = hl\_Filter.solveRot(q\_0, q\_1, q\_2, q\_3)$
  $q\_0, q\_1, q\_2, q\_3 = xsens\_Filter.execute()$
  $xsens\_StackedMatrix = xsens\_Filter.solveRot(q\_0, q\_1, q\_2, q\_3)$
  Solve for extrinsic rotation matrix
  take in 1 second of readings 5 seconds after streams start
  average these values to find the avg_orientation
  $R\_hl\_xsens, avg\_R\_hl\_xsens =$
   $xsens\_Filter.solveOrient(hl\_StackedMatrix, xsens\_StackedMatrix)$
  Correct streams
  $new\_accels = matmul(avg\_R\_hl\_xsens, acc\_streams)$
  $new\_gyros = matmul(avg\_R\_hl\_xsens, gyr\_streams)$
  return *Extrinsic rotation matrix, new_accels, new_gyros*
end

Temporal Calibration

Now that the extrinsic rotation matrix has been obtained, cross-correlation can be executed to determine the temporal offset between sensor streams.

Algorithm 3: Cross Correlation

Data: Coarse aligned streams, Stream timelines, Sampling rates
Result: Aligned streams, Correlation results, Delay in seconds

begin
  Cross correlation of streams
  for $axis \in GyroAxes$ **do**
    $xCorr = correlate(coarseAlignedxsensAxis,\ coarseAlignedHLAxis)$
    $index = \arg\max(xCorr)$
    if $index > 0$ **then**
      Xsens is leading
      $Append\ zeros * index\ to\ xsens\ on\ end$
      $Delete\ same\ amount\ from\ other\ side$
      $shiftedStream = appendAndDeletexsens$
    end if
    else if $index < 0$ **then**
      HoloLens is leading
      $Append\ zeros * index\ to\ xsens\ on\ beginning$
      $Delete\ same\ amount\ from\ other\ side$
      $shiftedStream = appendAndDeletexsens$
    end if
    Determine correlated delays
    delay = index / xsensSamplingRate
  end for
  return *delay, shiftedStreams, CorrResults*
end

In the preceding algorithm, how the cross-correlation algorithm is constructed is shown. Note that, in order to compare two discrete-time signals with each other their sample count must match with each other. Referring to the equation 3.25, we see that for finite discrete-time signals, n must be the same for both of the signals. In software implementation, having two signals that are different in length will commonly result in an error, or the programming language will silently add zeros to either the end of the signal or the front of the signal to match the lengths. Since the Xsens IMU and the HL IMU are vastly different in their sampling rates, this procedure can be done by either downsampling the Xsens IMU or upsampling the HL IMU. Upsampling the data will produce intermittent data points in between known data points. Not only will the

data vector be upsampled, but also the timestamps, therefore plotting the upsampled data vector against its new timestamps is also possible. On the other hand, downsampling the data will *erase* intermittent data points to match the sampling rate of the HL IMU. This may lead to the loss of valuable information and is therefore is not undertaken.

taking the $\arg\max$ of the cross-correlation function results in the *index* of the peak of that function. This index gives us the number of samples corresponding to the temporal difference between the two signals. By multiplying this value with the sampling rate of the Xsens, temporal offset in seconds is obtained.

There is a possibility to improve this algorithm however. In the reference [**?**] the discrete-time cross-correlation function is turned into a *cubic spline* and the same approach to finding the index of $\arg\max$ is executed once more. However, this time not the whole signal is searched but rather an interval around the index from the first cross-correlation function is constructed, which spans one sampling period in each direction. In addition, since the constructed cubic spline is a continuous-time smooth function, its derivatives can be analytically calculated. By making the derivative of the spline in the aforementioned interval equal to zero, the peak of the spline can be found. This corresponds to a *refined* iteration of the index, which results in the fine estimation of delay between sensor streams. A modified version of their approach has been implemented in the proposed calibration framework, which takes into account the dissimilarities between the Xsens and HL and the following algorithm shows the pseudocode for the implementation.

Algorithm 4: Spline Correlation

Data: Shifted streams, Correlation Results,Sampling Rates
Result: Finely correlated streams, fine delays in seconds

begin
  for $x, y, z \in XCorr\_functions$ **do**
    $create\, cubicSpline\, from\, XCorr\_function$
    $find\, derivatives\, of\, spline$
    $solve\, for\, deriv = 0$
    $find\, index\, of\ \arg\max(deriv = 0)$
    if $index > 0$ **then**
      Peak is on right of Xcorr-index
      $Append\, zeros * index\, to\, shiftedstream\, on\, end$
      $Delete\, same\, amount\, from\, other\, side$
      $fineCorrStream = appendAndDeleteshifted$
    end if
    else if $index < 0$ **then**
      Peak is on left of Xcorr-index
      $Append\, zeros * index\, to\, shiftedStream\, on\, beginning$
      $Delete\, same\, amount\, from\, other\, side$
      $fineStream = appendAndDeleteshifted$
    end if
    Refine the delay
    $fineDelayinSecs = index/SamplingRate$
  end for
  return *fineDelayinSecs, fineCorrStream*
end

In total, temporal calibration of sensor streams has been achieved with three layers of various correlation methods, in which the a coarse alignment method handled the problem of sensors publishing their data to different platforms, no networking and clock drifting. Consecutively, a discrete cross-correlation function has been constructed to compare the two sensor streams in order to obtain the delay in seconds between them. Finally, a refined version of the cross-correlation function with cubic splines is executed to estimate the fine delay in seconds.

Note that, the camera of the HoloLens is used throughout all of the experiments. The delay results obtained from cross-correlation are also used in a similar way to erase camera images from the beginning or from the end, depending on whether the HMS or the HL had a longer experiment duration. Also similarly to the algorithms discussed above, the first ROS timestamp is assigned as the first HL camera timestamp.

5 EXPERIMENTS

The main application area of the HMS and the HoloLens is the usage in indoor environments and pedestrian motion scenarios. In order to replicate such scenarios, the wearer must undergo certain motion patterns. Due to inherent IMU imperfections described in previous sections, not all motions will give clear results, if the motion pattern does not sufficiently excite the IMU axes.

Several preliminary experiments have been carried out to observe how the sensors output their data in various pedestrian walking patterns. In total, four walking variations were devised.

- Regular walking while looking straight ahead.
- Regular walking with abrupt and sharp stops, looking straight ahead.
- Regular walking while the head is also looking left and right (simulating a pedestrian crossing a light).
- Standing still and doing head movements in various directions.

At the end of the experiments, it was decided that head movements were the most successful, in that they produced *distinct* peaks in the data stream, which could be identified and matched with the actual movement happening in real life. On the other hand, movement patterns which did not have significant head motion produced no peaks in data and due to this the movement happening in real life was not discernible. These experiments were in the early stages of the thesis without detailed knowledge about gyroscope and accelerometer behavior and upon further research regarding other state-of-the-art calibration frameworks, especially [**?**], it was found out that sufficient excitation on all axes of the IMU is a mandatory step in the cross-correlation of angular velocity data. Consequently, head movement patterns were selected as a viable candidate for the calibration framework.

Note that, the accuracy of the sensors play a significant role in this decision. If the sensors were delivering accurate information, even the smallest motion patterns could be perceived, due to a total lack of noisy data. Since this is not the case, a compromise has to be made and the motion patterns related to this specific sensor set-up must be limited to head movements only. Moreover, since the framework [**?**] uses a calibration board, the experiment was done relatively close to the board in which a maximum displacement of about 1.5 meters was reached. Since the verification step requires that the calibration framework be compared with a state-of-the-art gold standard, this was also another reason in the selection of head movements for the primary motion pattern.

5.1 Human Experiments

In this section, the results from the human experiments will be discussed. One experiment is done with free movements with no calibration board, and the other is done with the board for the Kalibr framework with somewhat of a limitation on movement due to the fact that the camera must keep the board in line of sight.

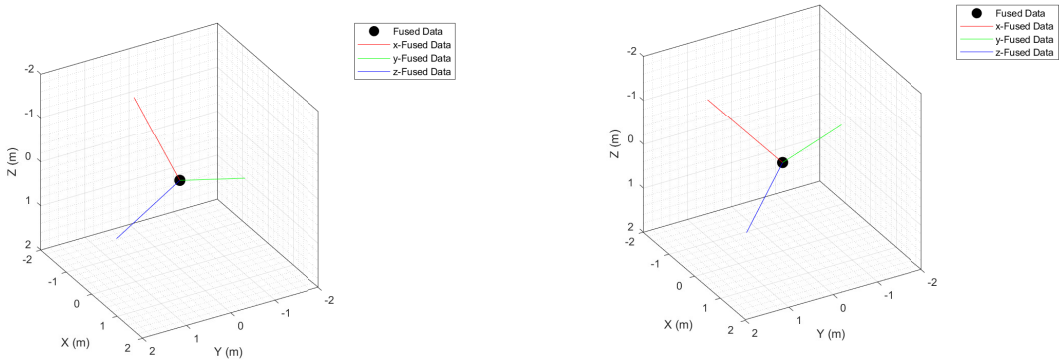
(a)

(b)

(c)

Figure 5.1: Various head movements for axis excitation during the free movement experiment.

In the preceding image, a subject is carrying out the free movement experiment in which it is aimed to produce as many distinct peaks in angular velocity readings as possible. The subject enters the command `rosbag record -a` and simultaneously, he clicks the `Start` button inside the holographic app. After he gets into position and waits for a few seconds so that a stable orientation with respect to the world frame can be established (see complementary filter in 4.2.2) and following this waiting interval, regular movements can be started. In the following image, orientation of the sensors are visualized by using the extrinsic rotation matrix obtained within the stable interval of the experiment. These matrices are plotted with the `plotOrientation` command in the MATLAB environment. It is clear that the sensors are not aligned with each other and any angular velocity readings would lead to suboptimal results if cross-correlation were to be undertaken.

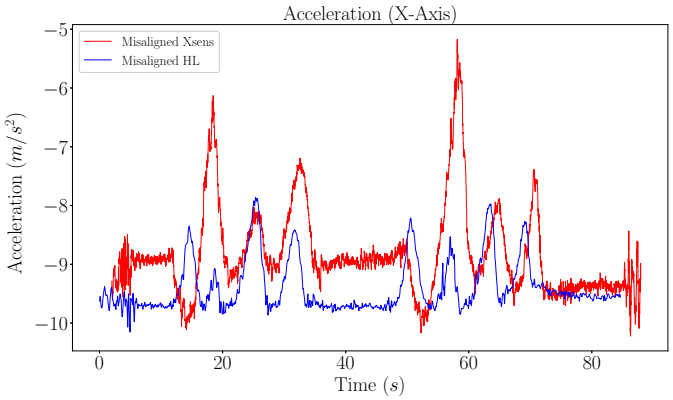


(a) Xsens orientation with respect to world frame.

