FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Multi-IMU:
Multiple Inertial Measurement Unit Systems in Mixed Reality

Adnane Jadid

FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Multi-IMU: Multiple Inertial Measurement Unit Systems in Mixed Reality

Adnane Jadid

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation.

Vorsitzender:		Prof. Dr.-Ing. Darius Burschka
Prüfer der Dissertation:	1.	Prof. Gudrun J. Klinker, Ph.D.
	2.	Assoc. Prof. Dr. Yuta Itoh

Die Dissertation wurde am 05.05.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.10.2021 angenommen.

بِسمِ اللهِ الرَّحمٰنِ الرَّحِيمِ

Acknowledgments

I would like to start by thanking my family for the immense support they show me, for the motivation they provide, and for the love they give. I cannot imagine what would my path be without all of you.

I would like to express my sincere thanks to Gudrun Klinker, for her scientific support over the past three years and for countless valuable food for thought. I greatly appreciate the trust you have placed in me.

My thanks also go to Yuta Itoh for the opportunity to do the experiments with Multi-IMU in his lab and use of the equipment, and for the fruitful discussions, which all contributed to my studies and this work in particular.

I would also like to thank my dearest friends, Florian and Daniel Detig. Knowing there are people in your life you can always count on gives an unmatched incomparable feeling. Thank you, brothers.

I dedicate this work to the memory of my Father.

Abstract

Mixed Reality (MR) is an immersive technology that allows the user to interact in real-time with both real and virtual objects. As such, MR can be viewed as a blend of real (physical) and digital (virtual) worlds that helps the boundaries between human, computer, and environment fade. One of the major underlying technologies of Mixed Reality is tracking.

Tracking is determining the pose (i.e., position and orientation) of an object in a chosen coordinate system. Tracking technologies used in MR should provide accurate real-time tracking, and that accuracy is the most challenging and crucial part to achieve in any MR application. Even a small tracking error can cause a large negative impact on the application, destroying the immersive experience and providing the wrong perception.

Various tracking technologies exist. One of them is realized through Inertial Measurement Units (IMUs). IMU sensors, especially modern ones based on Microelectromechanical Systems (MEMS), are comparatively low-cost, small, less power consuming, and free from some problems that persist in sensors of other types. However, as is generally the case, IMU tracking has its own disadvantages, the major being the systematic presence of errors in the obtained measurements. Systematic minimization or elimination of these errors is the principal goal that needs to be reached before tracking can be performed with IMUs.

To enhance tracking, systems with multiple sensors can be constructed. One of the cases that is of the major interest in this study is Multi-IMU, i.e., a tracking system with multiple inertial measurement units. The research in the field of Multi-IMU tracking is not currently as developed as for poly-sensor tracking. That fact, combined with the promising potential of the area has motivated the choice of the study direction.

This thesis contributes detailed insights into hardware implementation, data enhancement, sensor fusion, and pose estimation by the use of a Multi-IMU system.

Kurzfassung

Mixed Reality (MR) ist eine immersive Technologie, die es dem Benutzer ermöglicht, in Echtzeit sowohl mit realen als auch mit virtuellen Objekten zu interagieren. Als solche kann MR als eine Mischung aus realen (physischen) und digitalen (virtuellen) Welten betrachtet werden, die die Grenzen zwischen Mensch, Computer und Umgebung verschwimmen lässt. Eine der wichtigsten zugrunde liegenden Technologien der Mixed Reality ist das Tracking.

Tracking ist die Bestimmung der Pose (d.h. Position und Orientierung) eines Objekts in einem gewählten Koordinatensystem. Die in der MR verwendeten Tracking-Technologien sollten ein genaues Tracking in Echtzeit bieten, und diese Genauigkeit ist die größte Herausforderung und der entscheidende Teil, der in jeder MR-Anwendung erreicht werden muss. Selbst ein kleiner Tracking-Fehler kann einen großen negativen Einfluss auf die Anwendung haben, das immersive Erlebnis zerstören und eine falsche Wahrnehmung vermitteln.

Es gibt verschiedene Tracking-Technologien. Eine davon wird durch die Inertial Measurement Units (IMUs) realisiert. IMU-Sensoren, insbesondere moderne, die auf mikroelektromechanischen Systemen (MEMS) basieren, sind vergleichsweise kostengünstig, klein, verbrauchen weniger Strom und sind frei von einigen Problemen, die bei Sensoren anderer Typen bestehen. Allerdings hat das IMU-Tracking, wie allgemein üblich, seine eigenen Nachteile, von denen der größte das systematische Vorhandensein von Fehlern in den erhaltenen Messungen ist. Die systematische Minimierung oder Eliminierung dieser Fehler ist das Hauptziel, das erreicht werden muss, bevor das Tracking mit IMUs durchgeführt werden kann.

Um das Tracking zu verbessern, können Systeme mit mehreren Sensoren konstruiert werden. Einer der Fälle, der in dieser Studie von großem Interesse ist, ist Multi-IMU, d.h. ein Tracking-System mit mehreren Inertialmesseinheiten. Die Forschung auf dem Gebiet des Multi-IMU-Trackings ist derzeit noch nicht so weit entwickelt wie beim Poly-Sensor-Tracking. Diese Tatsache, kombiniert mit dem vielversprechenden Potential des Gebietes, hat die Wahl der Studienrichtung motiviert.

Diese Arbeit liefert detaillierte Einblicke in die Hardware-Implementierung, die Datenanreicherung, die Sensorfusion und die Pose-Schätzung durch den Einsatz eines Multi-IMU-Systems.

Contents

"I'd take the awe of understanding over the awe of ignorance any day."

— Douglas Adams

1. Introduction

Mixed Reality (MR) is an immersive technology that allows the user to interact in real-time with both real and virtual objects. As such, MR can be viewed as a blend of real (physical) and digital (virtual) worlds that helps the boundaries between human, computer, and environment fade.

Forward-thinking companies, including most of the computer-oriented industrial "giants", place their hopes into mixed reality technology: Microsoft Hololens, Samsung Odyssey, Lenovo Explorer are all mixed reality-based projects just to name a few. The applications proposed by Microsoft [188] include employee connection and collaboration enhancement, remote assistance, medical training, online shopping, gaming, and many others. Their technology was adapted by another industrial giant, Ford, to enhance vehicle prototyping, engineering, and design as well as to provide employee training and even increase sales [57].

Like any modern state-of-the-art technology, mixed reality is complex in nature. It is based on the advancement of such distinctly different fields as display technology, graphical processing, spatial sound, environmental input, computer vision, etc. And, of course, no MR-based product can be imagined without *tracking* hardware and software.

Tracking is determining the pose (i.e., position and orientation) of an object in a chosen coordinate system. Tracking technologies used in MR should provide accurate real-time tracking, and that accuracy is the most challenging and crucial part to achieve in any MR application. Even a small tracking error can cause a large negative impact on the application, destroying the immersive experience and providing the wrong perception.

There are numerous available choices for tracking devices: optical, magnetic, satellite, etc. In a large number of MR-based devices, tracking is performed with optical systems. These systems, although providing good quality of tracking and being in any case necessary for the functioning of any MR device, exhibit some common problems, such as high power consumption, occlusion, and generally large size.

An alternative or at least complementary tracking solution lies with the Inertial Measurement Unit (IMU) technology. IMU sensors, especially modern ones based on Microelectromechanical Systems (MEMS), are comparatively low-cost, small, less power consuming, and free from occlusion problems. It should also be noted that most (if not all) of the nowadays-produced smartphones are equipped with inertial sensors, so their availability to the general public cannot be questioned. However, as is generally the case, IMU tracking has its own disadvantages, the major being the systematic presence of errors in the obtained measurements.

Still, it is always preferable to invest in software development along with or even rather than hardware design: once the proper software is developed, its replication doesn't induce additional costs for the producer. Therefore, a lot of research is focused on increasing the tracking ability and performance of low-cost IMUs by eliminating errors they exhibit with the development of new software solutions.

A very promising and advantageous direction of tracking is tracking with multiple sensors. The multi-sensor tracking can be performed using combinations of trackers of different types. Common combinations are optical + inertial tracking [61, 113, 158], inertial + satellite tracking [195, 198, 303, 19], and Ultra Wide Band (UWB) + inertial tracking [119, 300, 75]. Also, a multi-sensor tracking system based on single-type sensors can be constructed. The focus of this work is on multi-IMU tracking, i.e., tracking with the use of several inertial measurement systems.

The research in the field of Multi-IMU tracking is not currently as developed as for poly-sensor tracking. That fact, combined with the promising potential of the area has motivated the choice of the study direction. In the course of the research, the following topics arose that are brought as the main point of this study.

- Using Multi-IMU Vs. Standalone IMU
  An important investigation is directed in this topic: Can a Multi-IMU system outperform a standalone IMU? Special interest is, naturally, focused on the case of a comparison of a low-end IMUs system with a high-end standalone IMU.

- Enhancing reliability and accuracy of IMU data
  The errors in pose estimation can come from the errors in the obtained sensor data, from how data are *obtained*, or how the data are *used*. An investigation of error sources, as well as methods for their suppression, is one of the major focuses of this work.

- Pose estimation with Multi-IMU
  When estimating pose with multiple sensors, various choices exist for the usage of individual and combined data flows. The development of a pipeline for pose estimation with Multi-IMU is one of the main goals of this study.

- Fusing Multi-IMU data
  Closely related to the previous topic, the fusion of Multi-IMU data is a major part of the developed pipeline.

- Artificial intelligence applied to Multi-IMU
  Artificial intelligence is a vast field of modern technologies that can be used to enhance performance in various applications. The use of artificial intelligence technologies for Multi-IMU fusion is carefully considered in this work.

Additionally to the goals described above, a specific focus of this work is on presenting methodical and systematic guidance for employing Multi-IMU in MR systems. This thesis contributes detailed insights into hardware implementation, data enhancement, sensor fusion, and pose estimation by the use of a Multi-IMU system.

This work consists of the following chapters.

Chapter 2 Background:
This chapter is devoted to a short review of the necessary background on human-machine interaction, immersive technologies, and artificial intelligence.

Chapter 3 Inertial Measurement Unit:
In this chapter, the major components of IMUs are presented. This chapter also provides an insight into the types of IMU errors, thus setting the base for data enhancement discussed in the following chapters.

Chapter 4 Calibration:
This chapter explains the calibration principles in general and reviews existing IMU calibration techniques. A section on Multi-IMU calibration is presented that provides insight on the practical enhancement of the data obtained with Multi-IMU.

Chapter 5 Filtering:
This chapter introduces the filtering methods utilized for minimizing the noise in the IMU data.

Chapter 6 Registration:
This chapter explains and reviews registration methods of IMU to other IMUs and to tracking sensors of different types. A section on practical Multi-IMU registration is presented, that proves the necessity of registration in the presence of multiple sensors.

Chapter 7 Synchronization:
In this chapter hardware and software synchronization in multi-sensor systems is considered. A section on practical Multi-IMU synchronization shows the need for synchronization in systems with multiple sensors.

Chapter 8 Fusion:
This chapter exhibits the analytic fusion methods and machine learning-based fusion techniques for systems with multiple sensors. A section on comparison of the most popular traditional fusion algorithms in an applied practical problem is provided in this chapter.

Chapter 9 Pose Estimation:
This chapter explains how to estimate the pose by using an exclusively Multi-IMU system. A pipeline, presented and explained in this chapter, systematizes and organizes the previously explained procedures that need to be performed for pose estimation with Multi-IMU in a traditional setting. A series of experiments is provided based on the proposed pipeline. Another approach for pose estimation based on machine learning is also presented in this chapter with results supporting its applicability.

Chapter 10 Multi-IMU Interaction Applications:
This chapter reviews existing and potential applications of Multi-IMU for interaction.

Chapter 11 Conclusion:
The last chapter summarizes the results, provides finalizing analysis, and gives future work suggestions.

2. Background

This chapter presents the necessary background for the work. Types of human-machine interactions are first discussed. Then, one of the types, namely, immersive interaction, is discussed in detail. Types of immersive technologies are described and compared. The concepts of the Reality-Virtuality continuum and Extended Reality are explained.

Next, important components of immersive technologies are discussed. The general overview is provided, and references to the corresponding chapters of this work are given.

Lastly, a large part of this chapter is devoted to the discussion of artificial intelligence as an important tool in implementing immersive technologies. Machine learning and deep learning technologies are reviewed. A large subsection on neural networks is presented in this chapter as neural networks are an essential instrument of deep learning and, therefore, artificial intelligence.

2.1. Human-machine Interaction

> "A good tool is an invisible tool. By invisible, we mean that the tool does not intrude on your consciousness; you focus on the task, not the tool."
>
> — *Mark Weiser*

Human-machine Interaction (HMI) is a broad field of study that covers communication and interaction ways, interfaces, and engineering solutions that provide interaction between a human user and a machine they operate. The term usually refers to both hardware and software components of the system. The design of HMI systems is usually closely related to the concepts of ergonomics, usability, and process optimization.

HMI design considers the accessibility of the machine controls, efficiency and speed of access, consistency of control actions, and the necessary efforts for operating the machine. The concept can also be extended to include the design of a workplace: the control panel placement, the data input devices arrangement, or microclimate and lighting of the workplace [43].

A functional HMI helps the user to reach top productivity with the machine, which generally calls for ensuring rich information exchange in the system, reduction of the user's cognitive load, and the increase of operational capability and productivity. For that reason, the field of HMI is always developing, to provide new ways for communication and interaction between the machine and its user.

Professor Jonathan Grudin, a pioneer of computer-supported cooperative work, in his work [101] states (concerning the field of HMI development), that " ... Our best chance to anticipate change is to find trajectories that extend from the past through the present. The future will not resemble the present, so it is worth trying to prepare for it." In this context, it means that in order to further develop the means of interaction between a machine and its user, one should first study the already existing types, and understand their advantages and disadvantages.

2.1.1. Batch interface

The first general-purpose computer, ENIAC, constructed for military use during the Second World War, was based on a system of vacuum tubes [52]. Because of the limitation of the technology, memory for ENIAC (and other vacuum-tubes-based machines) was extremely expensive. Therefore the machines were primarily used for computations and not for symbolic representation or information processing. The main goal of HMI design was to reduce operator burden by providing a mechanism for loading stored programs from a physical object instead of manually setting switches and reattaching cables.

The solution was presented in the form of punched cards or paper tape (for input) and line printers (for output). To submit a job for a batch machine, a deck of punched cards had first to be created on a separate device, called keypunch. The interface of that machine, in its turn, could also only be described as unforgiving because of very strict syntaxes in combination with a prone to mechanical failure. Then, after preparing the sequence of cards, an operator would feed the deck to the computer and wait for a printout with a final result or an error log.

So, this type of interface still only allowed for batch processing in the sense that all the details of a job needed to be specified before the run, and the output could only be received after all the processing is finished. Therefore, these interfaces are called batch interfaces.

After 1957 a monitor program was usually introduced into the process that allowed the user to monitor running services, provided better checking on submitted jobs, and gave more useful feedback for the user [27]. These monitor programs represented the first step in the direction of operating systems and explicitly designed human-machine interfaces.

The era of batch interfaces started with the construction of the first general-purpose computer in 1945 and ended in 1968 when command-line interfaces were introduced.