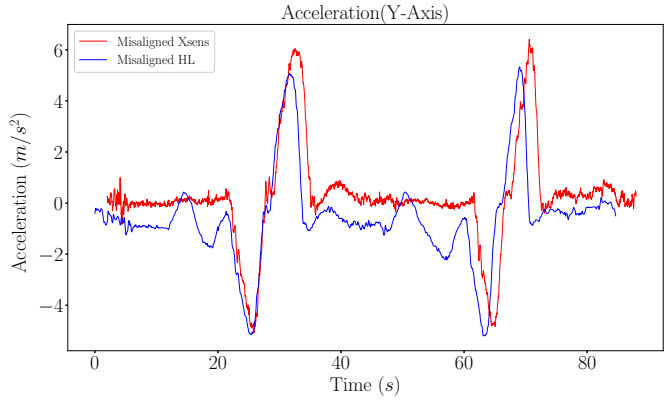
(b) HoloLens orientation with respect to world frame.

Figure 5.2: Sensor orientations with respect to Earth's gravity.

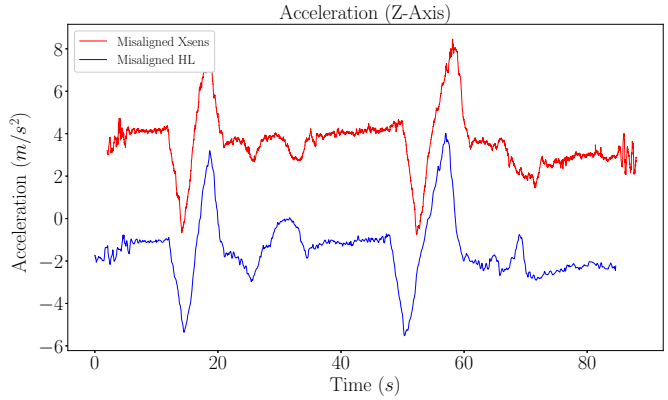
In the next image, raw acceleration and gyroscope values will be illustrated to show that, the axis misalignment is pronounced on compound motion patterns, where multiple axes are excited simultaneously. It will also later be shown that, by applying the method of stable region extrinsic rotation alignment, the axes will be aligned.



(a) Acceleration in x-axis.

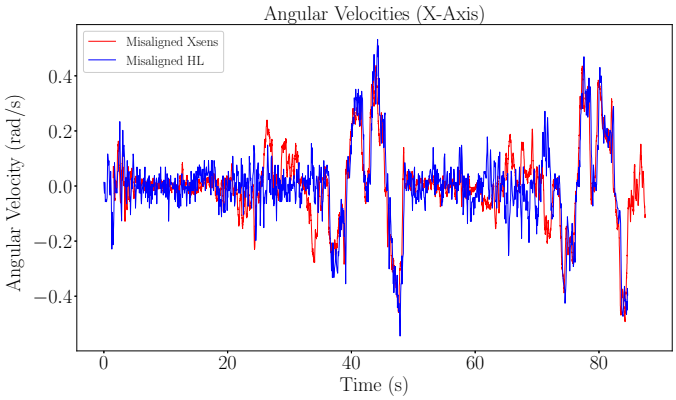


(b) Acceleration in y-axis.

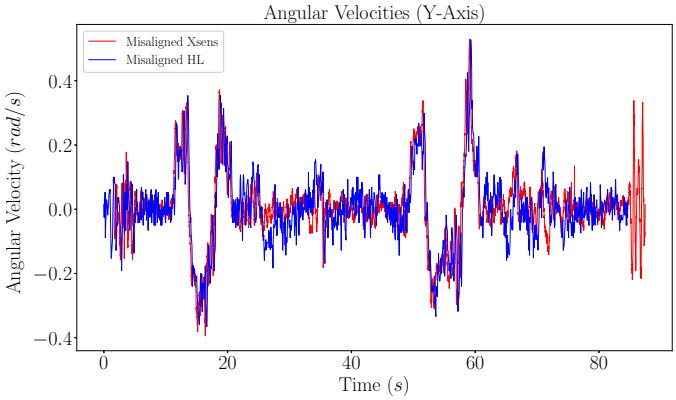


(c) Acceleration in z-axis.

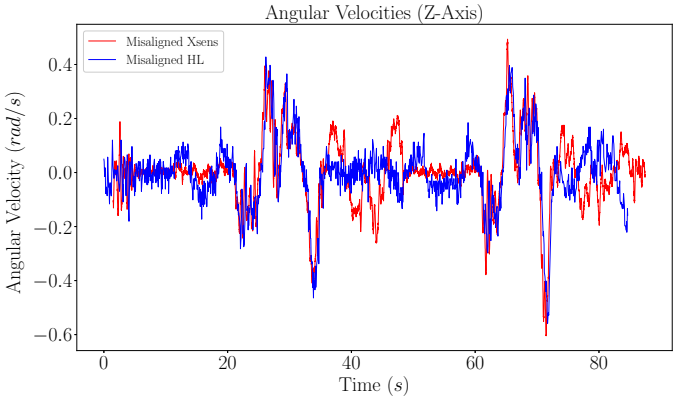
Figure 5.3: Acceleration of all axes without extrinsic rotation alignment.



(a) Angular velocity in x-axis.



(b) Angular velocity in y-axis.

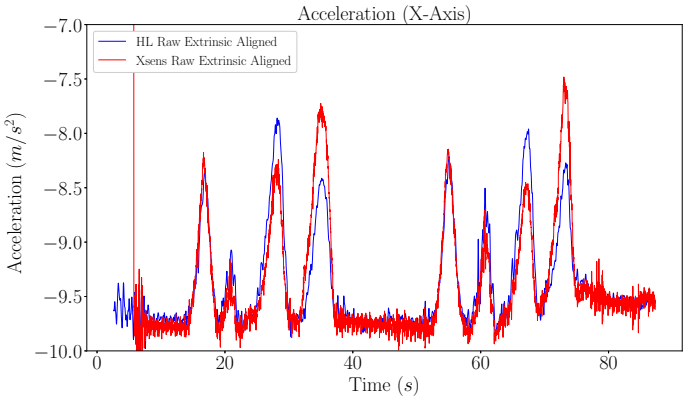


(c) Angular velocity in z-axis.

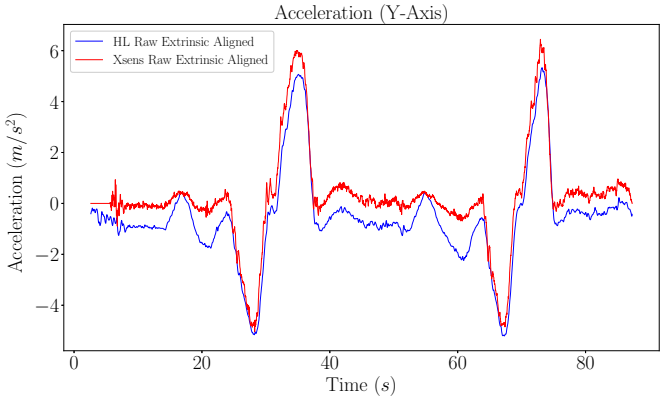
Figure 5.4: Angular velocity of all axes without extrinsic rotation alignment.

Regarding the figures 5.3 and 5.4, especially, in the span between the 20^{th} and the 60^{th} seconds,. clear misalignments regarding the gyroscope readings are observable. For the acceleration values, this problem is even more pronounced.

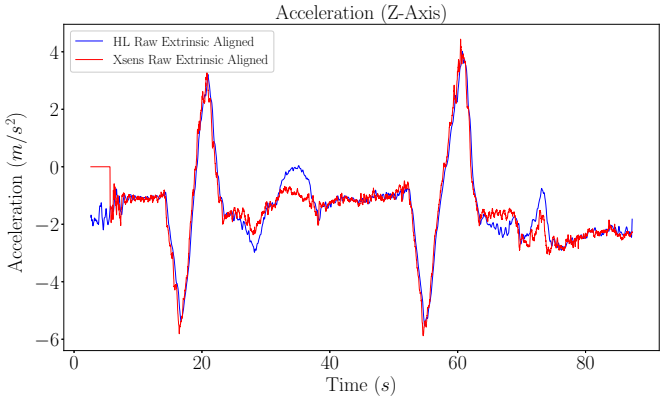
In the following figures, the misalignment is corrected with the extrinsic rotation matrix. Note that, for the sake of illustration, the previous figures have been manually aligned in post-processing to show what the results would look like, had the sensors been temporally calibrated. In the following, temporal calibration is automatically done and therefore, the plot legends indicate this fact.



(a) Acceleration in x-axis.

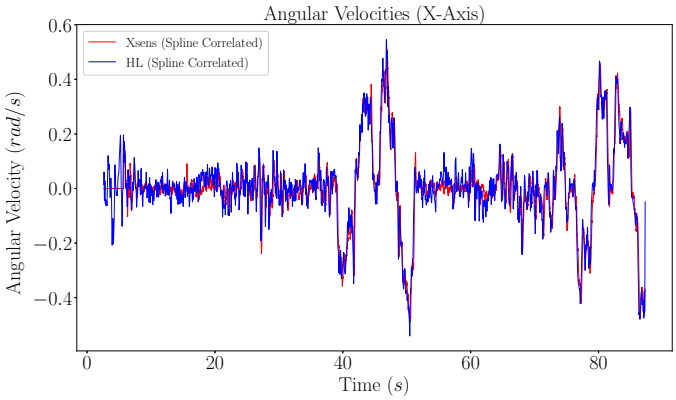


(b) Acceleration in y-axis.

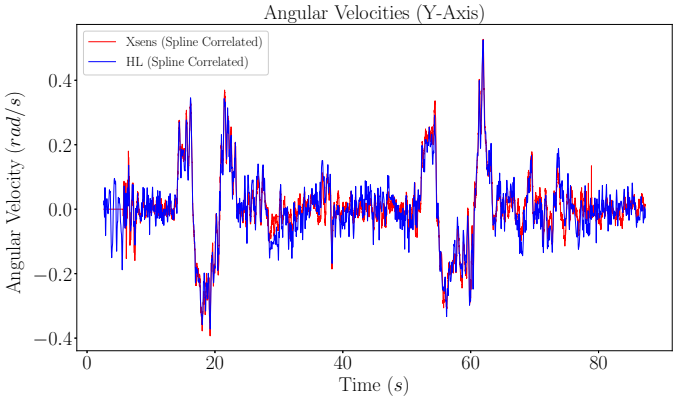


(c) Acceleration in z-axis.