2.1.2. Command-line interface

Command-line interfaces evolved from batch monitors combined with system consoles. The communication between components was implemented via request-response transactions; a special syntax and vocabulary were developed for the textual commands that represented the requests [101]. As opposed to the batch interfaces, with the new type of interface the user could change or adjust the job plan in response to feedback from the system. Despite this improvement, command-line interfaces still required the user to learn and memorize a lot of information to control.

As for the hardware, the earliest machines used teleprinters (devices used for the automatic telegraph transmission/reception, invented in 1902). On the one hand, it was a strategy to economize by using already existing tools, while on the other teleprinters were an interface that was familiar to many users and engineers.

In the mid-1970s, video-display terminals became widely spread, which helped cut the latency of the system, because the monitors allowed to deliver information much faster than printers. They also spared ink and paper, further insuring the availability of programming to users.

Although new types of interfaces were developed in the same type as the command-line interface, it remains in wide use both in industry and common-day computer use.

2.1.3. Graphical user interface

The graphical user interface (GUI) is a form of human-machine interface that provides the access to all the system objects and functionalities via graphical elements on a screen, such as windows, icons, menus, buttons, lists, etc.

The start of the GUI era is usually associated with the research made in the 1960s by Douglas Engelbart and his group that developed important components of both hardware and software suitable for a more convenient user interface [67, 68, 69]. Their inventions and developments included bitmap output to screens, computer mouse, and hypertext. All of that provided a base for the actual creation of a machine with a GUI that was brought from the conceptual stage to physical implementation and use by Xerox PARC research laboratory.

Although most of the operational systems that provide a graphic user interface only have it as an add-on over the system, some independent implementations incorporate GUI on deeper levels. For example, there exists a graphical realization of the BIOS Setup utility [9] that allows using the computer mouse for setup before the operating system is loaded.

2.1.4. SAA user interface

In 1987 IBM introduced the Systems Application Architecture (SAA) standard that contained the Common User Access specification, aimed to provide a common user interface for the entire product line of the company. This specification defined elements of the applications such as dialog boxes, menus, status bars, and keyboard shortcuts. Although the SAA standard itself didn't prove to be a successful set of guidelines [172], the elements of the Common User Access specification are still present in modern-day applications.

2.1.5. Direct neural interface

The history of direct neural interfaces (or brain-computer interfaces, BCI) started in 1924 with Hans Berger's development of electroencephalography [96], a method for recording the electrical activity of the brain.

Full research on the particular application of this invention to HMI was only started in the 1970s at the University of California, Los Angeles by the research group of J. Vidal [274,

275]. The research helped understand the processes and develop the tools for one- or two-way information exchange directly between a brain and an electronic device. After several years, the yielded results allowed to produce and introduce to the general public devices that helped regain the lost hearing, eyesight, and motor skills, thus creating a new field of neuroprosthetics.

Although medical treatment and rehabilitation remain one the major uses nowadays, there exist some alternative applications and concepts that include communication devices (based on pupil-size oscillations or directly on brain activity), hands-free mobile phone systems, and even for games control.

2.1.6. Touch interface

Touch interfaces were introduced in the 1980s [11], first in the form of touchpads, allowing the user to interact with the machine by using a stylus or even their fingers. In 2007 with the launch of the iPhone the touch technology was popularized to a new extent. The uses of this type of interface in the device screen (and not in an additional pad) combined a user-friendly data visualization technology with a straightforward input mechanism. With the improvement and refinement of the technology, touchscreens became more and more popular, showing up on computers, laptops, and tablets.

2.1.7. Voice interface

The development of voice user interfaces presented a means of "talking" to the machines. One of the most important features of these interfaces is that they can be used hands-free. That, combined with the now-common use of (almost) natural languages for HMI, provides for an efficient and intuitive means of interaction and communication.

Nowadays, voice user interfaces can be seen in automobiles, home appliances, television remote controls, and, of course, mobile phones and computers. Although graphical user interfaces can communicate much more information than purely auditory ones, the latter have their use, especially when used in combination with GUI. The greatest potential of pure voice interfaces is reached in the cases when the user has disabilities (not able to use a mouse/a keyboard or see the screen), when the user is in an eyes-busy / hands-busy situation (such as driving cars), or when there is no physical access to a muscle-machine interface device [200].

2.1.8. Immersive interface

Developments in computer modelling, improvement of device manufacture processes, and increase in levels of global computerization call for the use of new technologies for human-machine interaction.

Immersive technologies, i.e., technologies that allow the user to be immersed in the artificially created realities, are now being employed for that. Interfaces that employ immersive technologies visualize the digital content and combine it with the real-world scene, transforming the operation environment. The main advantage of immersive interfaces is the

sense of personal presence inside the specifically designed environment. That helps the user successfully obtain the desired results of HMI, be that a more realistic computer game experience or better use of a training device.

The next section is devoted to the description of immersive technologies that are commonly used for HMI.

2.2. Immersive Technologies

> "You are in the story, you speak to the shadows and they reply, and instead of being on a screen, the story is all about you, and you are in it."
>
> —Pygmalion's Spectacles, 1935
> *Stanley G. Weinbaum*

Immersive technologies are technologies that extend reality or create a new reality by leveraging three-dimensional space. Some types of immersive technologies extend reality by overlaying digital content in a user's environment. Others create a new reality by completely shutting a user off from the rest of the world and immersing them in a digital environment.

Although formally immersive technologies in the modern meaning of the term only started emerging in the mid-1980s, researchers and engineers were interested in the concept long before then. A lenticular stereoscope invented in 1849 by David Brewster [38] can be considered an "ancestor" of modern head-mounted displays. The device consisted of a pair of magnifying lenses; the user placed two images representing the same object/scene taken from two different points of view in such a way that the left eye only saw the left image, and the right eye - the right image. Such a particular placement created a 3D effect and even allowed the user to feel themselves inside the scene.

Another device invented in 1962 by Morton Heilig was called "Sensorama" [114]. It allowed the users to feel a deep immersion effect: To provide an experience of a street walk, his device not only showed a pre-recorded video but also played sounds of a busy street and even produced specific smells for different locations.

Although Morton Heilig proposed his device as an entertainment product or, naturally, as a piece of art, the same idea was implemented in flight simulator systems for pilot training. Starting from the 1970s, these training systems included computer graphics [212], and although the quality of the graphics was not that impressive, these systems already allowed for real-time interaction. Nowadays, immersive training systems are not only used for flight simulators but also for other specializations, especially for ones where personnel is required to deal with hazards or emergencies.

Naturally, the use of immersive technologies is not limited to educational and training fields. Another major consumer of them is the entertainment sector. And the impulse for the fast growth of immersive applications can be directly traced to the emergence of modern head-mounted displays, virtual reality glasses, and other devices that create the feeling of complete immersion in the scene. The devices of today not only allow users to look around

in the scene but also to interact with it. The major role in the success of a tool belongs not to the general realism of the image but to the well-organized and well-coordinated work of sensors that track and record the user's movements and transform them into visual and/or tactile experience.

The following subsections are dedicated to the description of the major types of immersive technologies.

2.2.1. 360-Degree

360-Degree technology is the most basic of immersive technologies. Content produced by and for this technology (both user-generated and company-engineered) focuses on showing images or videos with 360 degrees views, i.e., allowing the user to look in any direction. Examples of the applications include maps services with street view, museum tours, online shopping, and real-estate business.

Some authors consider 360-Degree a part of Virtual Reality technology, see for example [17].

2.2.2. Virtual Reality, VR

Virtual Reality (VR) is defined as a computer-generated, interactive, three-dimensional environment in which a person is immersed [14]. In VR, the user is shut off from the rest of the world while being surrounded by specifically engineered 3D content.

The major components of the hardware of a VR system are input devices, the computer/VR engine, and output devices. With the input devices, the user can interact with the virtual world: all the relevant movements/actions of the user are tracked and sent to the processing system in order to adjust the virtual environment. The usual input devices can include

- tracking devices/position sensors: electromagnetic, optical, mechanical, ultrasonic sensors, data-gloves, neuro-controllers, etc.;
- point-input devices: 3D computer mouse, joystick, trackball, touchscreen, etc.;
- voice recognition devices.

The VR engine or computer system is responsible for computing and generating graphical models, computer graphics (CG) processing and displaying in real time; it also handles the interaction between the user and the input-output devices.

The output devices pass the feedback from the engine to the user to simulate the senses with graphics, audio, touch, and sometimes even smell and taste. The most common choices for graphics output are the stereo display monitors and head-mounted displays (HMD). The latter provides a higher level of immersion by presenting a stereo view that is recognized as 3D by the human brain.

The field of virtual reality has been popular for some time now, which has led to the emergence of new terms for related and close notions. Some of them are clarified here.

360 VR
This term usually refers to any 3D content with VR mode, when the user can explore the content by looking in any direction. The major distinction from the previously discussed 360-Degree technology is the hardware: While for 360-Degree technology a mobile phone, a tablet, or another similar device can be used, 360 VR requires the use of a headset or HMD.

Mobile VR
This technology employs mobile phones as a processing part of the system: The head-mounted display should be connected to a smartphone.

True VR
This term was suggested for use to distinguish the original VR definition from all of the similar technologies that emerged later. True VR technology employs powerful computers and sensors to provide natural-looking content, keep track of the user's movement and surroundings, and adjust the content accordingly.

In-VR
In-VR technology specializes in considering the depth, producing a stereoscopic image. That means that if the user moves closer/further away from an object in VR, the content adjusts to that movement.

The application field of VR is wide and includes (but is not limited to) enhancing manufacturing process, professional training (military, medical, etc.), general education, virtual prototyping, ergonomic studies, entertainment, assembly sequence simulation, etc [209].

- **Design, manufacturing and maintenance**

  In the industry, VR has proven to be an effective tool for helping employees evaluate the product design. The use of simulation helps reduce time and money investments in design, manufacturing, and maintenance in industrial applications.

  Berg and Vance in 2017 made a review article [30] devoted to the industrial applications of VR. The authors conducted a survey of several laboratories and research groups that contributed to applying VR in industrial problems. The visited and surveyed facilities included organizations from the aerospace, agriculture, automotive, construction, consumer goods, energy, and military industrial domains. The authors also focused on surveying the use of VR for different participant roles. Table 2.1 provides an insight on the use of VR in different jobs within an industry.

  The real-life use cases that were observed by the authors include visibility/viewability, ergonomics/reachability, packaging, aesthetic quality/craftsmanship, storytelling, abstract data visualization, and communication across disciplines. The authors also note that the list of the use case categories is not exhaustive and is ever-expanding.

  The authors conclude that industry makes a lot of profit by cooperating with academic and research communities that study VR and by applying the results of the research to their particular practical problems.

Category	Responsibilities
Maintainer	Tasks within this category comprise configuring, calibrating, and upgrading both software and hardware components of a VR system. Exploring new technology and troubleshooting existing technology falls into this category.
Operator	Operators manage the scheduling of the system and help users interact with the system. Responsibilities range from turning on and preparing the hardware to altering software settings to support individualized use cases.
User	These are people who use VR for the benefits the systems provide. Users rarely have responsibilities that support the VR facility itself. Organizationally, users are outside of the other categories.
Builder	Before data can be loaded into the virtual environments, it must be acquired, converted, and touched up. Builders prepare digital content to be integrated into the virtual environment. Interactions and animations are added once content is prepared. They communicate with users to ensure the VR experience meets the intended goals.
Manager	Responsibilities consist of organizing large projects, managing staff, and setting goals for the VR facility. Tracking the use of the VR system can be an important part of Region of Interest calculations.

Table 2.1.: Participant roles. Source: [30]

Assembly simulation is one of the topics in industrial design and construction that has been of high priority and major concern for years. Various approaches that help assemble the desired system qualitatively, inexpensively, and quickly exist, based on a diverse range of perspectives. Most of the approaches concentrate on providing a modelling tool that allows for a virtual assembly process as opposed to a physical one [281, 163, 298].

Research works concerning assembly simulation with VR started rapidly emerging at the end of the 1990s. Paper [258] by a research group from the Aachen University of Technology, published in 1998, focuses on the integration of VR-based assembly simulation into computer-aided design (CAD) environments. A scientific group from the Technical University of Darmstadt published several works on collision detection in VR and virtual prototyping with VR, see for example [296, 56, 297].

Since the 1990s, virtual reality technologies have been developing as well as assembly simulation technologies. Paper [264] by Tao, Lai, and Leu focuses on manufacturing assembly simulation in VR in the context of the so-called Industry 4.0 era. According to the authors, the current major needs of the market are flexibility and efficiency

of the assembly methods that can help shorten the design and production time. By employing VR technologies, realistic, accurate, and functional digital models can not only be created and viewed but also interacted with, allowing for better end assembly simulation results.

The manufacturing applications also often focus on the topic of digital twins. The Digital Twin [111] integrates all data (test, operation data, etc.), models (design drawings, engineering models, analyses, etc.), and other information (requirements, orders, inspections, etc.) of a physical asset generated along its life cycle that leverage business opportunities. Figure 2.1 shows an example of the use of a digital twin in VR for physics simulations.

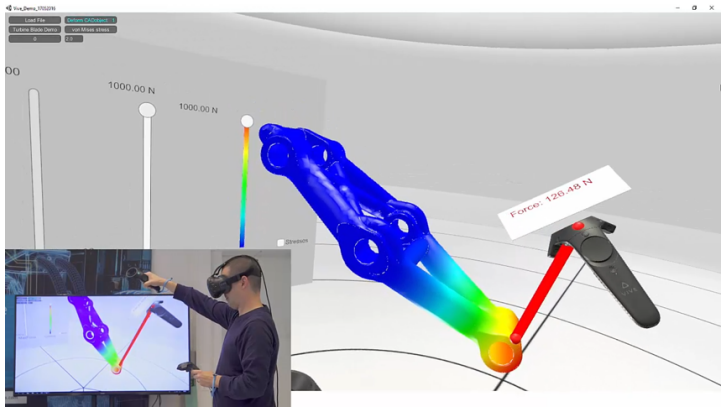


Figure 2.1.: Interactive 3D multi-physics finite element simulation in a virtual environment. Source: [111]

- **Education**

  The use of virtual reality in education has also been a much-researched topic for some time now.

  The major advantages of using VR in education were formulated in [287]:

  1. Immersive VR furnishes first-person non-symbolic experiences that are specifically designed to help students learn the material.
  2. These experiences cannot be obtained in any other way in formal education.
  3. This kind of experience makes up the bulk of our daily interaction with the world, though schools tend to promote third-person symbolic experiences.
  4. Constructivism provides the best theory on which to develop educational applications of VR.
  5. The convergence of theories of knowledge construction with VR technology permits learning to be boosted by the manipulation of the relative size of objects in virtual worlds, by the transduction of otherwise imperceptible sources of information, and by the reification of abstract ideas that have so far defied representation.

Although the article was published in 1993, the main idea is valid nowadays: the direct interaction with an object helps students learn, and since the interaction with the object itself can be dangerous, expensive, time-consuming, or not easily controllable, a virtual object should be used as a substitution.

Work [266] by Teklemariam, Kakati, and Das focuses on the use of VR technologies in design education. The major concern of the authors is the gap that exists between the imagined/designed products and their final implementations. By combining VR with CAD and allowing for getting haptic feedback along with space and form perception, this gap can be bridged through Rapid Prototyping. As the end result, the authors expect that students who use the VR-CAD systems can develop a better understanding of the design process and of the CAD systems themselves.

Paper [225] offers a review of VR applications for higher education. The paper offers the statistical data on the use of different VR technologies, showing the high-end HMDs are the most popular choice (76%) with mobile VR being the second choice (20%). It also provides insight into the relationship between the design elements of an application and the learning content and between the application domain and the learning content. The article shows high demand and potential for VR as a tool for higher education.

- **Training and simulation**

  As stated in research [213] conducted by Palmas, Labode, Plecher, and Klinker, the use of simulation in VR allows users to train without supervision in a safe and controlled environment, gain new skills, and then apply them to real-life situations. The authors propose also to combine VR and gamification to speed up learning. The paper presents the results of an experiment, where 50 participants were divided into two groups. Each of the groups received the same VR training on assembly, but group 1 used a gamified version, and group 2 used a non-gamified version. The results show that in terms of time taken for the provided tutorial and specific errors in the training session the group that used a gamified version performed better than the other one. Namely, group 1 finished the training 12.2% faster than group 2, and the mean count of committed errors of the first group is lower than that of the second group by 30.2%.

  Another research [214] presents and assesses a different type of training: training of public speaking skills. The authors collect the data from 44 participants and impose some metrics: timing (elapsed time as compared with the estimated needed time and with a set time-limit), transcript of the speech (with marked filler words), eye contact (audience, laptop, screen, floor, others), positioning (where the speaker was standing), body language (viewing direction, hand position, body direction), and response analysis (the response is modelled based on eye contact, voice, gestures, and body language). Almost all of the participants (95%) reported that they improved presentation skills by using the VR-based systems.

  VR can be also used to simulate a virtual courtroom [206]. In some cases that are by nature hard to try in court, the victims are reluctant to take a stand because of the

necessity of facing the (alleged) offender in an unfamiliar environment after already being strained by their situation. A common practice in these cases is to prepare the victims before the trial by sketching the courtrooms on paper, e.g., showing them the sitting arrangement. However, this method is outdated and is not proven to show any positive results. The suggestion of the author is to employ VR technologies to immerse the user into the unknown courtroom, thus preparing them for the actual trial. The paper shows the results of research in progress, but the provided results allow the authors to hope for successful implementation and application in practice.

Another example of VR usage in the training is in the field of future ship navigators training [176]. The aim of the research is to determine the role of VR in the educational process, as applied to marine navigation training. The results show great potential in providing professional training of new quality by immersing the trainees into the virtual reality where they can solve professional or quasi-professional tasks and develop necessary skills.

- **Health care** Although other immersive technologies find more applications in health care, VR can also be used, e.g., for rehabilitation purposes [236]. The paper provides a review of VR-based rehabilitation systems, studying the influence of immersion on the health improvement outcome, the connection between entertainment level and patients' adherence to the use of the system, and the influence of haptic feedback on the performance. Although the authors state that various systems have different results without any clear correlation to the proposed metrics, the general idea of using VR technologies for rehabilitation is shown to have a large potential.

- **Entertainment** The entertainment sphere nowadays exploits VR technologies in various spheres. The obvious example is games, but other applications exist.

  One of them is the cinematography. Paper [246] discusses the question of editing movies that are to be shown in VR. The major distinctive feature of these movies is that the viewer controls the camera orientation. The main research goal is to determine under which conditions does the traditional editing techniques provide an appropriate level of continuity perception. The authors propose some guidelines for editing, e.g., to use cognitive event segmentation to predict the behavior of the user and improve the editing process with the obtained information.

  Another interesting entertainment application is museum tours. Paper [160] investigates the possibilities of employing absorptive (i.e., education and entertainment) experience for the enhancement of user's general experience. The study shows that absorptive experience increases the interest in VR museum tours and also the interest in a personal visit to the site.

2.2.3. Augmented Reality, AR

Augmented Reality (AR) is a technology that allows for enhancing the real-world environment with artificial sensory information. In other words, AR supplements reality with additional virtual information. Roland T. Azuma defines AR as a technology that integrates 3D virtual objects into a 3D real environment in real time [15].

In 1997 Azuma defined the essential properties of AR systems [15]:

- It combines real and virtual objects in a real environment.
- It runs interactively and in real-time.
- It registers (aligns) real and virtual objects with each other.

Although the term augmented reality itself was coined in the early 1990s by Thomas Caudell, who worked for the USA aircraft manufacturing company Boeing [44], the head-mounted display considered to be one of the first AR devices was presented by Ivan Sutherland in 1967 [261]. Nowadays, the range of AR devices is genuinely wide. Three main classes are usually distinguished: head-mounted, handheld, and spatial devices [276].

As for the software part of AR device production, the work [233] by Reicher et al. provides clear guidelines. The research group presented an AR system architecture with specified relationships between the software components (Figure 2.2).

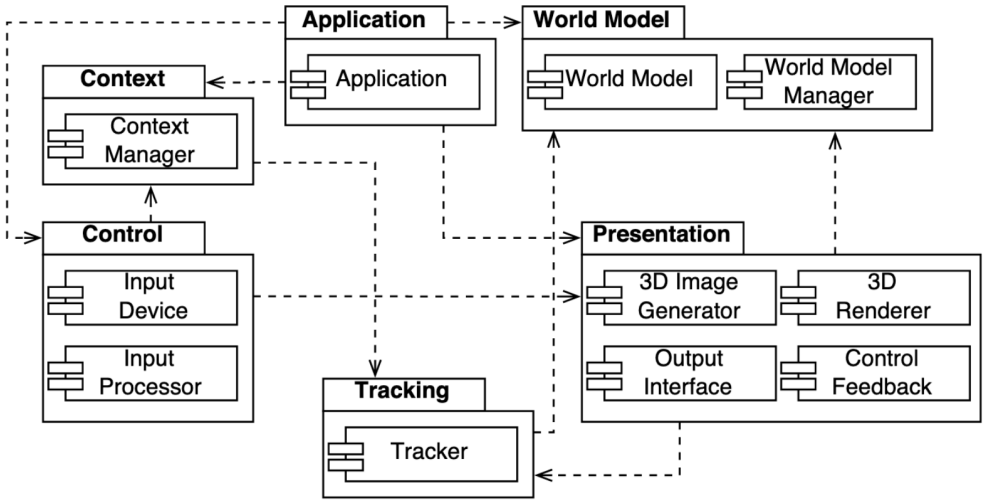


Figure 2.2.: Reference architecture showing subsystems and their dependencies. Source: [233]

In the last decade, AR has gained a lot of attention from the science community and is seen as a means to superimpose various information on the real world [41, 148]. Augmented reality has found application in various areas. Some applications are discussed below.

- **Design, manufacturing and maintenance**

  Like VR, AR also found its applications in industrial applications. For example, the research by a group of students from the group of Prof. Klinker [149] published in 2004 considers the use of AR for diagnosing machine malfunctions. The paper shows the investigation path of the posed problem, which leads to a potential solution, along with the prototypical demonstrator that illustrates the usage of AR in an industrial application.

  Another application is as a comparison tool in the automotive industry. Paper [203] by Nolle and Klinker investigates the possibility of using AR in the automotive industry for comparing produced (real) car parts with their respective models. By providing a comparison, the produced parts can be verified, e.g., if they correspond to the latest version of the design or if they were manufactured with adequate production precision. The paper focuses on visualization schemes that help superimpose CAD construction data on the real physical objects, with both real and virtual parts being visible in the same place.

  Paper [240] considers using AR for machine maintenance to reduce the time needed for the performed maintenance operations and procedures. The idea is that when the user catches the machine with their camera, the image is recognized and the components of the machine are labeled in real time. Potentially, the image can also be enhanced with instructions or demo videos explaining the maintenance process. The authors state that the system helps reduce the training time (if the system is used as a training tool) and also the maintenance time itself.

- **Education**

  Augmented reality also found application in providing a base for digital experiments in education [171]. The authors consider the use of mobile-based AR systems for increasing the overall interest of pupils and students in science and studies and for replacing the desktop educational applications with more affordable alternatives.

  Another research [271] focuses on the same topic from the perspective of the teachers. The authors conveyed a survey that showed that although the results that AR applications can potentially bring to the educational process can be desired by the teachers, not all of them are ready to introduce the new technologies in the studies. However, the general trend seems to show an increasing interest in AR technologies for education.

- **Simulation and training** Potentials of augmented reality in training have been reviewed in [255]. The authors first divide the professional training by its objectives into the classes of training on the job and training near the job. The authors then provide a wide range of possible activity types that can be assisted with an AR system, including, for example, warehouse operations, assembly, quality control, or decision making. The authors provide a qualitative assessment of the existing applications and conclude that

with a reasonable amount of care and patience, an appropriate AR application can be found or developed for the particular needs of the activity.

- **Health care**

  Health care is a major application area for immersive technologies, in particular, for augmented reality [77].

  Paper [186] focuses on rehabilitation therapy. The authors present an AR dance game that is designed to help upper limb amputees improve their rehabilitation. Within the game, the patients are following the advice of a virtual dance instructor specifically designed to enhance the rehabilitation process. The patient's position is tracked while they are performing the movement and hand gestures are analyzed. The gamification component also shows the patient their specifically calculated score based on the performance. The paper presents a prototype of the AR-based device that can become a base for a useful rehabilitation tool.

  Another medical application is a hybrid training system for orthopaedic open surgery [53]. The authors propose a training simulator for training surgeons in a safe and controlled environment that combines haptic, visual, and audio technologies. The distinctive feature of this research is the usage of patient-specific anatomical 3D models extracted from the patient's computer tomography data. The authors use HoloLens as the basis for their device. The overall performance shows great potential in the use of AR for medical training simulation.

- **Urban development**

  Another application sphere for AR technologies is urban planning and development. Jadid et al. in [129] proposed an optical-tracking-based AR system for participatory urban planning software. The main idea of the system is to allow combining the real world's landscapes with the planned features to help assess the overall impression of the developments in the surroundings. The addition of optical tracking allows for increased quality of immersion as compared with existing systems based on Global Positioning System (GPS) tracking.

- **Entertainment**

  An obvious application area for AR is entertainment.

  Eichhorn, Jadid, et al. in [65] consider the newly defined Superhuman Sports genre. The paper considers a Superhuman Sports ball game where players interact in mid-air with a drone ball. The presented implementation strategies range from smartphone-based AR to HMDs. A suitable pipeline with a tracking algorithm is proposed to allow for real-time 6 DoF (Degree of Freedom) pose estimation of the fast-moving drone. The drone's construction is proposed to be enhanced with an arrangement of LEDs placed on the cage of the drone.

Another entertainment application of AR is the enhancement of museum visits. Plecher, Wandinger, and Klinker in [221] proposed an AR-based approach to present the ancient Greek history and culture to students in a more attractive and enjoyable way. Namely, the main idea is to provide a reconstruction of missing parts of artifacts in the museum. Additionally, the authors propose gamification to further enhance the experience.

Comparison: VR and AR

Virtual and augmented reality technologies have a lot in common: both require a certain level of technical and technological resources, both work with 3D, and both aim at immersion and interactivity.

Nevertheless, the major distinction between the two should be acknowledged: While in AR a user has access to both real and virtual objects, in VR a user is completely shut off from the real world. In other words, the world in VR is entirely synthetic. This difference influences the development strategies for the devices. Namely, the VR devices are usually constructed to be operated within one room, and the AR devices should be highly portable to allow for the full usage of the system's features.

2.2.4. Mixed Reality, MR

The term Mixed Reality (MR) was initially introduced in 1994 by Paul Milgram and Fumio Kishino. Their research was focused on the so-called Reality-Virtuality (RV) Continuum. A schematic from [191] explaining the introduced terms is presented in Figure 2.3.

Figure 2.3.: Reality-Virtuality (RV) Continuum. Source: [191]

The schematic as a whole is the Reality-Virtuality Continuum. The ends of the schematic represent respectfully the Real Environment and the Virtual Environment. Mixed Reality fills the area between the two. So, naturally, MR refers generally to a system that combines real and virtual objects and information [190]. This area includes the already introduced Augmented Reality along with Augmented Virtuality (AV), which can be seen as the opposite of AR, as it augments the virtual environment with real information.

Other researchers offer an alternative point of view on Mixed Reality. In work [256] by Speicher et al., Mixed Realities are considered an immersive technology that provides a new level of interaction among humans, computers, and the environment (see Figure 2.4).

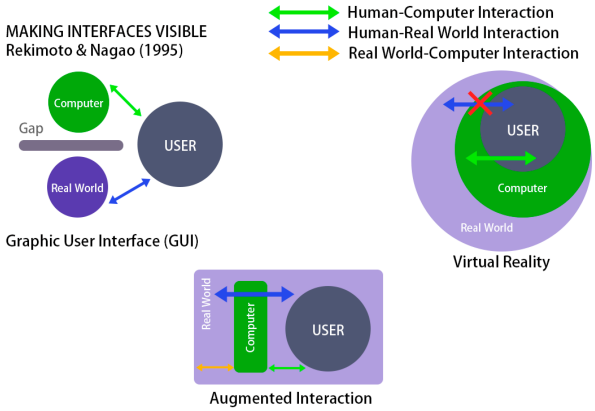


Figure 2.4.: Human-Computer-Environment Interaction. Source: [256]

2.2.5. Extended Reality, XR

Extended Reality (XR) is the term used to describe the full spectrum of Immersive Technologies.

Research [215] by Palmas and Klinker aims at stating a common set of characterizations and concepts to classify and define extended reality training. The authors explain the concept of gamification and immersive learning, stating that the success of extended reality-based training comes from combining computer-generated content, extended reality technologies, environmental and interface fidelity, and a methodological approach customized for the learners' needs.

The main proposed classification concept introduced in the paper is the training domain. The authors single out seven training domains: physical reality, stationary extensions, mobile extensions, augmented reality, mixed reality, virtual reality, virtuality. It should be noted, that the list can be extended with the invention of new technologies. It is also important to add that a single application can be accessible from within multiple domains, however, it should be categorized by the domain that provides the application most advantages.

The continuum for XR training applications allows classifying not only the existing training formats but also the potential ones, by including future emerging technologies into the continuum.

All in all, Immersive Technologies are aimed at enhancing the real world or creating a new one to allow the user to feel new depths of experience.

The next section focuses on the key components of immersive technologies.

2.3. Key Components of Immersive Technologies

Any immersive system that implements XR technology requires specific hardware and software. According to this work's focus, the key components are discussed for the visual

content. Nevertheless, similar techniques are usually employed for other types of information, even though each particular one has its own specifics.

The list of key components of an immersive system suggested by Ivan Sutherland in [261] remains relevant nowadays: An XR system requires software and displays, trackers, and graphic computers. Naturally, as discussed in the previous sections, immersive systems can include other elements that help enhance the experience, but this list represents the crucial ones.

2.3.1. Overview of Components

Figure 2.5 presents a schematic with the key components of an XR system. On the large scale, the system consists of input and output hardware and of processing software and hardware.