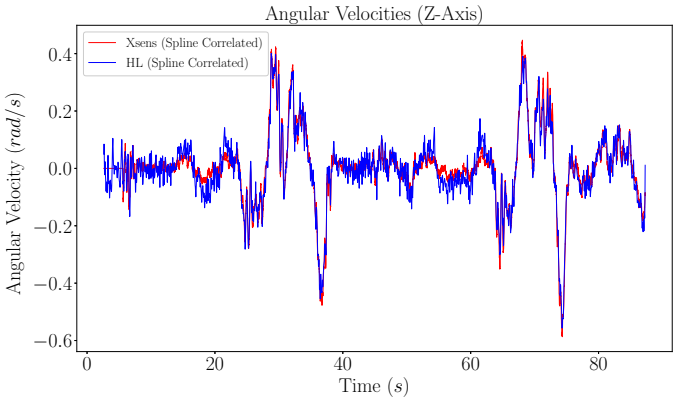
Figure 5.5: Acceleration of all axes with extrinsic rotation alignment.



(a) Angular velocity in x-axis.



(b) Angular velocity in y-axis.



(c) Angular velocity in z-axis.

Figure 5.6: Angular velocity of all axes with extrinsic rotation alignment.

A clear improvement on all axes is present. Note that although the gyroscope values are nearly

identical, the same cannot be claimed for acceleration readings. Especially in ares where one axis is undergoing extreme motion, which is shown by significant peaks, other axes are affected by this movement, indicating to the suspicion of internal axis misalignment or axis sensitivity problems. These will be further discussed in section 6.
In addition to this experiment, another experiment with a calibration board for the Kalibr framework has been carried out. Note that, since the human experiment has no way of verification with a ground truth, the comparison with the Kalibr framework was chosen as a way to verify both the extrinsic rotation matrix and also the temporal calibration. Moreover, as discussed in section **??**, in the absence of ground truth which is usually the case for consumer-grade applications, Kalibr is treated as a ground truth due to its high performance. Since a subject cannot do the same movement with perfect precision, some movement differences are bound to happen. In addition, due to the motion blur induced while looking left and right, yaw excitation around the z-axis is not as pronounced as the other axes.

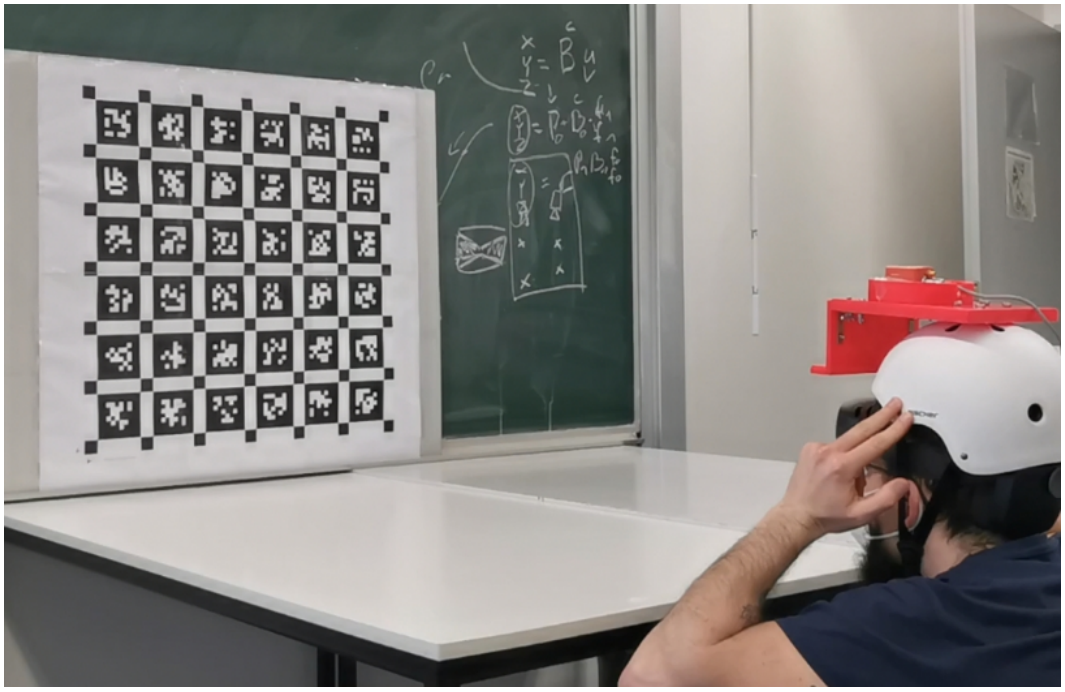
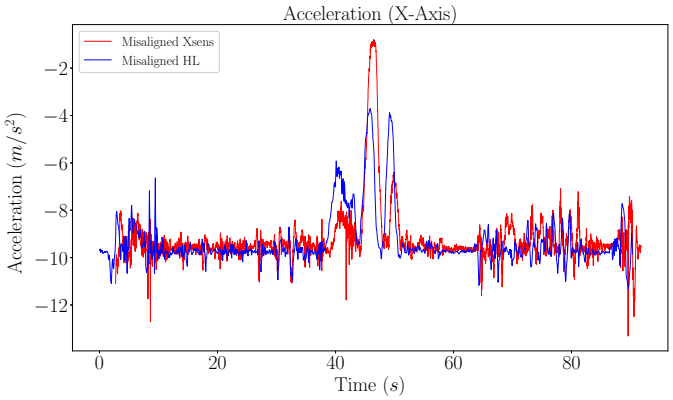
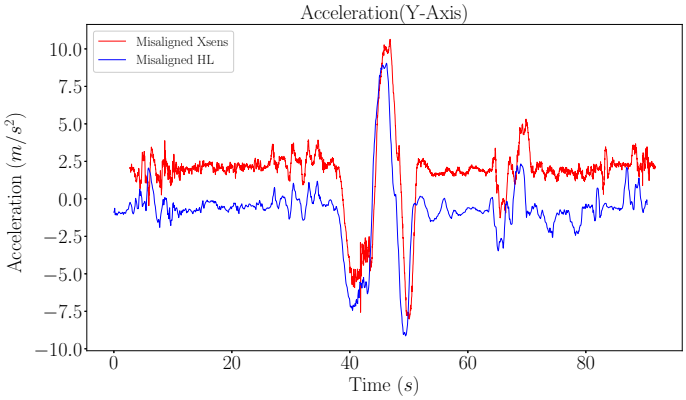


Figure 5.7: A snapshot of the experiment with a calibration board.

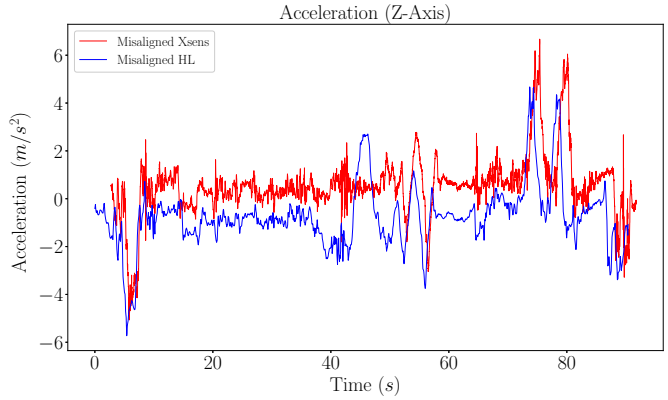
In the next figures, the same acceleration and angular velocity values are illustrated for the calibration board experiment.



(a) Acceleration in x-axis.

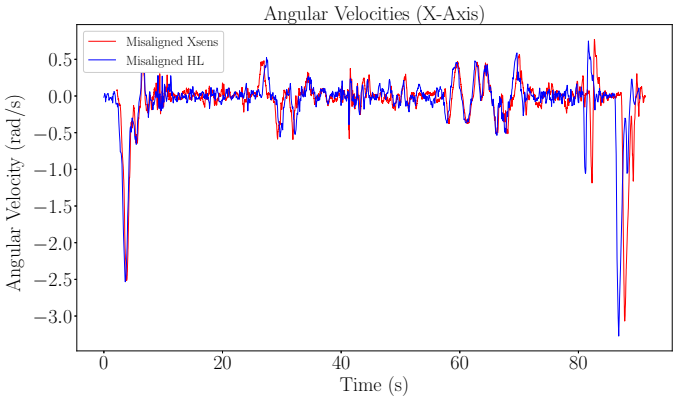


(b) Acceleration in y-axis.

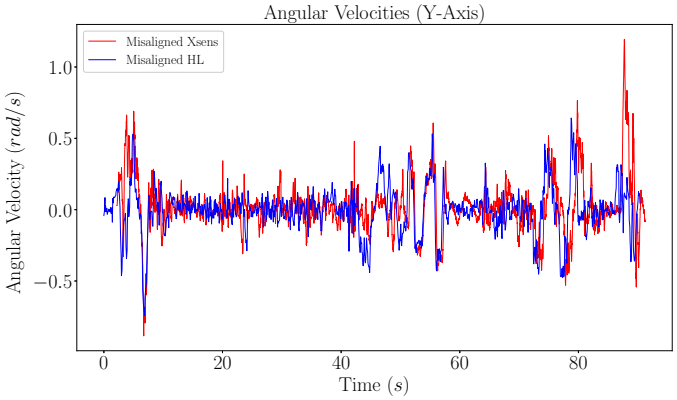


(c) Acceleration in z-axis.

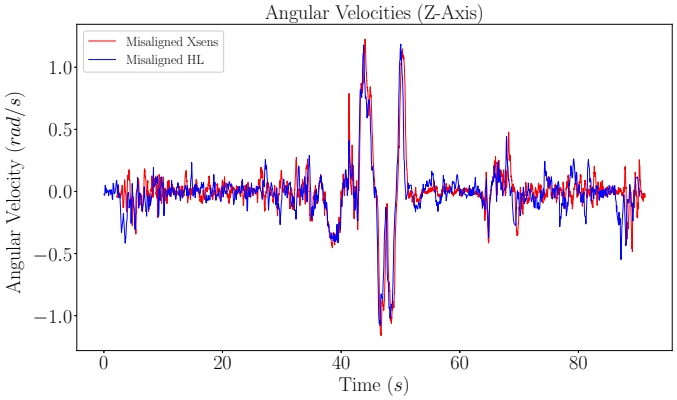
Figure 5.8: Acceleration of all axes without extrinsic rotation alignment.



(a) Angular velocity in x-axis.

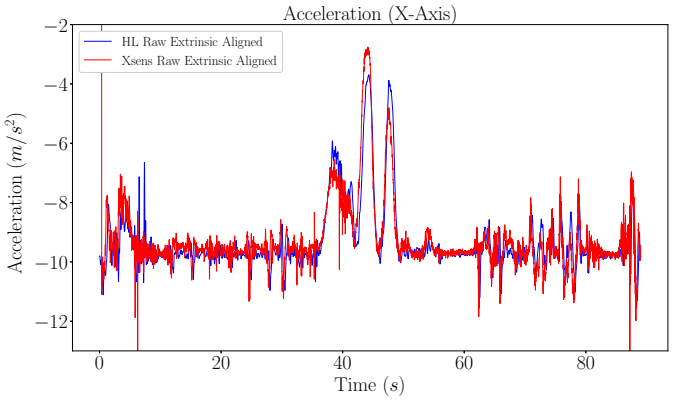


(b) Angular velocity in y-axis.

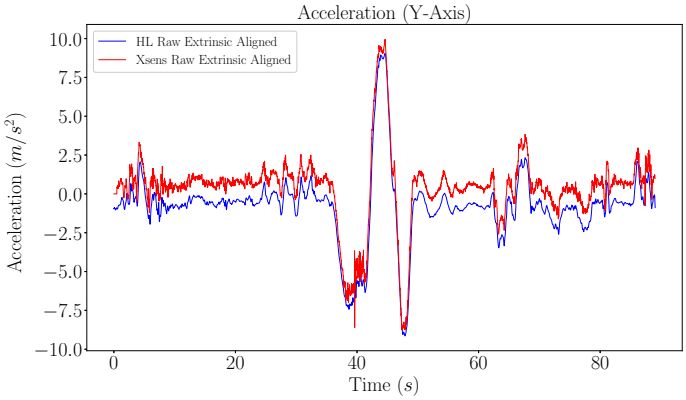


(c) Angular velocity in z-axis.

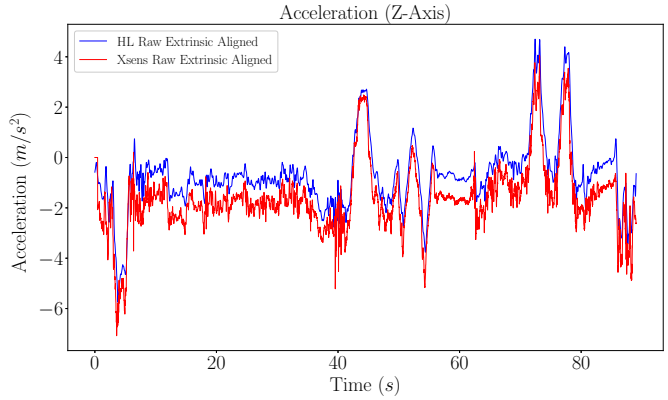
Figure 5.9: Angular velocity of all axes without extrinsic rotation alignment.



(a) Acceleration in x-axis.

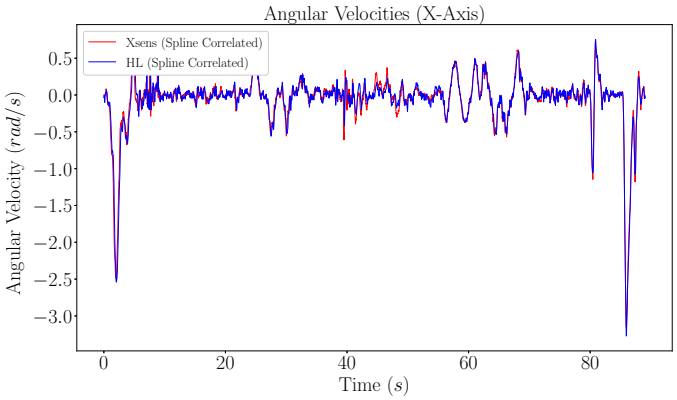


(b) Acceleration in y-axis.

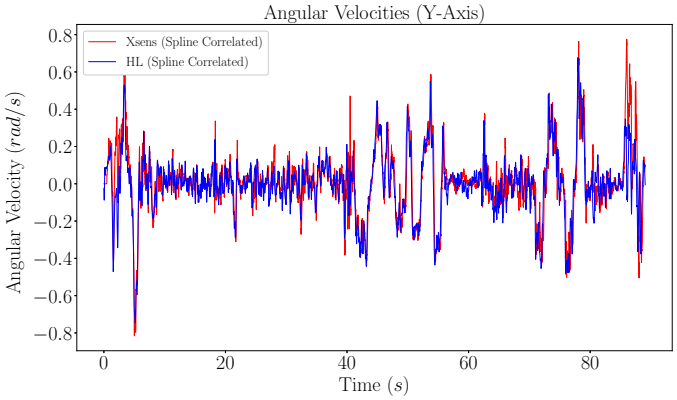


(c) Acceleration in z-axis.

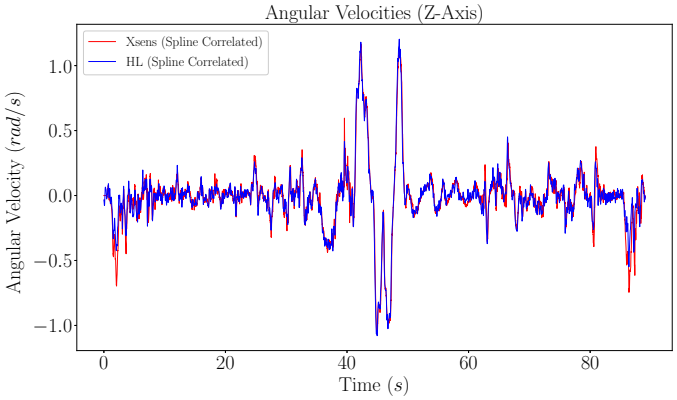
Figure 5.10: Acceleration of all axes with extrinsic rotation alignment (w. board).



(a) Angular velocity in x-axis.



(b) Angular velocity in y-axis.



(c) Angular velocity in z-axis.

Figure 5.11: Angular velocity of all axes with extrinsic rotation alignment (w. board).

For the verification of the results with respect to the Kalibr framework, the cross-correlated and extrinsic rotation aligned IMU results have been converted into bags with Epoch timestamps, corrected with proper Kalibr syntax and then transformed into a merged bag along with the corrected HL images. Since the Kalibr framework supplies the user with a result report when its optimization problem is solved, the extrinsic rotation matrix and the temporal calibration can be compared with the proposed framework. If the sensors are properly aligned, the Kalibr framework should give an identity matrix for the extrinsic rotation matrix and the temporal calibration must match the result that was previously achieved with the three-layer correlation scheme.

	Rotation Matrix	**Temporal Alignment**
Proposed	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	x-axis: 0.397760 seconds y-axis: 0.409813 seconds z-axis: 0.409813 seconds average: 0.405795 seconds
Kalibr	$\begin{pmatrix} 0.998141 & 0.058724 & -0.0162681 \\ -0.058700 & 0.998273 & 0.002000 \\ 0.016357 & -0.001041 & 0.999865 \end{pmatrix}$ $\mathrm{EUL}^{\circ}_{(\mathrm{zyx})} = [3.363, 0.9396, 0.0572]^{\mathrm{T}}$ $\mathrm{EUL}^{rad}_{(\mathrm{zyx})} = [-0.0587, -0.0164, -0.001]^{\mathrm{T}}$	norm: 0.412186 seconds

Table 5.1: Comparison with the Kalibr Framework.

The extrinsic rotation matrix and the temporal alignment of individual axes are highly similar to the Kalibr results, indicating that the proposed method does find the correct alignment and the temporal offset in between two data streams. Further discussion will take place in section 6

5.2 Rigid Body Experiment

In addition to the experiments mentioned in the previous section, a rigid body experiment with a mannequin head and a 6 DOF hexapod has been carried out. The following image shows the sensor set-up on top of the hexapod.

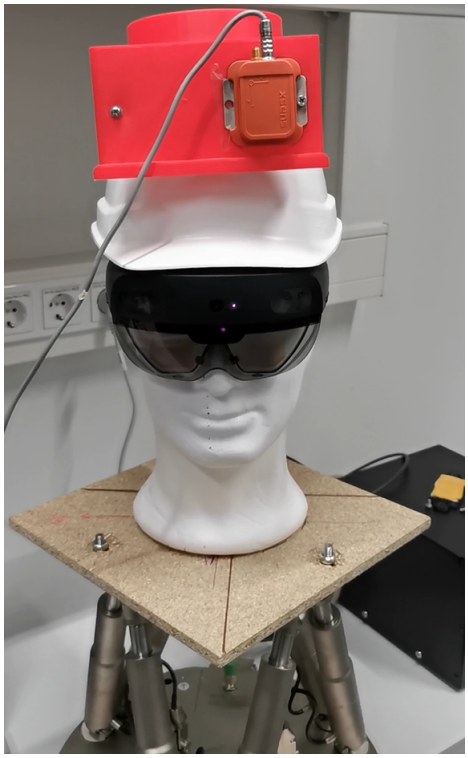


Figure 5.12: Rigid Body Experiment Set-Up.

The hexapod is able to carry out translational and rotational motion, albeit at a limited capacity, amounting to a maximum of 3 centimeters in x-axis and y-axis and a maximum of 2 centimeters in z-axis, in both directions. Therefore, the imitation of walking around the environment is severely limited, if not completely eliminated. On the other hand, the rotational motion around the axes is somewhat better than the translation, totaling to a maximum of 8 degrees in roll and pitch axes and a maximum of 6 degrees in yaw axis. However, the same capacity is not found when the hexapod is going in the opposite direction which makes the roll, pitch and yaw axes be able to rotate at a maximum of 3.5, 4 and 9 degrees. These are severe limitations, but fortunately under these circumstances, motion patterns could still be observed.