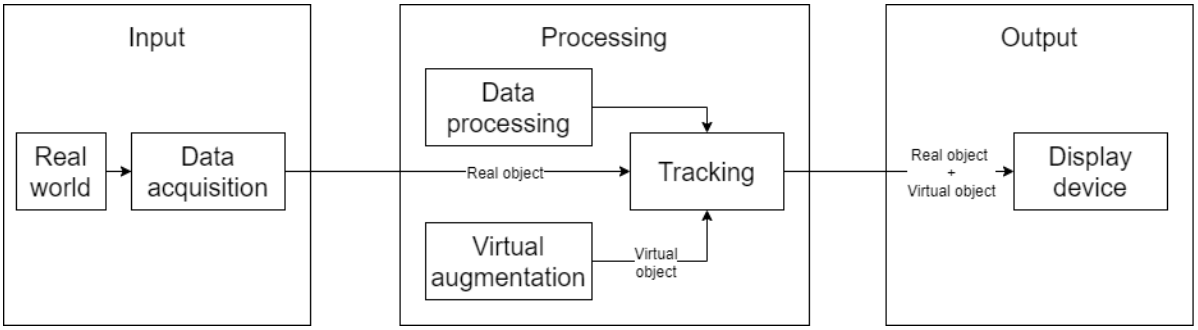


Figure 2.5.: Key components of an Extended Reality system

Input Hardware

The input hardware is used for data acquisition. Different types of input devices can be used at this stage; a review of the major types is provided in Subsection 2.3.2.2.

Interaction should also be taken into consideration at this stage: The input for the interaction (i.e., the user's actions) should be acquired and reported to the processing subsystem.

All of the devices that are used for data acquisition require preliminary calibration. This work has its primary focus on the usage of IMUs (Inertial Measurement Unit). Chapter 4 is therefore devoted to the definition of relevant terms and discussion of various approaches and methods for IMU calibration.

Processing Hardware and Software

The processing subsystem is, naturally, the most complex part of an XR system. At this stage the virtual graphics are generated and introduced to the system and the data acquired on the input stage is processed.

Data processing includes the following procedures:

Filtering

First, the data acquired from the input tracking devices should be filtered: stochastic errors and noise should be eliminated and, for some types of devices, the data should

be improved or enhanced. For example, for camera-acquired images, the brightness, contrast, and color balance can be adjusted at this stage.

Chapter 5 is devoted to filtering techniques.

Registration
Next, the registration should be performed to transform the data collected from different sensors to the same coordinate system, i.e., to find the transformation matrices.

Chapter 6 focuses on registration.

Synchronization
Next, the sensors should be brought to the same time base, i.e., synchronized.

Both hardware and software synchronization are discussed in Chapter 7.

Fusion
Nowadays, XR systems usually include several sensors of different types to improve the accuracy of tracking. So, at this stage, the fusion of the data acquired by multiple sensors should be performed.

Chapter 8 discusses the fusion algorithms.

Pose estimation
Finally, with correctly preprocessed data, pose estimation (or tracking) can be performed.

The objects can be tracked by motion (e.g., with IMUs), by measuring the strength of the signal with the sender-receiver method (e.g., GPS or UWB), by parsing the specific features of an image (e.g., QR-markers or geometric shapes), etc.

Chapter 9 is devoted to the topic of pose estimation.

Virtual graphics rendering
After the data from the input stage are processed, the virtual graphics generated in the processing subsystem can be rendered into correct positions with respect to the user and environment.

Output Hardware
After the collected data from the input stage has been processed, the results should be presented to the user. The most popular choices of the output hardware for XR systems are

Head-mounted Display, HMD
A head-mounted display (HMD) is a visual display that is more or less rigidly attached to the head. Figures 2.6 and 2.7 show examples of different HMDs. HMDs can be further broken down into three types: non-see-through HMDs, video-see-through HMDs, and optical-see-through HMDs. Non-see-through HMDs block out all cues from the real world and provide optimal full immersion conditions for VR.

Optical-see-through HMDs enable computer-generated cues to be overlaid onto the visual field and provide the ideal augmented reality experience. Conveying the ideal

Figure 2.6.: HMD for MR Microsoft HoloLens 2. Source: [188]

augmented reality experience using optical-see-through head-mounted displays is extremely challenging due to various requirements (extremely low latency, extremely accurate tracking, optics, etc.). Due to these challenges, video-see-through HMDs are sometimes used. Video-see-through HMDs are often considered to be augmented virtuality and as such have some advantages and disadvantages of both augmented reality and virtual reality.

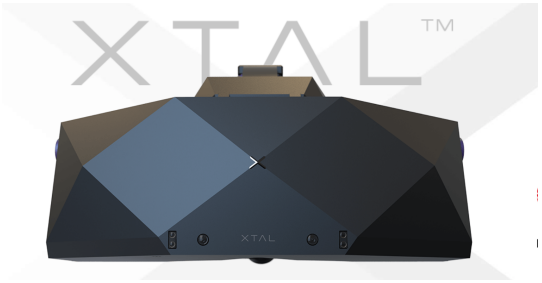


Figure 2.7.: HMD for MR vrgineers XTAL. Source: [278]

Spatial AR
Spacial AR (SAR) augments real world objects and scenes without the use of special displays such as monitors, HMDs, or hand-held devices (see Figure 2.8).

Figure 2.8.: SAR system Extend3D Werklicht Pro L. Source: [72]

SAR makes use of digital projectors to display graphical information onto physical objects (Figure 2.9). The key difference in SAR is that the display is separated from the users of the system. Because the displays are not associated with each user, SAR scales

naturally up to groups of users, thus allowing for collocated collaboration between users.

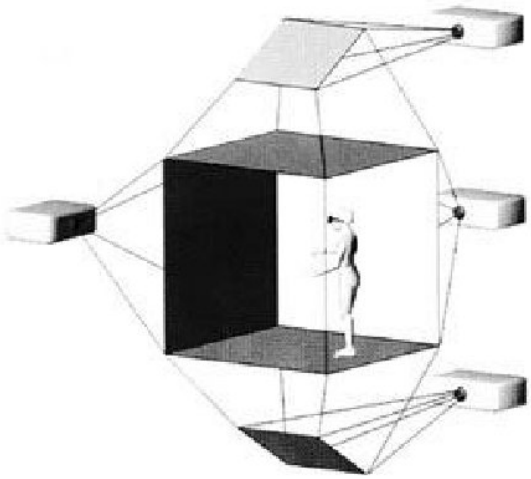


Figure 2.9.: CAVE system. Source: [54]

Hand-held Display
Hand-held displays are output devices that can be held with the hand(s) and do not require precise tracking or alignment with the head/eyes (in fact the head is rarely tracked for hand-held displays). Hand-held augmented reality, also called indirect augmented reality, has recently become popular due to the ease of access and improvements in smartphones/tablets. In addition, system requirements are much less since viewing is indirect—rendering is independent of the user's head and eyes.

2.3.2. Tracking

Extended reality applications require alignment of virtual and real worlds to provide a proper immersive experience to the users. Tracking is the crucial component without which this alignment cannot be achieved.

Tracking refers to determining the pose of an object, i.e. its position and orientation with respect to some coordinate system in real time. Good-quality tracking still remains a major challenge for XR applications, because tracking has to be as precise, accurate, and robust as possible in order to create the illusion that the virtual content is a part of the real world [15].

An accurate tracking system is required for XR systems because even a small tracking error may cause a noticeable misalignment between virtual and real objects [224]. Tracking was and still is the most popular research area in XR [304].

2.3.2.1. Tracker characteristics

Extended reality applications require high-quality tracking for providing an immersive experience. The essential characteristics of tracker in XR systems are as follows:

- **Latency**

  Latency is the delay between the change of the position and orientation of the target being tracked and the report of the change to the computer [22]. If the latency is greater than 50 ms, it will be noticed by the user and the XR scene will not seem realistic.

- **Update rate**

  The update rate is the number of times per second that the tracker device reports data to the computer, and depending on the trackers it can vary between less than 20 and more than 1000 updates per second. The pose update rate must be at least twice the true target motion bandwidth, or an estimator may track an alias of the true motion. Since common arm and head motion bandwidth specifications range from 2 to 20 Hz [80], the sampling rate should ideally be greater than 40 Hz.

- **Accuracy**

  Accuracy is a measure of the error in the position and orientation reported by the tracker. In trackers based on emitters and receivers, it decreases as the user moves farther from the fixed reference point [22].

- **Resolution**

  Resolution is the smallest change in position and orientation that can be detected by the tracker. Particularities depend on the type of tracker used.

The next subsection presents types of sensors commonly used in tracking for XR.

2.3.2.2. Sensor-based tracking

Tracking can be provided by a variety of sensors [285] such as:

- **Mechanical**

  These devices measurement position and orientation by using a direct mechanical connection between a reference point and the target.

  Mechanical trackers have very high accuracy and reliability, low time delay. They are not affected by occlusion or magnetic field disturbance. On the other hand, mechanical trackers are not flexible, have a very limited range, and are limited to one user.

- **Optical**

  Optical or vision-based tracking techniques use image information to track the position and orientation of a camera [291]. There are two main approaches for vision-based tracking: marker-based and marker-less. Also, a distinction is made between "inside-out" (ISO) and "outside-in" (OSI) tracking. Inside-out means that the user's device is tracking itself. In outside-in applications, external devices are installed that support tracking and communicating the tracking object's current position to the user's device [220]. Vision-based tracking is currently the most active area of research in XR [304].

Optical trackers aren't that robust as they can be affected by occlusions. Also, generally, the optical trackers can be heavy and/or bulky. Optical tracking requires high computational power and is usually quite expensive. On the other hand, optical trackers can have very high accuracy, are easy to use, and have a large work range and high speed.

- **Acoustic**

  Acoustic or ultrasonic tracker systems use ultrasound waves for measuring the position and orientation of target objects.

  Ultrasonic trackers are generally cheap and have a small weight and size.

  However, to work properly, a large number of transponders is needed in the known location. The system delay is rather high. As the optical tracker, the acoustic tracker also needs a line of sight to work and is easily disturbed by ultrasonic waves in the environment.

- **Magnetic**

  Magnetic trackers use one fixed transmitter and multiple sensors.

  Magnetic trackers are robust and exact, exhibiting good noise immunity. Most of the trackers are low-cost.

  However, the error increases with larger distances. Also, tracking is easily disturbed by ferromagnetic objects (like furniture) and other magnetic fields.

- **GPS**

  The Global Positioning System (GPS) can be used for outdoor tracking.

  In principle, the satellite network covers any location on Earth. In practice, however, the sensors need direct contact with the satellites, which presents problems in buildings. The accuracy of GPS tracking is limited to about 10-100 m, although some GPS devices can achieve accuracy of about 0.1-1 meters, which is still not enough for most XR applications.

- **Inertial**

  Accelerometers and gyroscopes are inertial sensors used to record user's movements.

  The position is not recorded directly but is obtained by integration of the collected sensor data. The result tends to drift with time and gets imprecise. Also, the measurements prove to be inexact at low speed.

  Inertial sensors can provide tracking both in close vicinity and large field track; they are not affected by vision or magnetic field disturbances. Also, the sensors are of small size and the prices start from low numbers.

All of the listed sensors have their advantages and disadvantages. By combining several sensors of different types into a *hybrid* tracking system, the advantages can be enhanced and

the disadvantages – minimized. Hybrid tracking techniques help enhance the quality of tracking data [293] but at the same time increase the complexity of the tracking process [235].

As explained earlier in this chapter, providing a good quality of tracking is challenging but necessary for producing a system that can guarantee engaging immersion. Therefore, mathematical tools have to be developed and employed that can help achieve this goal. The next section is devoted to a large branch of numerical mathematics and computer science that can provide the necessary methods and approaches: artificial intelligence.

2.4. Artificial Intelligence

> "It may be that machines will do the work that makes life possible and that human beings will do all the other things that make life pleasant and worthwhile."
>
> — *Isaac Asimov*

Artificial Intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving [83].

The goal of AI is to help the user solve posed problems by learning and solving them like a human. Different levels of AI are shown in Figure 2.10.

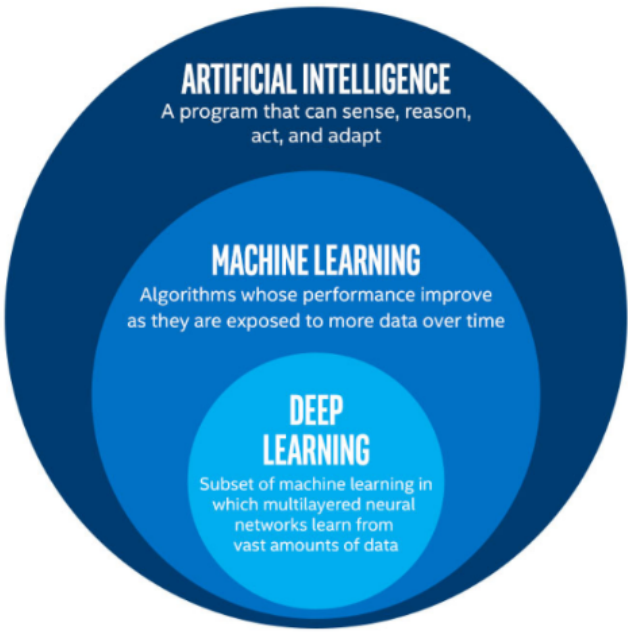


Figure 2.10.: Different levels of artificial intelligence. Source: [210]

The next subsections are devoted to a brief introduction to all these levels.

2.4.1. Machine Learning

Machine Learning (ML) is a level of Artificial Intelligence that is responsible for the abilities of the machines to learn and improve performance without being specifically programmed for a specific task. The goal of ML is the development of mathematical models based on the provided data that is then used by the machine for learning and solving the presented problem.

Machine Learning provides (or helps provide) state-of-the-art solutions for various topics and research areas. The development and availability of computing hardware now allow almost everyone to use the techniques of ML for production, research, and education.

ML provides solutions for problems with various types of data: text, numbers, images, audio, etc, by recognizing and identifying specific features in the raw data and making predictions and conclusions based on these features. Machine Learning algorithms apply statistical analysis to find patterns and generate models that are then used for predictions; the more the data, the better the prediction.

The work "An Inductive Inference Machine" by Solomonoff [254], recognized as the pioneering work on Machine Learning, was published in 1957, bringing the new concept into the research community.

Machine Learning is primarily used for the following three types of problems:

- **Regression**

  Regression problems consist of predicting the next point given a set of data points, e.g., financial audit, weather forecast. Various regression types exist, such as linear, logistic, polynomial, lasso, etc.

- **Classification**

  In classification problems, the goal is to predict the class of the dataset, based on training with multiple labeled classes, e.g., distinguishing between pictures with dogs and cats. Various classification types include binary classification, multi-class, multi-label classification, and imbalanced classification.

- **Clustering**

  Clustering problems consist of finding similarities in the data points and creating clusters of similar points. Many well-known clustering algorithms exist, e.g., K-Means, hierarchical, density-based, etc.

2.4.1.1. Traditional programming Vs Machine learning

A schematic presented in Figure 2.11 shows the principal difference between the traditional programming and machine learning approaches: in traditional programming the user wants to get the "answers" by using the data and a set of rules, which need to be implemented. In

the machine learning approach, the data *and* the answers (labels) are known, and the rules need to be learned.