A trajectory was planned, in which the hexapod would go to the maximum capacity in all directions in a linear fashion, returning to origin each time it completes the trajectory. First, translational motion in all axes must take place and when that finishes, rotational motion about all axes will take place. The trajectory of the robot is given by providing waypoints that correspond to timestamps in a python script, which communicates with the robot on a low level. The waypoints are then transformed into viable motor commands so that the actuators at the base of the hexapod can move the platform in the direction that the waypoints intend.

Even though the maximum motion range of the hexapod is advertised as 5 centimeters in all directions and 15 degrees roll,pitch and 30 degrees for yaw axes, none of these values could be achieved with the experiments. It is stated in the hexapod's manual that compound motion could limit the maximum movement capacity, however even though all axes were *individually* excited and the hexapod was commanded to back to origin, such values proved to be unreachable.

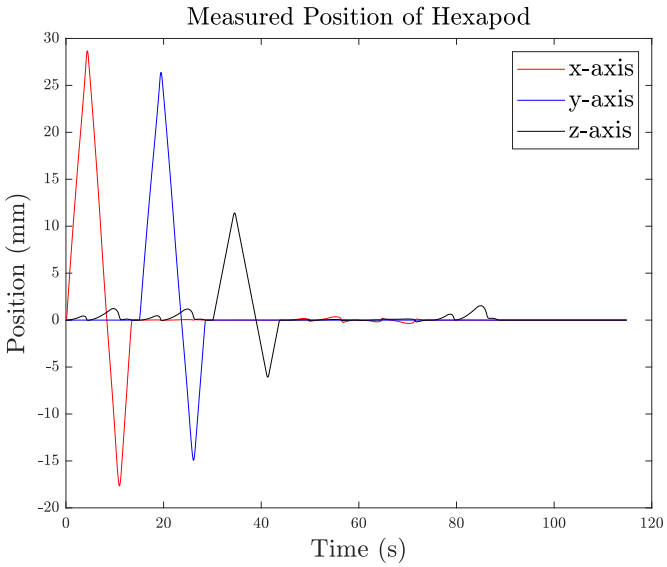


Figure 5.13: Hexapod Position.

In the preceding image, the movement of the hexapod is plotted against time, as discussed before, although the same command is given in both directions, the maximum distance cannot be reached, hence the unsymmetric behavior on both sides of the origin point. Translational motion could not be picked up by the IMUs due to them being too slow and too small. Therefore, this interval of the experiment is manually discarded and only the rotational motion interval is analyzed. Similarly, in the next image the attitude angles are illustrated.

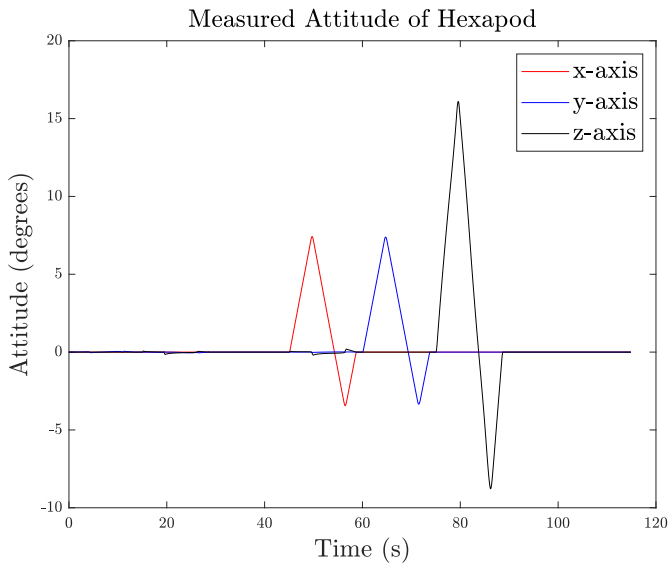


Figure 5.14: Hexapod Attitude.

In addition, the following image shows the body frame angular velocity values similar to what a gyroscope would have measured. This was also helpful in determining if the sensors were following the ground truth or not.

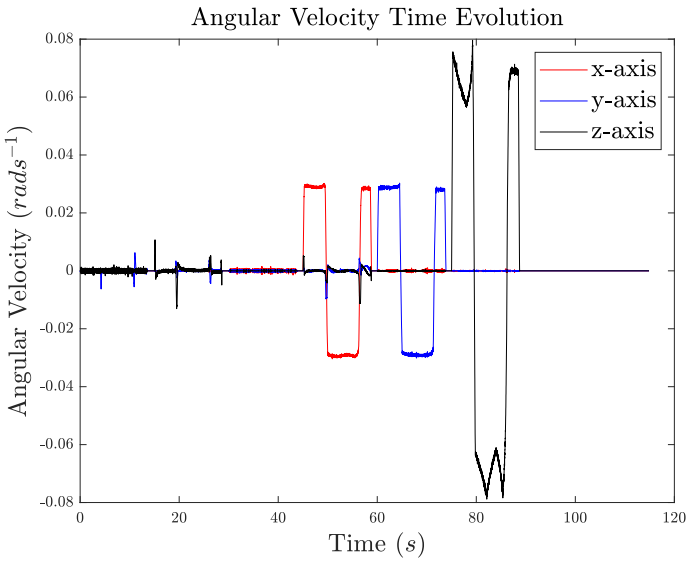
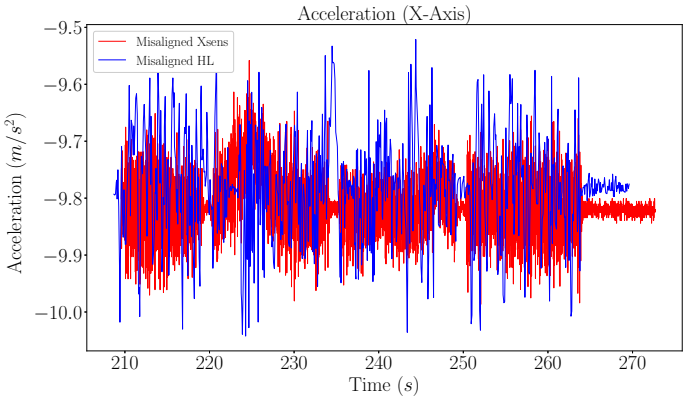
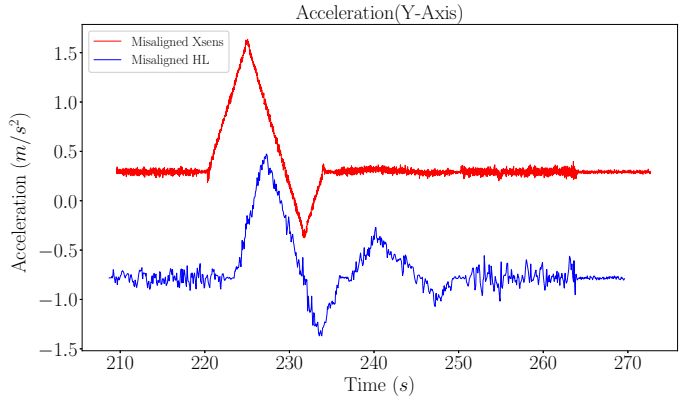


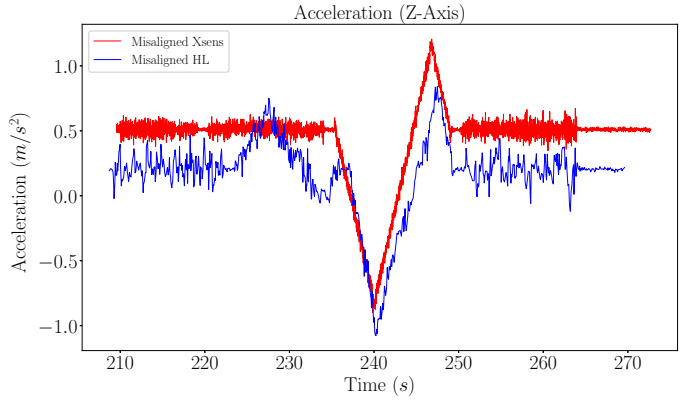
Figure 5.15: Hexapod Angular Velocity Values.



(a) Acceleration in x-axis.

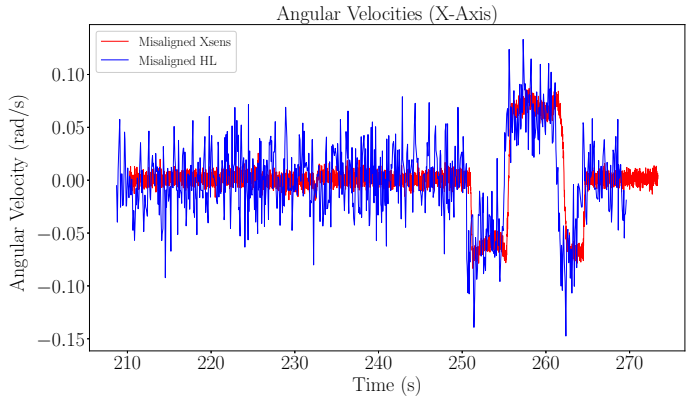


(b) Acceleration in y-axis.

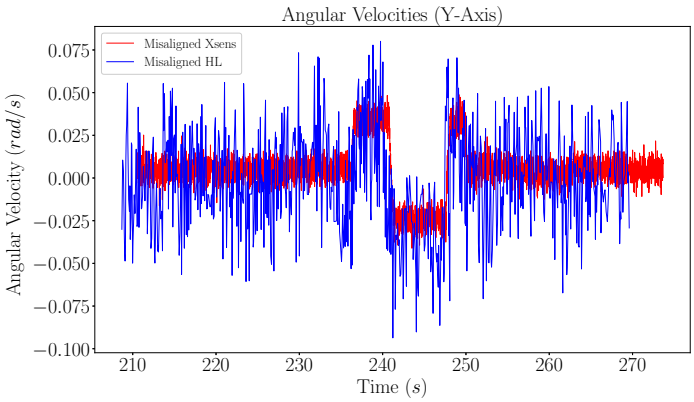


(c) Acceleration in z-axis.

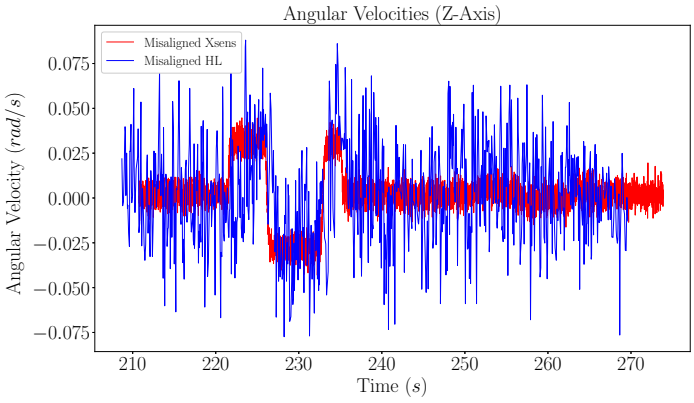
Figure 5.16: Hexapod acceleration of all axes without extrinsic rotation alignment.



(a) Angular velocity in x-axis.



(b) Angular velocity in y-axis.

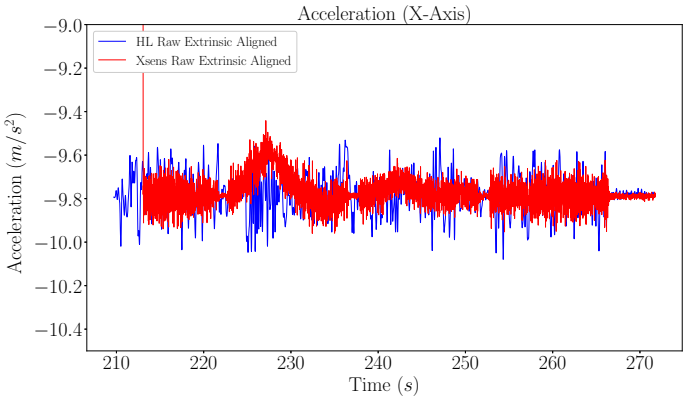


(c) Angular velocity in z-axis.

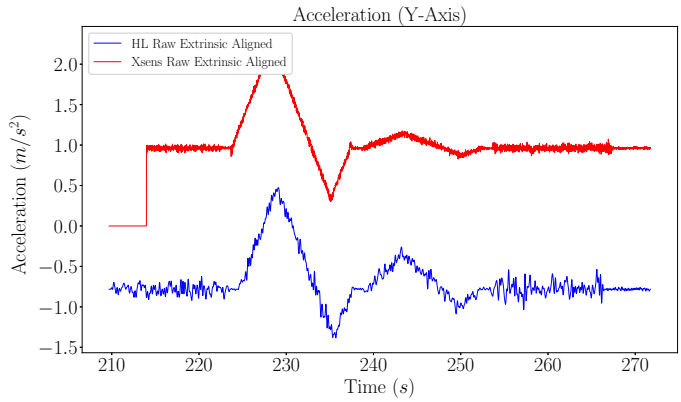
Figure 5.17: Hexapod angular velocity of all axes without extrinsic rotation alignment.

Note that in these figures, the timeline does not start from zero because the rigid body experiment set-up needed to be manually started and while the trajectory was loading, the HL and the Xsens had already been running. This is merely a cosmetic fact, not affecting the cross-correlation in any way, because the irrelevant parts of the data streams have been manually erased.

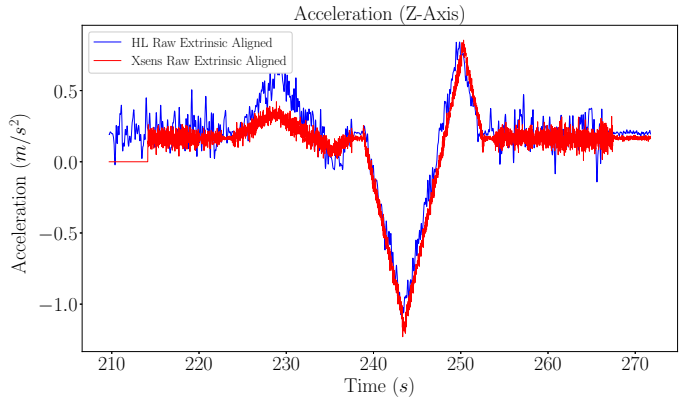
The difference between an industrial-grade and a consumer-grade is clearly demonstrated in the preceding images. Because the human experiments had high excitations, plotting the overall data stream does not show the noisy behavior and the data stream seems rather like a straight line. In the case of the rigid body experiment with minimal motion, the HL gyroscope values are barely readable, whereas Xsens performs really efficiently. Suprisingly, the accelerometer readings are similar to each other with enough excitation.



(a) Acceleration in x-axis.

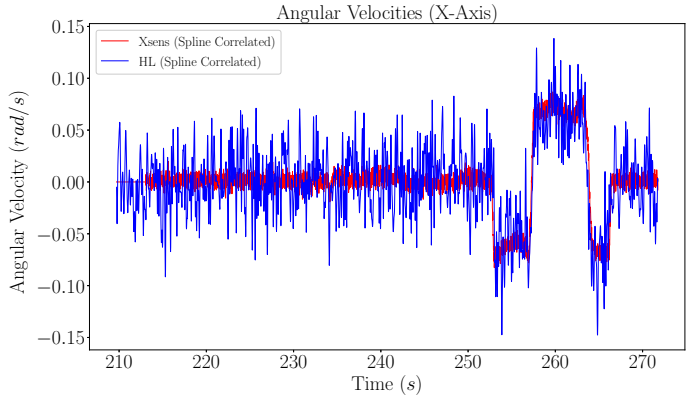


(b) Acceleration in y-axis.

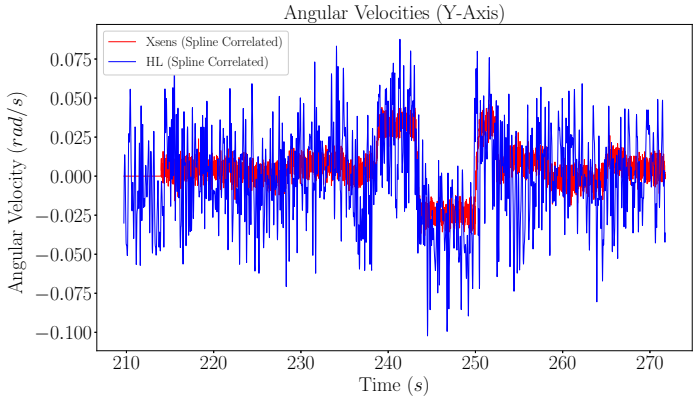


(c) Acceleration in z-axis.

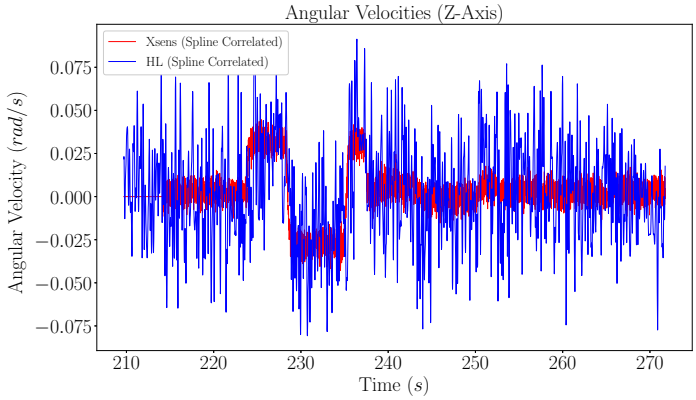
Figure 5.18: Hexapod acceleration of all axes with extrinsic rotation alignment.



(a) Angular velocity in x-axis.



(b) Angular velocity in y-axis.



(c) Angular velocity in z-axis.

Figure 5.19: Hexapod angular velocity of all axes with extrinsic rotation alignment.

Similar to human experiments, a table discussing the extrinsic rotation matrix and temporal offset results is illustrated below.

	Rotation Matrix	**Temporal Alignment**
Proposed	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	x-axis: 3.350865 seconds y-axis: 4.278982 seconds z-axis: 4.435677 seconds average: 4.024436 seconds
Kalibr	$\begin{pmatrix} -0.950526 & 0.29522118 & -0.096659 \\ -0.267507 & -0.61972534 & 0.737821 \\ 0.157918 & 0.72717602 & 0.668039 \end{pmatrix}$ $\mathrm{EUL}^{\circ}_{(\mathrm{zyx})} = [164.2819, 9.0869, 47.4288]^{\mathrm{T}}$ $\mathrm{EUL}^{rad}_{(\mathrm{zyx})} = [-2.8673, -0.1586, 0.8278]^{\mathrm{T}}$	norm: 4.774644 seconds

Table 5.2: Hexapod comparison with the Kalibr Framework.

Note that the extrinsic matrix obtained from the Kalibr framework is nowhere near an identity matrix. Due to the lack of excitations coupled with noisy gyroscope readings, Kalibr's optimization problem cannot fit a correct spline over the discretely sampled IMU streams (which was also verified with Kalibr's internal plot functions which visualize the splines). Here, we see the detrimental effect of taking the gyroscope norms for the temporal calibration instead of individually aligning the axes. The temporal offset obtained from Kalibr is not good enough and this too contributes to the incorrect calculation of the extrinsic rotation matrix.

6 CONCLUSION

In this section, the results of the experiments will be analyzed and remarks regarding the spatiotemporal calibration of heterogeneous systems will be shared. Although more experiments have been carried out regarding human motion and rigid body motion, only three experiments have been shown in this work so that the key concepts could be shared with the reader. Further experiments with raw data can be found under the following link.[1]

6.1 Discussion on Experiments

Rigid Body Experiments

Note that the axes of hexapod and the IMUs are not the same. This is merely a naming convention, and the corresponding axes can be listed below.

- z-axis of the hexapod corresponds to x-axis of the IMUs.
- y-axis of the hexapod corresponds to y-axis of the IMUs.
- x-axis of the hexapod corresponds to z-axis of the IMUs.

Even though the comparison with a state of the art framework suffices to verify the accuracy and the integrity of the proposed framework, rigid body experiments were carried out to further analyze the behavior of the HoloLens under known input. As discussed in 5.2, the hexapod does not exhibit a flexible motion range and this had a detrimental effect on both observing the raw readings of the sensors and the benchmarking procedure with the Kalibr framework.