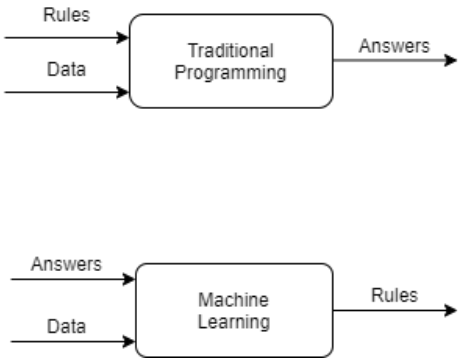


Figure 2.11.: Traditional programming Vs Machine Learning

2.4.1.2. Types of Machine Learning

Figure 2.12 shows the schematic representing the types and subtypes of Machine Learning. Although nowadays a lot of techniques exist that can be considered ML techniques, they can all generally be divided into three large classes: supervised learning, unsupervised learning, and reinforced learning. These three classes are considered in more detail in the following subsections.

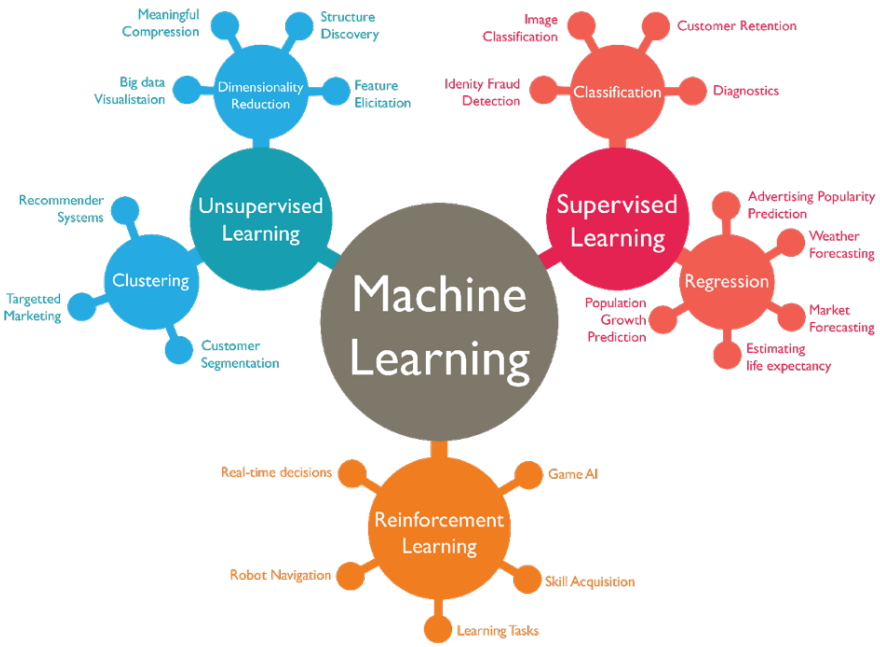


Figure 2.12.: Classification of Machine Learning. Source: [211]

2.4.1.3. Supervised learning

The majority of practical Machine Learning nowadays belongs to the class of supervised learning. The supervised learning problem can be formulated as follows: given input variables x and output variables Y, the algorithm needs to learn the mapping function f such that

$$Y = f(x). \tag{2.1}$$

Then, if the mapping function is identified correctly, the algorithm can predict outputs Y for new (similar) data points x.

Supervised learning is most commonly used for classification and regression problems.

2.4.1.4. Unsupervised learning

As opposed to supervised learning, the dataset for unsupervised learning doesn't have labels for the data points, and the algorithm tries to draw the inference just from the given data, without knowing the output. The unsupervised learning algorithms model the distribution and structure of the data to get more information about the dataset, find insights, and predict the outcome.

Unsupervised learning is often used for clustering and dimensionality reduction. The latter refers to the reduction of feature variables set of the data. Dimensionality reduction can be performed through feature selection or feature extraction. In feature selection, the greedy algorithms select a subset of presented features. Similarly, in feature extraction the goal is to reduce the dimension of the feature set, however, in this case, the resulting features might not be directly corresponding to the original ones (i.e., the new features are generated based on combinations of the original ones).

2.4.1.5. Reinforcement learning

The reinforcement learning strategy is rather different from both supervised and unsupervised learning. The idea of reinforcement learning is to reward "good" behavior and/or penalize "bad" behavior. It is generally achieved by making sequences of decisions through selected agents in a game-like scenario where the agent is rewarded or penalized depending on the taken actions. The rewards and penalties are defined by the model's definition. The applications of reinforced learning include computer chess programs, game AI, real-time decision-makers, and so on. Figure 2.13 shows a schematic of a basic reinforcement learning system.

The key components of a reinforcement learning system include

- Environment

  The real (physical) world where the agent operates.

- State

  The current state of the agent.

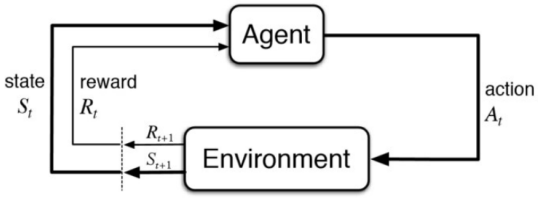


Figure 2.13.: Reinforcement Learning architecture. Source: [32]

- Reward/penalty

  Feedback from the environment.

- Policy

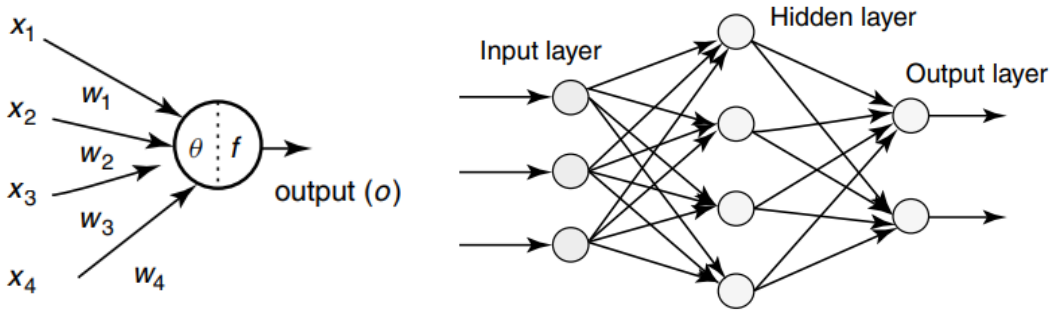
  A function that maps the state of the agent to its actions.

- Value

  Potential reward that the agent can receive by taking action in current state.

2.4.2. Neural Network

Neural Network (NN), or, more correctly, *Artificial* Neural Network (ANN), is a mathematical model that was originally inspired by biological (genuine, natural) neural networks [194]. Neural networks are often used nowadays to solve complex problems without explicit instructions. Biological neural networks are webs of interconnected neurons, and a neuron is one biological cell that can process information and pass it to the next cell. An artificial neuron is an artificially engineered version of a biological neuron, and an ANN is a connected web of artificial neurons.



(a) Artificial neuron

(b) Standard multi-layer NN

Figure 2.14.: Basic NN components. Source: [1]

Neural networks include three major components: an input layer, a hidden layer, and an output layer (see Figure 2.14(b)). If the data pass from the input layer to the output layer (via

the hidden layer(s) in between) in a strictly feed-forward fashion, the neural network is called a feed-forward network [1].

Inputs $x_1, \ldots, x_n$ are given to a single neuron of the network (see Figure 2.14(a)), and based on them the neuron generates an output signal flow O. The relationship is described by

$$O = f(\text{net}) = f\left(\sum_{j=1}^{n} x_j w_j\right), \tag{2.2}$$

where w_j are the weights of the nodes, function $f(\cdot)$ is the activation transfer function, and the net is represented by a scalar product of the weights and inputs.

The output value O is computed as

$$O = f(\text{net}) = \begin{cases} 1, & \text{if } x \geq \theta \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

The neurons of this type are called Linear Threshold Units, and parameter θ is the threshold level. The weights can be set explicitly based on additional knowledge about the data or can be found/updated by the neural network itself, using the back-propagation techniques, delta rules, or perceptron rules [1].

The usage of ANNs offers various advantages: the technique allows solving complex problems by identifying non-trivial patterns, the computations can be done in a sequential centralized *or* distributed fashion, depending on the application and the available resources.

The disadvantages of NNs are generally the black-box principle (the processing of the hidden layers is not transparent), the networks are prone to overfitting, the quality of the output is hard to predict, and the required computational power is usually large.

Nevertheless, the use of NNs provides new insights, and with the development of computational capability more and more applications rely on the use of artificial neural networks.

2.4.2.1. Activation functions

Activation Functions are the equations for the neurons that determine the output of the network. Depending on the definition, the activation functions normalize the output of the neurons into a range of $[0; 1]$ or $[-1; 1]$. Thus, activation functions determine if the particular neuron is activated or not.

Various activation functions exist, that are specifically used for different types of problems and different types of datasets. A short review of the most commonly used functions is presented below.

- **Binary Step**

  Binary step functions are purely based on a threshold: A neuron is activated if the inputs are higher or lower than a certain threshold. If the neuron is activated, the values

pass without any change to the next steps. An example of a binary step function with a 0-threshold is as follows:

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x \leq 0 \end{cases} \tag{2.4}$$

Figure 2.15 shows the plot of this activation function.

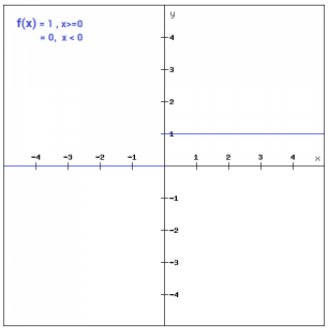


Figure 2.15.: Binary Step activation function. Source: [104]

The use of this activation function is limited for back-propagation because it provides a zero gradient, which means that the weights and biases are not updating during back-propagation.

- **Linear**

  Linear activation functions prevent the degeneracy of the gradient. The equation describing this function is as follows:

$$f(x) = ax. \tag{2.5}$$

Figure 2.16 shows the plot of a linear activation function.

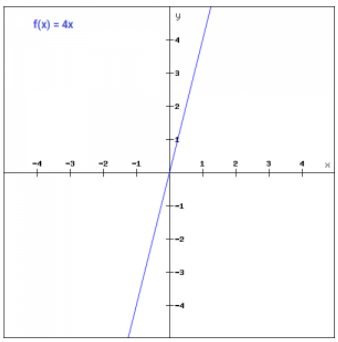


Figure 2.16.: Linear activation function. Source [104]

Although the gradient does not become zero in this case, it becomes a constant, i.e., in every step the gradient is the same. Nevertheless, the weights and biases will be updated in back-propagation.

- **Sigmoid**

  The sigmoid activation function is a non-linear activation function that is most commonly used for binary classification problems. The equation describing this activation function is

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{2.6}$$

  The plot of the sigmoid activation function is given in Figure 2.17.

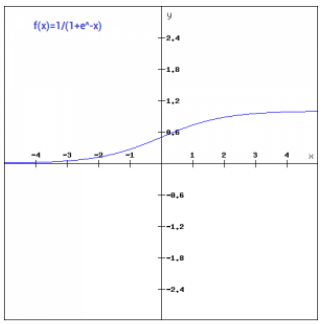


Figure 2.17.: Sigmoid activation function. Source: [104]

  The outputs of this function lie between 0 and 1; the function is not symmetric around zero. As the gradient approaches zero, the network doesn't learn. As can be seen from the plot, for very high or very low values the prediction changes slowly, leading to gradient degeneracy. Moreover, the calculation of the function is computationally expensive.

- **Softmax**

  The softmax activation function is a combination of multiple sigmoids. The softmax activation function is used for multi-class classification because it can show the probability of a data point belonging to multiple classes rather than just putting a point into a single class. The mathematical expression of softmax is as follows:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \tag{2.7}$$

- **Tanh (Hyperbolic Tangent Function)**

  Hyperbolic tangent function (usually simply denoted as tanh) is often used in a specific class of neural networks, namely, in recurrent networks. This activation function is

similar to sigmoid, except it is symmetric around zero. The formula is as follows:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{2.8}$$

Figure 2.18 shows the plot of the tanh activation function

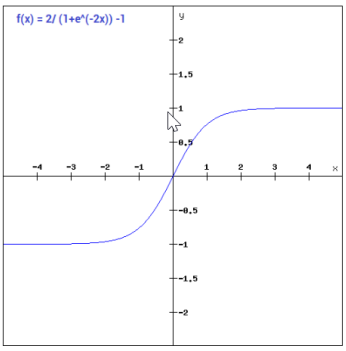


Figure 2.18.: Tanh activation function. Source: [104]

The properties of the tanh function are similar to the sigmoid function, except the output values lie in $[-1; 1]$ and the gradient is steeper because of that difference.

- **ReLU (Rectified Linear Unit)**

  The Rectified Linear Unit (ReLU) activation function is a non-linear function that behaves as the linear function for positive inputs but deactivates the neuron for negative ones:

$$f(x) = \max(0, x) \tag{2.9}$$

Figure 2.19 shows the plot of this function.

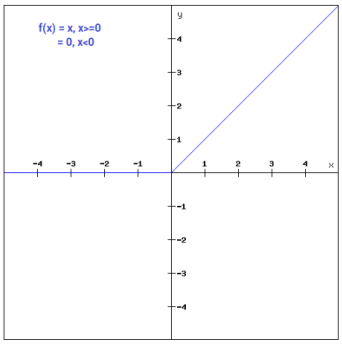


Figure 2.19.: ReLU activation function. Source: [104]

Because not all of the neurons stay activated at all times, this function is more computationally efficient than sigmoid or tanh. However, because of the same reason, during

back-propagation some neurons do not get an update of weights and biases, leading to the emergence of "dead" neurons.

- **Swish**

  This activation function has been developed by Google Brain Team in 2017 [228]. Swish activation function is defined as

  $$f(x) = x \cdot sigmoid(x) = \frac{x}{1 + e^{-x}}. \tag{2.10}$$

  The function is differentiable at any point, which helps the optimization process.

- **Leaky ReLU**

  Leaky ReLU is a modification of the ReLU function that addresses the dead neuron problem. Leaky ReLU defines the output for negative inputs as a linear function with a very small slope:

  $$f(x) = \begin{cases} 0.01x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0. \end{cases} \tag{2.11}$$

  Figure 2.20 shows the plot of the Leaky ReLU activation function.

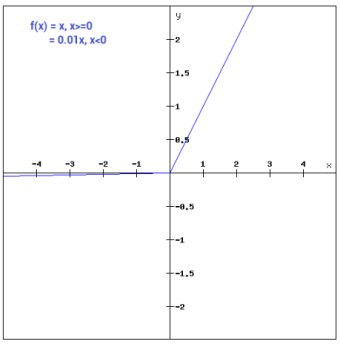


Figure 2.20.: Leaky ReLU activation function. Source: [104]

The gradient does not degenerate for negative values, thus causing no dead-neuron problems in back-propagation.

Table 2.2 shows a comparison between the described functions.

All of the presented activation functions have their use cases; depending on specific properties of the problem one or the other (or their combination) can be used. The choice of the activation function plays a huge role in the performance of a neural network.

Table 2.2.: Simple comparison of the Activation Functions

Function	**Range**	**Use Case**	**Problem**
Binary Step	0 to 1	Binary Classification	Weights and biases aren't updated
Linear	$\propto x$	Simple Regression	Updating factor is constant
Sigmoid	0 to 1	Binary Classification	Vanishing Gradient
Softmax	0 to 1	Multi-Class Classification	Always activates the neuron
Tanh	-1 to 1	Regression, NLP	Value saturation
ReLU	0 to x	Regression	Dead neurons
Leaky ReLU	$\propto \alpha x$ to x	Regression	Not consistent

2.4.2.2. Loss functions

Loss functions provide a metric that shows the quality of the model for a given dataset: they calculate the loss given the input and the output and then can reward or penalize particular predictions.