The temporal calibration of sensor streams relies solely on the gyroscope readings and regarding the figure 5.17, achieving this is relatively straightforward, even under noisy circumstances. In order to determine the extrinsic rotation between the sensors, quaternions are employed within the Kalibr framework, which aid in the estimation of orientation with respect to Earth's gravity as well. The inaccurate results regarding the table **??** originate from this orientation estimation, because as the figure 5.16 shows, even though the stable region approach with the complementary filter is able to align the z-axis quite well but the y-axis shows significant discrepancy between the sensor readings. The x-axis is irrelevant because it signifies the direction

[1] https://github.com/ozgunkaratas/HoloLens2ForCV/tree/main/Samples/StreamRecorder

toward gravity and due to insufficient excitation in that direction, the acceleration values are not meaningful. The difference between the raw readings is attributed to the inherent *scaling factors* and *axis misalignment* which were discussed in section 3.3.3. The Kalibr framework assumes that the sensors are scaled equally, therefore it solves the optimization problem, assuming that the acceleration values experienced in all axes are scaled correctly. Coupled with the insufficient excitation on axes, the Kalibr framework did not reach a satisfactory extrinsic rotation matrix, although the complementary filter within the proposed framework was able to estimate the extrinsic rotation matrix more successfully. This claim is made because if the figures **??** and 5.16 were to be compared with each other, even under insufficient axes excitement, the extrinsic rotation matrix aligns the accelerometer readings on the y-axis and the z-axis (see between 220^{th} and 235^{th} and between 240^{th} and 255^{th} second for the alignment of small bumps).

This type of error can be remedied if the candidate is extensively tested with known ground truths (with sufficient movement) with a reference IMU. This would give the necessary parameters to set up the equation 3.32. Preferably a rotating table of a robotic arm with 6 degrees of freedom can achieve such calibration, which were not available during the experiment sessions.

Human Experiments

In contrast to the rigid body experiments, the human experiments have given clear results, indicating good extrinsic rotation alignment obtained from the proposed framework and also from the Kalibr framework. This is not only backed up by the figures shared in **??**,**??**,**??**,**??** but also from the results obtained from **??**. In addition, the temporal alignment of the proposed framework is even better than the cross-correlation of the norm of all gyroscope axes, which is employed inside the calibration framework, due to preserving individual axis information.

Similar to what was experienced in the rigid body experiments, the scale error and misalignment error between the IMUs are present in all axes, see figure **??** and **??**. Especially in the intervals between the 25^{th} and the 40^{th}, as well as between 50^{th} and 70^{th} second, due to compound motion on all axes, it is clearly observed that the z-axis and y-axis of the HL accelerometer are affected from each other. This proves the suspicion that the internal axes of the HL are misaligned to a certain degree.

The claim that the extrinsic rotation matrix correctly calibrates both sensors is solidified with experiments that contain compound motion, because the shape of both data streams, be it angular velocity or acceleration, are nearly identical and only differ in magnitude, leading to the suspicion that this is caused by internal errors, which are *unquantifiable* without sufficiently excited ground truth comparison.

In addition to these factors, the internal axes of the HL gyroscope and HL accelerometer are also not perfectly aligned, whereas the Xsens IMU's alignment matrix is a near perfect identity matrix. This also contributes to the errors discussed previously, with no way of knowing which error is more prominent among all, thereby rendering the perfect calibration of the sensors as of current standards, unlikely.

There is another factor contributing to the discrepancy between the accelerometer readings, it is the so-called lever-arm effect. Since points on a rigid body will *not* undergo same acceleration due to reasons discussed in subsection 3.3.2, the sensed accelerations on the Xsens body frame and the HL body frame will differ from each other. This is addressed in the following section.

6.2 Verification of Distance Between Sensors

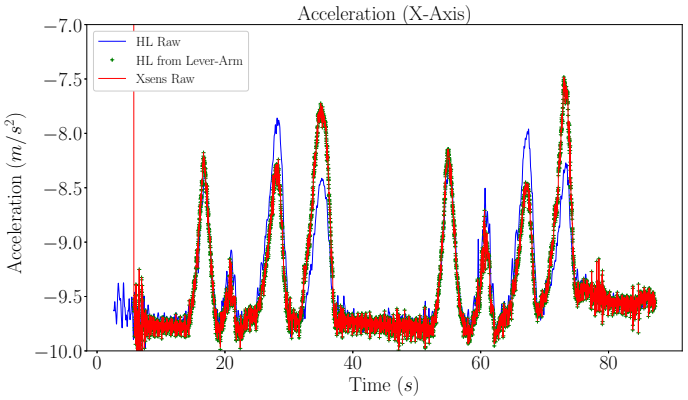
In order to fully calibrate the sensors in three-dimensional space, the distance between them must also be estimated, along with their orientation and temporal offset. Measuring the distance is possible by analyzing CAD models or by hand with a measuring tape. In consumer-grade applications, the usage of CAD models is virtually non-existent, because such models are either never created due to practical or economic reasons. Measuring the distance by hand is the common option, which was also employed in this work as well.

Moreover, as it was discussed previously, each time the HMS and the HL are worn, the distance between the sensors change. This change is not extreme for the same person (meaning that the change can be neglected and assumed constant), but the same claim is violated for different head sizes. Thus, other methods to find this distance must be carried out.

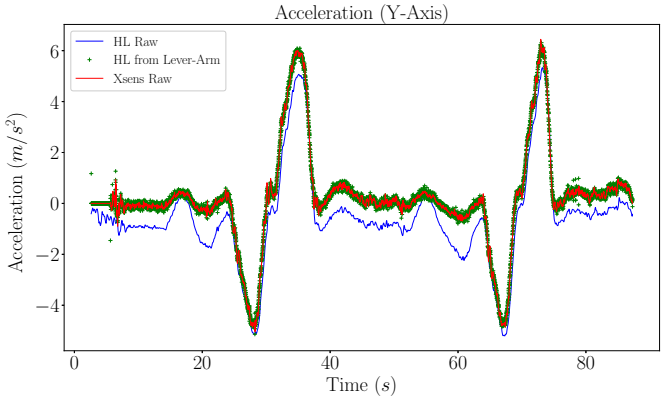
The equation 3.31 serves as a way to visualize the distance between two points on a rigid body, which under perfect circumstances, should give the same acceleration for both of the sensors. This is not possible, due to reasons discussed in the same section. It may be a viable option to solve this equation within an optimization problem, in which the distance between the sensors is approximated, which is the same method employed inside the Kalibr framework as well. However, solving this optimization problem is not handled by using the data from two IMUs but rather from a reference IMU with high accuracy and a camera, from which ego motion is extracted. This suggests that calculation of the distance between the sensors, i.e the lever-arm distance, is unlikely with sensors that have noisy data output, which is the case for the HL IMU. Coupled with the fact that scale factors are present which lead to significant discrepancies as discussed previously, the task is even more challenging, if not impossible.

The HL camera is also subject to sampling rate fluctuations and motion blur, which makes the ego motion extraction even more challenging when a third party calibration framework like Kalibr is utilized to analyze its raw data. Even though Kalibr has given a matching result when it used the gyroscope values to find the extrinsic rotation matrix in comparison to the proposed framework, all values diverged when the camera was introduced to the system. The second prediction (after the camera is introduced) falsely calculated the rotation matrix and the distance between the sensors was calculated as 30 centimeters, which is clearly wrong. These are all artifacts originating from the non-uniform and noisy data obtained from the HL camera and the HL IMU.

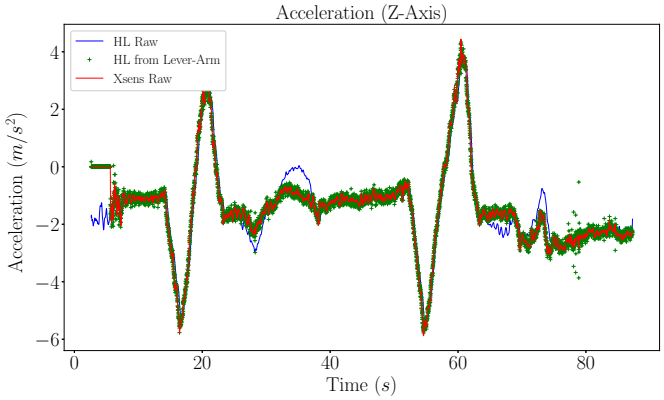
By hand measurements, the distance between the sensor origins was estimated to be about 15 centimeters in height. By plugging this parameter into the equation 3.31, it is theoretically possible to obtain the true acceleration value of a point on a rigid body by using the acceleration values of another point on the same rigid body. Inversely, by knowing the true acceleration of two points and solving for the position vector, the distance between the sensors can be calculated. Such an operation has been carried out and the results are as follows.



(a) Acceleration in x-axis.

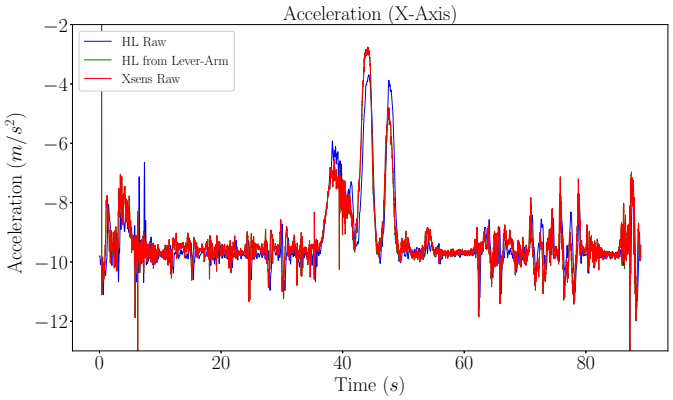


(b) Acceleration in y-axis.

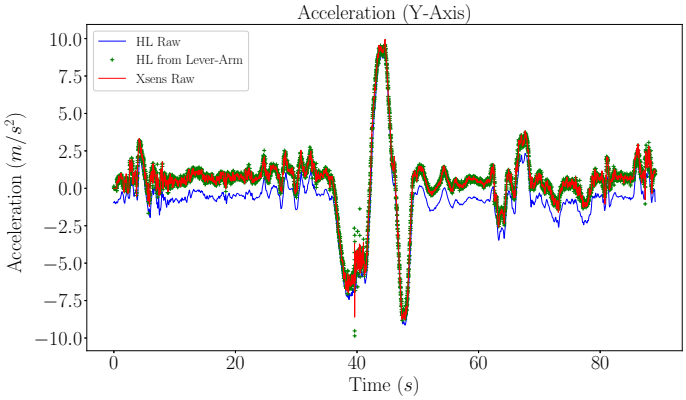


(c) Acceleration in z-axis.