The two main types of loss functions are regression loss and classification loss.

Regression Loss: The most commonly used regression loss functions are Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}_i| \tag{2.12}$$

and Mean Square Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}_i|^2. \tag{2.13}$$

Both MAE and MSE have the range from 0 to ∞ and they are both negatively oriented, i.e., lower values mean better quality of prediction.

MAE computes the difference between the actual value and the predicted values in a provided test dataset. If the direction is considered instead of the absolute values, MAE becomes Mean Bias Error (MBE), an estimator that measures the average model bias.

MSE computes the squares of all the differences before computing the average, thus the large differences are penalized on a larger scale. Sometimes a Root Mean Square Error is used, when the square root is taken from the resulting value (reducing the penalties for large errors).

Classification Loss: Two of the most commonly used loss functions for classification problems are Multi Class SVM Loss and Cross-Entropy Loss.

The mathematical expression for SVM (Support Vector Machine) loss is given as:

$$SVMLoss = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1). \tag{2.14}$$

The idea of SVM loss, also known as Hinge Loss, is to give a higher score to a correctly predicted class as compared with the sum of all of the incorrectly predicted classes. The SVM loss function is convex, which allows for using convex optimization techniques.

Another choice of a classification loss function is the Cross-Entropy Loss also know as Logarithmic Loss or Log Loss. The expression of Cross-Entropy Loss is given by

$$LogLoss = -(y_i \log(\bar{y}_i) + (1 - y_i) \log(1 - \bar{y}_i)), \tag{2.15}$$

where y_i is the actual label and $\bar{y}_i$ is the predicted value.

Cross-Entropy loss can be used both for binary and multi-class classification problems. The loss increases when the probability of being in a particular class deviates from the actual label.

2.4.2.3. Hyperparameters

Hyperparameters are (tunable) parameters that control the learning process (as opposed to other parameters, like node weights, that are derived via training).

One of the most important hyperparameters in the learning rate [98].

Learning rate Learning rate defines how much change is needed in the model after each iteration and update of the weights. The choice of the learning rate for a model is on the one hand very important, and on the other hand very difficult. If the value of the learning rate is too small, the training process might be slow or even get stuck. If it is too large, the weights might be updated too fast causing instabilities in the training process.

Figure 2.21 shows the effects of learning rate values on the training process.

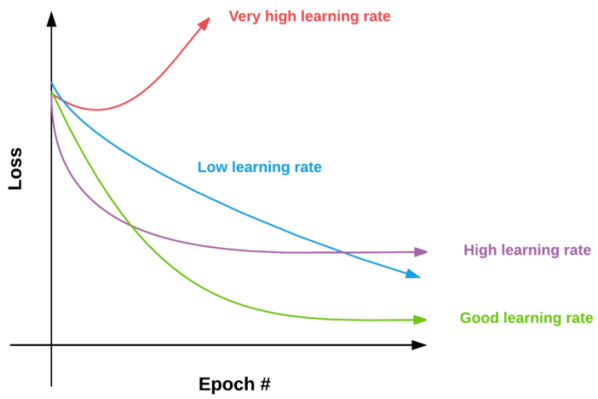


Figure 2.21.: The effect of learning rate. Source: [140]

2.4.2.4. Optimization algorithms

Optimization algorithms are used in a tandem with back-propagation techniques for updating weights and biases in the network in order to reduce the error. The optimization is generally

focused on obtaining a value of θ that reduces the loss function $C(\theta)$.

One of the most popular algorithms in neural networks is the gradient descent algorithm. Gradient describes the direction of the increase of the function. The minimum is found by following the anti-gradient:

$$\theta = \theta - \eta \nabla C(\theta; x, y) \tag{2.16}$$

where η is the learning rate and $\nabla C(\theta; x, y)$ is the gradient of the weight parameter θ.

The optimization algorithms are usually classified as Constant Learning Algorithms and Adaptive Learning Algorithms. In the former, the hyperparameters are defined apriori based on the model, data type, and any knowledge of them that the user has. One of the most commonly used methods of this class is Stochastic Gradient Descent (SGD). Figure 2.22 illustrates the iterations of SGD.

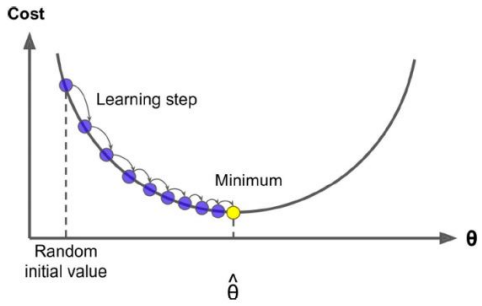


Figure 2.22.: Stochastic Gradient Descent process

There are other variants of gradient descent algorithms such as e.g. Batch Gradient Descent and Mini Batch Gradient Descent.

The second class of optimization algorithms, adaptive learning algorithms, allows for automatic tuning of the parameters [2]. Some adaptive optimization algorithms are Adam, Nadam, AdaGrad, AdaDelta, RMSProp, etc. Arguably the most popular optimization algorithm from this group is Adam. This algorithm has a straightforward implementation, is memory- and computationally efficient, and is suitable for large datasets.

2.4.2.5. Regularization

When working with neural networks, users often face the problem of overfitting,that is when the algorithm shows good performance on training sets but performs poorly on test data. Regularization is the group of techniques that helps prevent or avoid overfitting.

Some of the commonly used regularization techniques are discussed below.

- **L1 and L2 regularization**: L1 and L2 regularization are arguably the most common regularization techniques. Both L1 and L2 modify the cost function by adding a regularization term to the loss function. As a result, the value of the weight matrices decreases leading to simpler models and consequently reducing overfitting.

The equations for the L1 and L2 regularized cost functions are as follows:

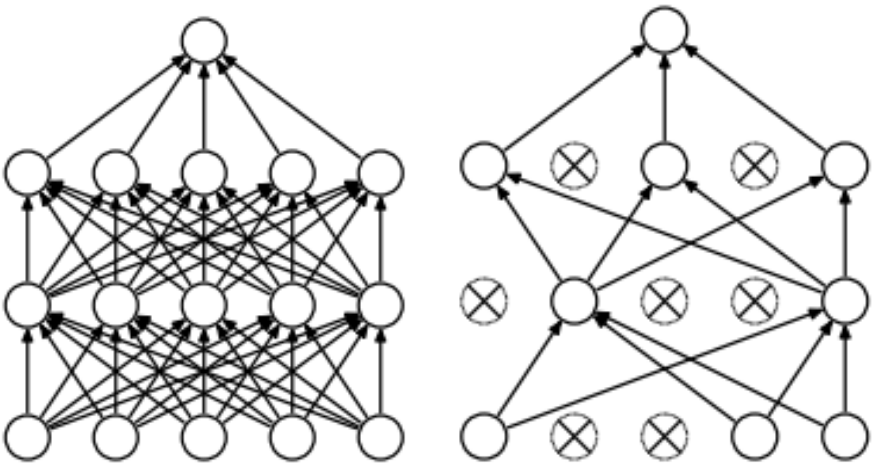
$$\text{cost function } = \text{Loss} + \frac{\lambda}{2m} * \|w\|_1, \tag{2.19}$$

$$\text{cost function } = \text{Loss} + \frac{\lambda}{2m} * \|w\|_2^2, \tag{2.20}$$

where $\|w\|_1 = \sum_{i=1}^{n} |w_i|$ and $\|w\|_2 = \sqrt{\sum_{i=1}^{n} |w_i|^2}$ are the 1-norm and 2-norm of the vector respectively.

L1 regularization is also known as the Lasso algorithm. It is known to generally provide sparse feature vectors [268]. L1 and L2 regularizations are not suitable where only a few features are relevant for the optimal prediction [199].

- **Dropout**: Dropout is another technique that helps with overfitting. Figure 2.23 illustrates the basic idea of dropout in a standard neural network.



(a) A chunk of NN before dropout (b) After applying dropout

Figure 2.23.: Schematic of dropout idea in a neural network. Source: [257]

In Figure 2.23(a), a standard neural network with 2 hidden layers is presented, where all of the neurons are activated. Figure 2.23(b) shows the same NN after applying dropout: the network becomes thinned. The crossed nodes have been dropped and deactivated. When dropout is applied to a network, it can be said that the network is being trained with a collection of 2^n thinned networks where the nodes share weights extensively, with n being the total number of nodes in the network.

So, dropout is a regularization method where a large number of networks run in parallel, and some layer outputs are randomly ignored, i.e., the outputs after a dropout layer are randomly subsampled; therefore, it is better to use dropouts in a wide and deep

network with a large number of nodes [257]. The training process gets noisy because of using dropouts and it enforces the nodes to take more or less responsibility for the inputs probabilistically.

- **Data augmentation**: Overfitting also happens when there is less data for training. To get more data especially in image classification problem data augmentation is another technique that augments the samples in the training data. For example, to generate more images, can be just flip the image horizontally and vertically.

2.4.3. Deep Learning

Deep Learning (DL) is a subclass of Machine Learning that considers more powerful methods generally able to solve more complex problems. Deep learning models rely on the availability of powerful CPUs (Central Processing Unit) and GPUs (Graphics Processing Unit) and the ever development of the processing power to use the models with a large number of underlying layers. While Machine Learning algorithms sometimes require interventions from the user in order to properly perform feature engineering, the algorithms of Deep Learning are able to map complex functions from the input to the output without manually tackling the features [28].

Deep learning is considered to be one of the most robust branches of machine learning and artificial intelligence. As philosopher and polymath Nick Bostrom said, "Machine intelligence is the last invention that humanity will ever need to make" [36]; deep learning is considered to be at the pinnacle of human innovations that will potentially lead to the ultimate artificial intelligence.

Deep Learning is also sometimes called Deep Structured Learning or Hierarchical Learning. Basically, the mechanism of Deep Learning is a neural network with multiple hidden layers, i.e., a Deep Neural Network (DNN). Sometimes DNNs are called Deep Learning (Neural) Networks. Figure 2.24 shows the schematic of a regular ANN and a Deep NN.

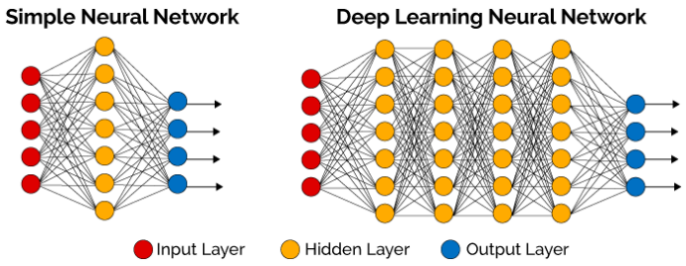


Figure 2.24.: The difference between a standard neural network and a deep neural network. Source: [273]

The hierarchy of layers allows the development of complex patterns by gradually using simpler ones. The network strives to identify high-level features seeing low-level properties. Deep learning is successfully used in image and text classification, natural language process-

ing, autonomous driving, weather and exchange rate forecasts, human gesture prediction, etc.

Two classes of DNN are most commonly used: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). These two classes are considered in more detail further.

2.4.3.1. Recurrent Neural Networks (RNN)

A standard ANN has no memory of the previous states except the last one: the ANN starts from the state where it was trained. Because of that, after the initial training is performed, there is hardly any evolution in the network. The ANN only learns from the data currently presented to the network, which is only useful for independent inputs. Since it is in general not true for every input sequence, a type of neural network was introduced that helps overcome this disadvantage – Recurrent Neural Networks.

The major distinguishing feature of recurrent neural networks is that an RNN has loops in itself (see Figure 2.25), allowing the neurons to remember information [207]. All the inputs are related to each other, as opposed to the performance of other types of ANNs, and the outputs don't only depend on the present inputs, but also on previous states.

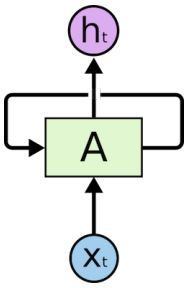


Figure 2.25.: Basic RNN Unit. Source: [207]

In Figure 2.25, representing a basic unit of RNN consisting of one neuron, the network is given an input X_t and it gives the output h_t. The loop forces the data to be passed through several steps in the network. An unfolded version is presented in Figure 2.26: the RNN is basically a copy of the same ANN connected by chains, and each copy passes information to its successor [207]. In this way, the state of the unit at time t is an input at time $t + 1$

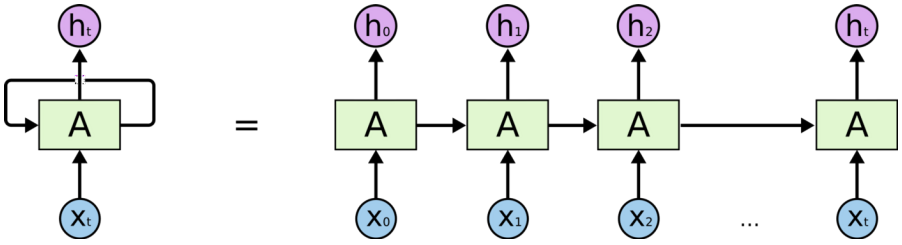


Figure 2.26.: Basic RNN Unit (Unfolded). Source: [207]

The above figure describes itself, the state of the unit at time *t* is also an input for the next step at time *t+1*. In theory, this should mean that all the previous information is available at the new step; in practice, however, if there are many steps involved, the network exhibits vanishing gradient behavior [29, 207]. This long-term dependency problem is illustrated in Figure 2.27.

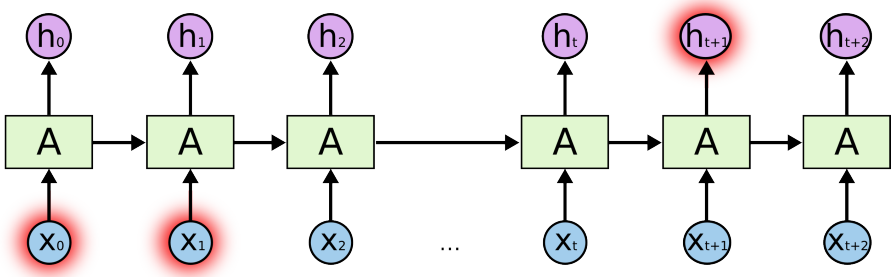


Figure 2.27.: Long-term dependency in RNN. Source: [207]

So, as the basic version of RNN has this type of long-term dependency, different modifications have been suggested to overcome this problem. Long-Short Term Memory (LSTM), Bidirectional LSTM, and Gated Recurrent RNNs are considered in more detail next.

Long-Short Term Memory (LSTM) Long-Short Term Memory, also known as LSTM, is one of the most popular choices of RNNs when long-term dependencies are considered. That means that it is a suitable tool for sequence learning that can eliminate the vanishing gradient problem. LSTMs were first proposed by Hochreiter and Schmidhuber [118] in 1997 and they still remain popular and actual nowadays.

As stated above, the major distinguishing characteristic of LSTMs is that they remember selected pieces of information for a long period of time. While typically RNNs have chain-type formations of modules repeating themselves, in the most basic case an RNN can consist of a single tanh layer (see Figure 2.28).

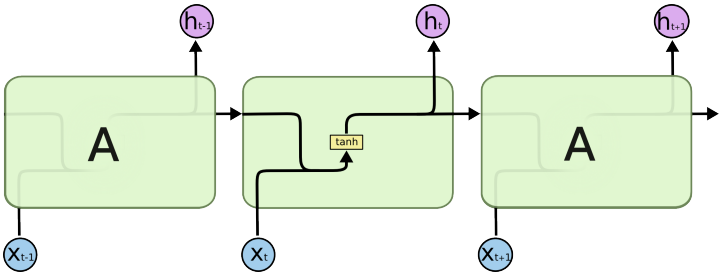


Figure 2.28.: Standard RNN containing a single layer. Source: [207]

In LSTM, sigmoid activation functions are used along with tanh. As explained in the relevant subsection earlier, tanh keeps the values in the range $[-1; 1]$, while sigmoid has a range of $[0; 1]$. The strategy is to multiply the values by 0 if they need to be forgotten and by

1 otherwise, thus learning which data is important and need to be kept and which can be discarded. To implement this, LSTMs have gates of three types: input, forget, and output gates [94]. The cell architecture is presented in Figure 2.29.

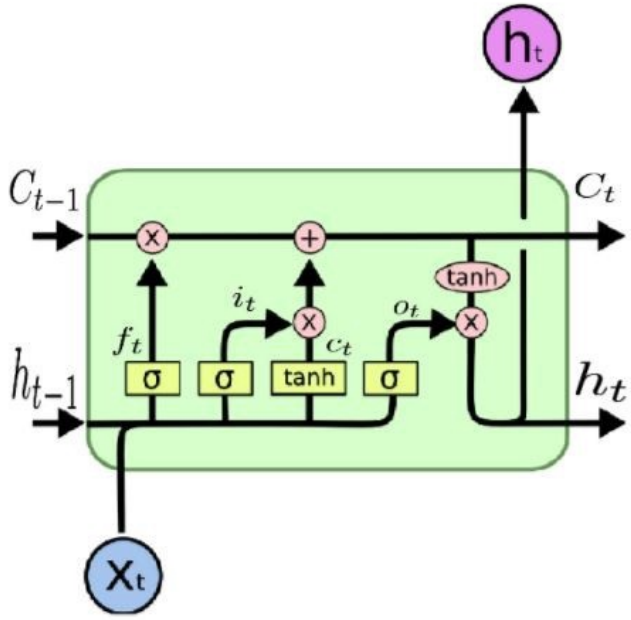


Figure 2.29.: LSTM cell architecture. Source: [207]

The gate equations for LSTMs are as follows. For the input gate:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i), \tag{2.21}$$

for the forget date:

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f), \tag{2.22}$$

and for the output gate:

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o), \tag{2.23}$$

where $\sigma(\cdot)$ is the sigmoid function, $w_\cdot$ is the weight vector, h_{t-1} is the output of the previous LSTM block at time step $t-1$, x_t is the input at the current time step, and b is the bias.

The equations describing the cell state, the candidate cell state, and the final output of the current cell are as follows. The cell state:

$$\bar{c}_t = \tanh(w_c[h_{t-1}, x_t] + b_c), \tag{2.24}$$

the candidate cell state:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \bar{c}_t, \tag{2.25}$$

and the final output:

$$h_t = o_t \cdot \tanh(c_t). \tag{2.26}$$

This describes the standard (or basic) LSTM. A range of modifications exists, e.g., an LSTM with peephole connection [93] where all the gate layers have reading access to the current state. Other notable modifications include Depth Gated RNNs [292] and Clockwork RNNs [151].

Bi-Directional LSTM The next important RNN type is Bi-Directional LSTM or, shortly, Bi-LSTM. In a standard Bi-RNN (see Figure 2.30), recurrent layers are duplicated and get the inputs in the direct and reversed order (thus the name, bidirectional). With this, Bi-RNNs see information "from the future" [244]. Bi-LSTMs are very popular in speech recognition problems, and, although not the universal tool, they still have an edge over the unidirectional LSTMs [100].

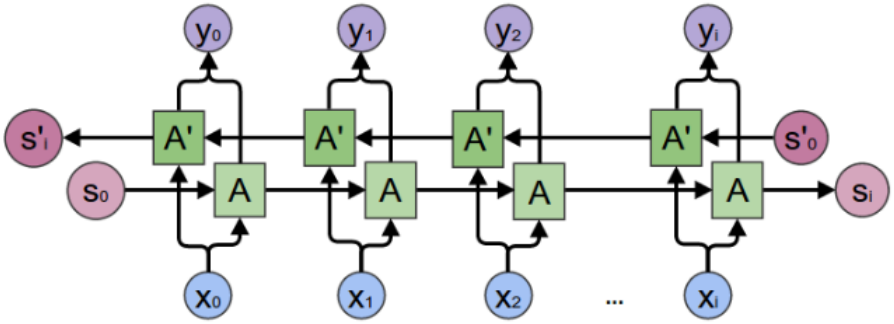


Figure 2.30.: Standard Bi-LSTM architecture. Source: [4]

Although having a lot of advantages, Bi-LSTMs require a lot of trainable parameters, i.e., they are very resource-demanding.

Gated Recurrent Unit (GRU) Gated Recurrent Unit or GRU is a simpler version of LSTM. LSTMs have 3 gates, and GRUs only have two, meaning that they are less computationally expensive, using less memory and training in less time. The architecture of a GRU is shown in Figure 2.31.

The equations describing a GRU are as follows. The update gate:

$$z = \sigma(x_t U^z + s_{t-1} W^z), \tag{2.27}$$

the reset gate

$$r = \sigma(x_t U^z + s_{t-1} W^t). \tag{2.28}$$

The output is

$$h = \tanh(x_t U^h + (s_{t-1} \cdot r) W^h), \tag{2.29}$$

and finally

$$s_t = (1 - z) \cdot h + z \cdot s_{t-1}. \tag{2.30}$$

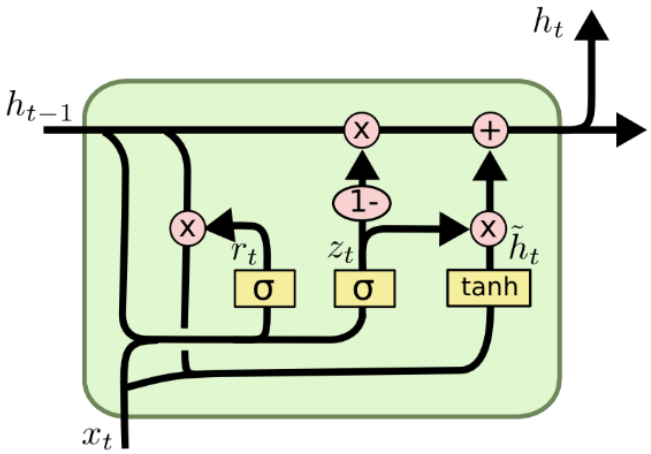


Figure 2.31.: Standard GRU architecture. Source: [207]

The information flow of GRU is similar to the one of LSTM, except it has no memory unit and exposes all the content in a cell. The update gate z corresponds to the input and forget gates of LSTM, deciding which information to add and which to forget. The reset gate r is applied to the previous hidden state.

2.4.3.2. Convolutional Neural Networks (CNN)

Convolutional Neural Network (CNN) is a DNN that uses convolution instead of general matrix multiplications at least in one of the layers [98]. It allows using fewer parameters for the computations comparable with fully connected NNs. CNNs are inspired by the processes in the human brain, with the connectivity pattern of CNN resembling the connectivity pattern between neurons of the brain. CNNs are commonly used for the classification of image data, but lately, they have also been shown to give promising results on time series data [106].

Convolution is a mathematical operation on two functions, f and g, that produces a third function, $[f * g]$, describing how the shape of one function is modified by the other. The following equation shows the convolution of two functions over a finite time range:

$$[f * g](t) \stackrel{\text{def}}{=} \int_0^t f(\tau) g(t-\tau) d\tau. \tag{2.31}$$

Figure 2.32 shows the operation of convolution. Here the green curve is the convolution of the blue and red curves as a function of time; the vertical green line indicates the time. The grey region is the product $g(t-\tau)f(\tau)$ as a function of t, and the area of that region is the convolution.

The convolution operation for finite domains, e.g. images, can be written as

$$s[i,j] = (I \times K)[i,j] = \sum_m \sum_n I[m,n] K[i-m, j-n], \tag{2.32}$$

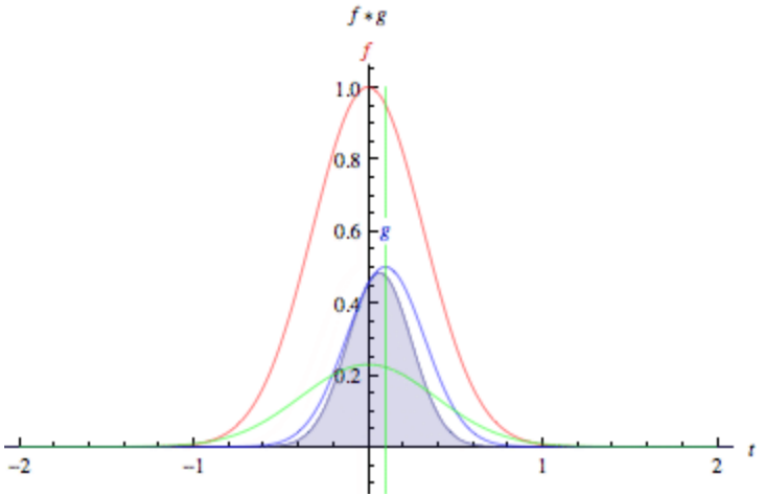


Figure 2.32.: Visualization of convolution. Source: [284]

where I is the input image with dimensions $m \times n$, K is the kernel, and s is the resulting output at indices i, j.

CNN consists of several blocks (or layers), each of which performs a specific operation on the preceding layer's output. Figure 2.33 shows different potential layers of a CNN.

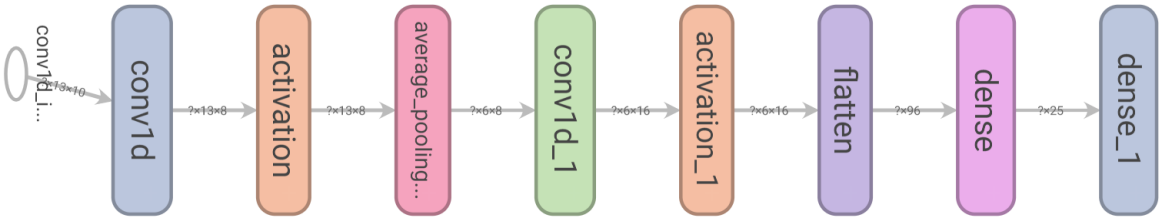


Figure 2.33.: A typical CNN architecture representing different layers of CNN

The layers are shortly explained next.

- **Convolution layer**
  The convolution layer detects local features in the dataset obtained from the previous layer and maps them to a feature map. A filter or a kernel is passed over the matrix of the dataset applying convolution operation on the pixels with weights defined in the filter. Figure 2.34 visualizes the convolution operation.

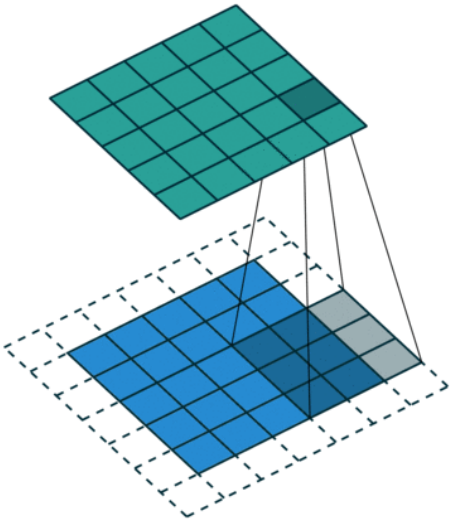


Figure 2.34.: Visualization of convolution operation of CNN layer. Source: [245]

- **Activation layer**
  The activation layer, or non-linearity layer, allows for back-propagation in the CNN. Activation functions, presented in the corresponding section earlier, are used in this layer.

- **Pooling layer**
  The pooling layer performs down-sampling to reduce the matrix size, minimizing the overfitting effects in the meantime. The use of pooling layer allows for faster network training and more precise feature extraction. The most common methods are maximum pooling and average pooling. The mechanism of max-pooling is visualized in Figure 2.35.

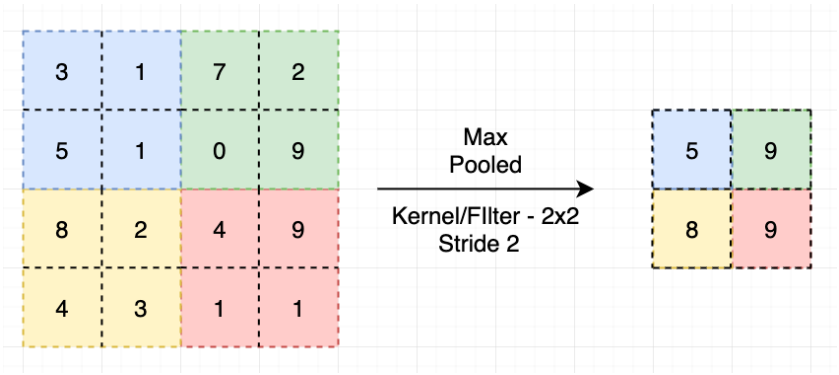


Figure 2.35.: Visualization of max-pooling. Source: [230]

- **Fully connected layer**
  The fully connected layer or dense layer maps the output from the previous block into a class of probability distributions.

This chapter was devoted to the review of the relevant background, aimed to provide the necessary knowledge and information from related fields that allow for better understanding and perception of the following research.

"Don't fight forces, use them."

— Buckminster Fuller

3. Inertial Measurement Unit

An Inertial Measurement Unit (IMU) is an electronic device used for the detection of the current object orientation based on the principles of inertia of mass. The current displacement and/or orientation is obtained by the tracking systems from the measured inertial forces (relative to a reference inertial space). These forces can be used to calculate the linear acceleration and angular velocity of the object by the use of Newton's laws. Knowing the initial absolute position of the object, by calculating the relative changes of these values, the actual current position or orientation can be determined.

Due to its working principle, inertial tracking can be used both indoors and outdoors [155], which allows for a wider range of use of the technology. One of the most recognized applications is Inertial Navigation Systems (INS). Although the technology was originally developed for rockets [97], it is now also successfully used in the shipping industry [51]. INSs are navigational systems that calculate the position (normally latitude and longitude) and also usually heading, pitch, and roll and sometimes sway and surge of a vehicle. The position of the vehicle can be computed in relative or absolute terms. Generally, because of the causes described further in this chapter, an additional aiding device is used for assisting the navigation. For example, for surface applications, GPS is often employed, while for subsea applications specific acoustic positioning systems like USBL (Ultra Short Base Line), SBL (Short Base Line), or LBL (Long Base Line) are used. In an aided tracking system, the inertial subsystem collects short-term data and the satellite/acoustic/other additional subsystem corrects the accumulated errors.

Another popular application field of inertial tracking nowadays is MR systems. The devices generally used for INSs are not suitable for MR applications due to their size and cost. An application for MR in mobile devices only became possible with the evolution of MEMS (Micro-electromechanical System) inertia sensors, which allow 6 Degree of Freedom (6 DoF) tracking while being exceptionally compact in size. For 6 DoF inertial tracking, the circuit must include at least a three-axis accelerometer and a three-axis gyroscope, so that the angular rate and acceleration around each axis can be measured.