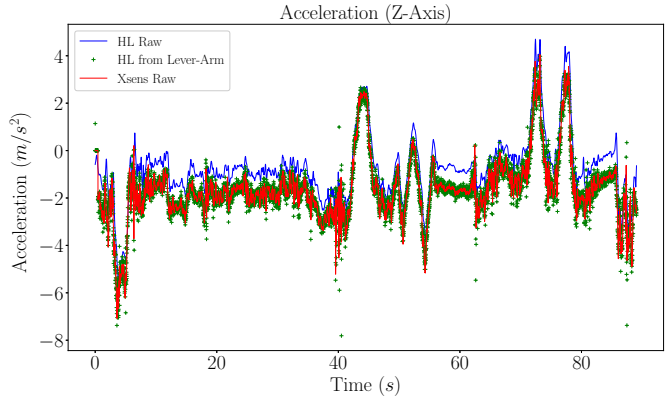
Figure 6.1: Acceleration of all axes with lever-arm (human experiment).



(a) Acceleration in x-axis.



(b) Acceleration in y-axis.



(c) Acceleration in z-axis.

Figure 6.2: Acceleration of all axes with lever-arm (board experiment).

In the preceding images, the acceleration sensed from the Xsens is used to calculate the

acceleration of the origin of HL, while also taking the angular velocity values of Xsens because they are more reliable. The green dots, which are somewhat buried underneath the Xsens stream, represent the HL acceleration calculated with the lever-arm effect accounted for. As it can be seen, the change in value is insignificant, due to the following reasons.

- 15 centimeters are too small to make a discernible difference in accelerometer readings.
- Even if this difference was pronounced, it lies underneath the noise barrier of both sensors.
- The difference is acceleration is clearly due to misalignment and scale errors and not due to the lever arm effect.
- Unquantifiable errors make it impossible to discern the distance between the sensors due to a lack of ground truth.

Due to these limitations, the calibration framework is limited in its ability to discern the distance between the sensors. However, not including the lever-arm is not detrimental to the total accuracy of the calibration since as provided before, the difference is overshadowed by misalignment and scale errors and the rectifying these errors would improve the overall accuracy of the framework.

6.3 Real-time capability of the calibration framework

The main reason why the calibration framework is done in python is because the HoloLens does not publish its data to an external recipient, such as a ROS topic. No low-level communcation protocol exists for the HL thereby rendering the usage of drivers non-optional. With the implementation of a `rosbridge_suite`, which uses a JSON API to communicate with non-ROS applications over the internet, it is theoretically possible to send HL data to an Ubuntu computer. This was ruled out in the early stage of research because it would have introduced unknown delays and also because HL could not connect to the internet at the time of writing the thesis.

It is also possible to capture raw HL data and convert it to `rosbags` and then use this in a quasi-real time state. However, this was also ruled out because of the non-negligible clock drift of the HL and since verification with a state-of-the-art framework was also a task, `rosbags` were created after the proposed framework was done calculating all necessary parameters. The automated process of doing all this work is better realized on a python environment than a ROS node.

Linear interpolation, upsampling, quaternion calculation and Fourier transforms are the computationally demanding algorithms used in this thesis. Such algorithms are suitable for real-time applications if they are used in a windowed-mode, meaning that they have a maximum horizon of samples that they can work on before they discard these samples and move on to the next batch. Due to erratic change in sampling rate and the non-uniform publication of data from the HL IMU, downsampling the Xsens to match with the HL was ruled out due to the streams not matching with each other. A rate balance filter from [**?**] is used to downsample their sensors, but the sensor streams are stable and the distance between timestamps are sufficiently equidistant. Owing to these reasons, upsampling is a must between the Xsens and the HL.

6.4 Regarding generalization of heterogeneous sensors in the unified framework

Bringing dissimilar systems together is not a matter of theoretical prowess but a matter of engineering expertise, since the sensors models are already known but limitations such as sampling rate discrepancies, noisy output etc. pose a challenge for merging these systems together.

An exteroceptive sensor (e.g. a camera) was benchmarked along with the IMUs for its own performance metrics and it was found out that the obtained images were rather unsuitable for ego motion extraction with a calibration board, which indicates that a natural landmark approach can be taken instead. Since the HL camera also publishes a homogeneous matrix (comprised of the rotation matrix described extensively in this thesis and the position vector that points from the origin to the center of the HL camera) this can actually be used to extract a better ego motion pattern. In fact, this homogeneous matrix gives a sufficiently accurate pose estimate because of the internal scene understanding algorithms. Touching upon the main task of the thesis, it is possible to use the HL camera and the Xsens IMU simultaneously, which would lead to an even better spatiotemporal calibration framework. In the scope of this thesis, such fusion was not undertaken, nonetheless, the framework extracts this homogenous matrix, linearly interpolates it, upsamples it and extracts quaternions that are equivalent to the rotation matrices at each point in time.

It was mentioned in the section 2 that the golden standard frameworks are specified to one area and using them to benchmark other sensor suites will always come with challenges. This challenge is especially pronounced in the case of the HL IMU, whose sampling rate fluctuates erratically. Despite all of these challenges, with the algorithms utilized in this work, spatiotemporal calibration was achieved.

6.5 Summary of Contributions

In this section a brief recap of the contributions will be shared. The performance and noise profiling for the IMUs have been carried out by means of the Allan variance test, whereas the intrinsic and extrinsic parameters of the HL camera have been obtained by subjecting the camera to a checkerboard-pattern calibration board test and the sampling behavior of the sensors under various overhead conditions have been observed as well.

After undertaking various walking patterns, suitable pedestrian motion candidates have been obtained which sufficiently excites both accelerometer and gyroscope axes, which is a crucial step in temporal calibration. In addition to human trials, a 6 DOF hexapod has been used to do rigid body experiments with a mannequin head to see how the sensors would perform under known, linear motion inputs.

A complementary filter has been implemented in the calibration framework which calculates the orientation of the sensors with respect to Earth's gravity. From these orientations, the extrinsic rotation matrix between the sensors have been calculated. The selection of this filter over KF variants was due to further real-time applicability possibilites since KF variants were suffering in that area and also because the accuracy of the complementary filter proved to be sufficient.

The proposed framework has been benchmarked with a state-of-the-art golden standard, which employs an optimization problem to solve the extrinsic rotation and temporal calibration between two IMUs. Nearly identical results have been reached, solidifying the claim that the proposed framework is indeed correct in its calculations. The same benchmarking has been done by using the aforementioned hexapod but due to insufficient excitation on gyroscope and accelerometer axes, the results have been inconclusive regarding the verification with the state-of-the-art framework. A robotic arm with a wider range of motion was recommended to carry out the same experiments to solve the excitation problem. Moreover, the experiments have shown an inherent scale and misalignment error, which were clearly present after the extrinsic rotation alignment. The recommended solution to handle this problem is to do rotation table and ground truth experiments.

It was shown that, by utilizing rigid body kinematics equations, the distance between sensors could be theoretically calculated, albeit it was not possible to verify this claim because the scale and misalignment errors were overshadowing the difference in accelerometer readings. The pre-calculated distance was plugged into the rigid body kinematics equation to see how the lever-arm compensated IMU readings would look like and it was observed that the distance proved insignificant changes, indicating that solving this problem would not increase the overall accuracy of the calibration framework.

The calibration framework and the *Stream Recorder App* can be found under the following link and link. The unified data acquisition framework can be constructed by using these software, by which the following operations will be undertaken.

- The raw data will be parsed.
- HL IMU and HL camera will be aligned with each other (internal delays mitigated).
- The dynamically changing sampling rate is estimated for all sensors.
- Linear interpolation is carried out to achieve equidistant sensor readings.
- Coarse alignment of data streams is undertaken to deal with the timestamping issue.
- HL IMU readings are upsampled to that of Xsens' sample count.
- A complementary filter is applied to calculate the extrinsic orientation between the sensors.
- Discrete cross-correlation is applied for temporal offset calculation.
- A finer spline cross-correlation is applied to determine a refined temporal offset.
- The homogeneous matrix obtained from HL camera is parsed with the temporal alignment information.
- The rotation and translation obtained from HL camera is upsampled to match Xsens' sample count.
- Employ rigid body kinematics to find the theoretical true acceleration of two points on a rigid body.
- Plot all results.
- Create a `rosbag` with unified Epoch timestamps containing IMU and camera readings.

In the next section, further improvements regarding the calibration framework and the possibility of extending it with other sensors will be discussed.

7 FUTURE WORKS

The main limiting factor within the span of the experiments was pinpointed to be the internal misalignment and scale factor errors in the HL IMU, which was demonstrated with the accelerometer readings. In order to fully merge the HoloLens with the HMS, these errors must be eliminated by means of extensive rigid body experiments with sufficient excitation to that the 6 DOF behavior of the HL accelerometer can be profiled. This will not only improve the temporal calibration to its maximum capacity, but also it is a mandatory step to merge acceleration readings between sensors. Due to the small distance between the sensors, the lever-arm effects were ruled out to be insignificant. This claim was also backed by the fact that acceleration values were identical in peak intervals, meaning that the streams were indistinguishable from each other. Thus, spending effort on the compensation of the distance between the sensors should be discarded and in extreme cases where high accuracy is a must, the Xsens IMU can augment the HL IMU completely, thereby solving the lever-arm effect altogether.

The thesis had set out to implement the full framework in real time on ROS, albeit this proved to be infeasible due to hardware limitations regarding the HoloLens. It is in theory possible to convert the raw HoloLens data into a `rosbag`, provided that the timestamps are in Epoch and the clock drift has been compensated for, and then port the source code of the calibration framework into ROS nodes. Nonetheless, due to the erratic sampling rate of the HL, the raw streams need to be upsampled. By employing a sliding window approach which takes the samples from the raw streams for a pre-determined time interval, the calibration framework can run for these samples and produce results, before discarding them and move on to the next window.

Within the scope of this work, two IMUs have been actively tested while the HL camera was used for verification purposes. As discussed in previous sections, the homogeneous matrix obtained from the HL camera can provide better information than the HL IMUs due to its superior scene understanding algorithm. For further improvements, this matrix can be utilized in a SLAM-oriented approach. Since the HMS also has a Velodyne LiDAR housing on top, the `rosbag` created from this can be merged with the rest of the sensors as well.

The clock delays which were discussed in 2.1 were accepted as non-existent within the scope of this work because they are unquantifiable. This assumption should be made for further improvement of the framework because the amount of effort put into determining these delays will likely not improve the accuracy.

A lot of effort was put into tinkering with the inner workings of the Kalibr framework during the experiment stages because it was thought that the Xsens IMU can be supplied to the optimization problem instead of the ego motion extracted from the camera. The idea behind this modification was using the capability of the continuous-time optimization problem to solve both the extrinsic rotation and the distance between the sensors. This modification improvement can be studied in the future works as well.