As mentioned above, an IMU usually includes an accelerometer, a device that measures linear acceleration, and a gyroscope that measures angular velocity. Additionally, sensors of different types can be mounted inside of an IMU, such as a magnetometer, a barometer, or a thermometer. High-end IMUs come with a case (an enclosure) and are also typically equipped with digital signal processing hardware/software, communication hardware/software such as a micro-USB port or a Bluetooth adapter. Low-cost IMUs only include electrical components, which need to be connected to a microcontroller.

3.1. Principal IMU Components

3.1.1. Accelerometer

An accelerometer is a device that measures the linear acceleration of a locatable in one direction, i.e., the measurement state space is a one-dimensional second derivative of the absolute position. To get a three-dimensional state space, three devices can be combined into a system: Three accelerometers are mounted at right angles to each other, so that acceleration can be measured independently in three axes X, Y, and Z. Because accelerometers are relatively cheap they can be deployed in large quantities. On the other hand, to estimate the absolute position of the object the collected acceleration should be twice integrated that inadvertently leads to a large drift of the resulting position estimates. However, accelerometers usually provide a very high update rate and proved useful for many applications.

Accelerometers can be implemented based on various technologies. The most popular ones are mechanical, capacitive, electrical (micro-electromechanical), piezoelectric, con Hall-effect based, and thermal [280].

Mechanical accelerometers are the most basic type that uses the implementation of the original idea associated with acceleration measuring. They consist of a small object with some significant mass attached to a spring, and that spring subsystem is encased in a lightbox. When the system accelerates, the outer box moves immediately while the mass stays in place, and the attached string stretches with a force. By measuring the length of the stretched spring, acceleration can be measured.

Capacitive accelerometers consist of two metal plates, one of which is attached to the box's wall, and the other – to the mass. If the motion of the mass changes the distance between the plates, the corresponding change in capacitance can be converted to measure the acting force and, therefore, acceleration.

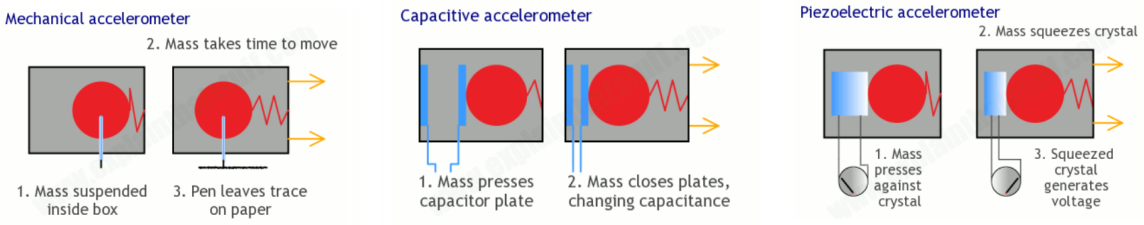


Figure 3.1.: Schematics of accelerometers. Source: [288]

Electrical accelerometers combine the idea of the mechanical device with electrical sensor units. The whole system represents an electrical circuit, and the transformed mass and box objects work together as a capacitor. That technology allows for a faster and more precise measurement of acceleration. Because the main force in these devices is electrical and not mechanical, it is possible to significantly reduce the size of the device, so that the final product can be used as a microchip [145].

Piezoelectric accelerometers have a hard inner beam that is constantly pressing into a piezo

crystal. The crystal then generates an electric current, and based on the voltage acceleration can be measured.

Hall-effect is the effect of voltage difference production across an electrical conductor when exposed to a magnetic field. So accelerometers that use this effect measure the data by sensing changes in the magnetic field.

Thermal accelerometers (also sometimes called convective accelerometers) contain a small heater that heats an air (or gas or fluid) bubble. A temperature sensor then measures the temperature and when the system accelerates the colder substance pushes the heated bubble. The change in the temperature measurement is then converted into acceleration.

An important subclass of accelerometers is pedometers. Pedometers have a binary output: once the acceleration surpasses a specified threshold the output becomes one. These devices are usually implemented as mercury or ball switches. Their main usage is step counting.

3.1.2. Gyroscope

A gyroscope is a device that measures angular velocity. The underlying technology is based on the conservation law for angular momentum. Alternatively, some gyroscopes use the precession effect and measure the torque on the rotation axis. Both approaches only allow for 1D measurements, therefore, as with accelerometers, gyroscopes are usually used in systems of three to provide a full three-dimensional measurement state space.

Nowadays gyroscopes with a high update rate can be built in very small form factors. Nevertheless, the measurements provided by these highly advanced sensors are still prone to drift errors and need correction (achieved by the addition of absolute value sensors).

The main types of gyroscopes (as classified by the underlying technology) are mechanical, optical, and vibrating structure gyroscopes (including MEMS) [216].

Mechanical gyroscopes are the most basic and common type. Figure 3.2 shows a simple gyroscope consisting of a spinning disc constrained to a single axis of rotation mounted within pivoted support frames (called *gimbals*). When the disc is spinning, any rotation applied to the body will rotate the gimbals so that the disc remains stationary. This follows from the principle of conservation of angular momentum and can be used to measure the angular velocity as the body rotates in space. These types of gyroscopes are used in the navigation of large aircraft and missile guidance and control. Since they are typically noisier than other forms of gyroscopes, they are often replaced with more modern forms of gyroscopes.

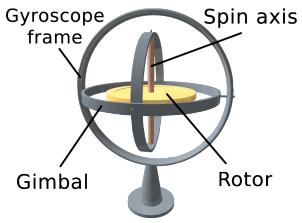


Figure 3.2.: Flywheel gyroscope

Gas-bearing gyroscopes include a rotor suspended by pressurized gas [253]. The presence of gas reduces the friction between moving parts. The gas-bearing gyroscopes are much quieter than other forms of gyroscopes and also have greater accuracy. The Hubble Telescope constructed by NASA is equipped with gas-bearing gyroscopes as the most accurate ones [197]. Even though a new crucial component is introduced in gyroscope of this type, they are still often classified as mechanical.

Optical gyroscopes are based on a different physical law than the previously described types. Optical gyroscopes use two coils of fiber optic cable spun in different orientations. By the Sagnac effect, when the device is tilted, the two beams of light travel different distances, and these distances can easily be measured. Due to the lack of moving parts (and thus, friction), optic gyroscopes are durable and are nowadays successfully used in spacecraft.

Vibrating structure gyroscopes (see Figure 3.3) are another type based on the Coriolis effect. Even when its support rotates, a vibrating body continues vibrating in the same plane: the body exerts a force on the support, and the rotation rate can be measured from the measurement of this force. This type of gyroscopes is generally simpler and cheaper to produce than devices of different types similar in accuracy. MEMS vibrating structure gyroscopes are inexpensive, accurate, and compact, which allows for their wide usage in mobile phones, cameras, smart-watches, and other devices that usually require all of these qualities.

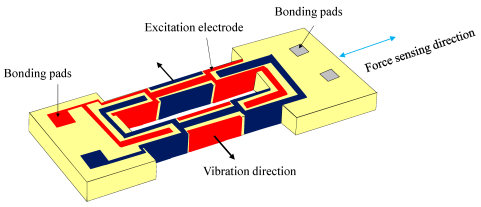


Figure 3.3.: Schematic of vibrating structure gyroscope. Source: [170]

3.1.3. Magnetometer

Much like a compass, a magnetometer is a device used to measure magnetism. Generally, this refers to the magnetic field of the Earth and the magnetometer reports the direction, strength, or relative change of the magnetic field. Due to this, magnetometers constitute a useful tool for determining the heading or orientation of a tracked object.

The magnetometers are usually classified by the nature of the provided measurement [64]:

Vector magnetometers measure the vector of a magnetic field: both magnitude and direction.

Scalar magnetometers (also called total field magnetometers) measure only the magnitude of the vector magnetic field but not the direction.

An alternative classification is suggested in [132]:

Absolute magnetometers use an internal calibration or characteristics of the sensor itself to measure the *absolute* magnitude or vector magnetic field.

Relative magnetometers measure *relative* magnitude or vector magnetic field with respect to a fixed reference. With the use of relative magnetometers variations of the field can be measured.

This concludes the section related to the principal IMU components. The next section is devoted to the review of IMU error types and the suggested techniques that help minimize, neutralize, or avoid them.

3.2. IMU Errors

The goal of using an IMU or a Multi-IMU system is to track an object, i.e., to obtain the current position and/or orientation from the data provided by IMU sensors by integration. The data collected from the sensors can be corrupted by errors and then during the integration, the errors from the measurements are accumulated, causing the so-called *drift error*. To avoid that, errors in the collected data should be minimized or compensated for.

IMU errors can be classified by their nature into deterministic and stochastic. Deterministic errors include misalignment errors and scale factor errors. Stochastic errors are mostly represented by noises. Sensor biases usually consist of multiple components, with some of them being deterministic and some – stochastic. Generally, all of the errors have an influence on the output of a sensor (see Figure 3.4).

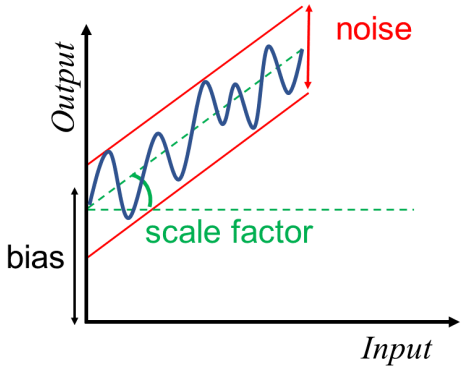


Figure 3.4.: IMU error types. Adapted from [204]

The next subsections offer a review of deterministic and stochastic sensor errors.

3.2.1. Deterministic Errors

As mentioned in the introduction of this section, the deterministic errors of IMUs measurements can be classified into three main categories: biases, scale factor errors, and misalignment errors.

IMU biases are errors present in the measurement regardless of the forces or rates induced on the sensor. It means that when no input is applied to the sensor, measured output presents

a non-zero value, which is called bias. Accelerometer bias is usually measured in *milli-g* or m/s^2; gyroscope bias is measured in deg/s or deg/h.

Biases can be further subclassified into three types: fixed, stability, and instability biases. Fixed biases (or offsets) are entirely deterministic. Stability biases change randomly from run to run of the IMU, and instability biases change as a stochastic process of time.

The *fixed biases*, or *offsets*, in the IMU measurements are the only biases that are entirely deterministic in nature. Fixed biases can create one of the largest errors in an IMU mechanization; the errors increase as a function of time squared for the accelerometer and time cubed for the gyroscope [76].

Another deterministic IMU error that is classified as bias appears when the gyroscope is affected by acceleration. This bias component, called g-dependent bias, is proportional to the magnitude of the relationship between input acceleration and gyroscope output.

Scale factor error is an error that represents how the sensor output reacts to a change in the input. Formally, it is the ratio of a change in the output signal to a change in the input acceleration/angular rate which is to be measured. In Figure 3.4, this error corresponds to the angle of the measured output (in the ideal case, the scale factor is equal to 1, i.e., the measured output equals the sensor input). The magnitude of the scale factor is expressed in parts per million (ppm) or percent.

The scale factor error includes fixed terms, temperature-induced variations, asymmetry, and non-linearity error parts. Asymmetry of the scale factor is measured as the difference between the scale factor on a positive input and a negative input (specified as a fraction of the scale factor measured over the input range). Although it can be seen as a non-linearity error, the IEEE Standard for Inertial Sensor Terminology [125] specifies that this case should be considered separately from other nonlinear components. The major part of scale factor errors is fixed terms and temperature-induced variations, both of which are deterministic errors.

Misalignment or cross-coupling errors arise due to imperfect construction of the IMU, in particular, due to a flawed alignment of the three sensing axes in the accelerometer or gyroscope. As shown in Figure 3.5, if the sense-axis is not aligned with the corresponding body axis, the readings from the sense-axis will be affected by the forces acting on the other two axes, i.e., movement in any of the axes registers in other axes. The unit of misalignment is $milli - radian$.

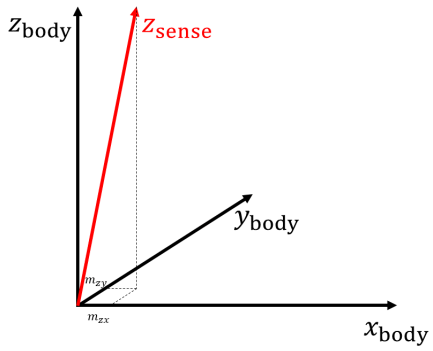


Figure 3.5.: Misalignment error. Adapted from [76]

Deterministic errors can be measured or estimated with specifically designed tests and then minimized or completely eliminated with error compensation algorithms. This procedure is called *calibration*; Chapter 4 is devoted to the review of calibration techniques.

3.2.2. Stochastic Errors

Stochastic errors are sensor errors that do not have a deterministic nature. The major distinguishing feature of stochastic errors is that there may not be any direct relationship between the input and the error. Bias drift, scale factor instability, and random noise are all stochastic MEMS errors.

As discussed in the previous section, some components of bias have stochastic nature. **Bias drift** or bias instability occurs due to a change in bias during the run. In other words, bias instability represents the variations in bias that change with time. *Stability biases* change randomly from run-to-run of the IMU, and *instability biases* change as a random process that is a function of time.

Scale factor instability represents the variations in the scale factor that change with time. Scale factor instability characterization needs a long-term dynamical rate test and the effect of the scale factor instability is not very observable and quite negligible [204].

Random variations in bias and scale factor usually correspond to the low-frequency components of the stochastic errors.

Random noise is another major stochastic error. The noise is an additional signal that interferes with the output signals of the sensor. The noise can come from the considered sensor itself or from other electronic equipment present in the setting. Unit of the random sensor noise density is $deg/h/\sqrt{Hz}$ or $deg/s/\sqrt{Hz}$.

Another stochastic error that is also usually classified as random noise is *quantization noise*. Quantization noise results from the analog-to-digital conversion. This noise is non-linear and signal-dependent.

Sensor noise is responsible for the high-frequency components of stochastic errors.

Stochastic errors, as follows from their name, change with time or from run to run. Therefore, the calibration procedure cannot be used to eliminate them. Stochastic errors have to be modelled as stochastic processes and then eliminated with specifically designed algorithms. This procedure is called *filtering* and is discussed in Chapter 5.

"What I dream of is an art of balance."

— Henri Matisse

4. Calibration

As discussed in Chapter 3, calibration is a technique that helps to detect and neutralize the deterministic part of IMU sensors' errors.

Although in general IMUs can be equipped with different types of sensors, the basic components that provide the measurements are accelerometers and gyroscopes. As explained in Chapter 3, the main errors that are present in these sensors can be classified as bias, scaling factor errors, misalignment, and noise. Following the idea of [12], the sensor measurements can be represented as

$$\begin{bmatrix} u_{A_x} \\ u_{A_y} \\ u_{A_z} \end{bmatrix} = \begin{bmatrix} B_{A_x} \\ B_{A_y} \\ B_{A_z} \end{bmatrix} + \begin{bmatrix} 1 & M_{A_{xy}} & M_{A_{xz}} \\ M_{A_{yx}} & 1 & M_{Ayz} \\ M_{A_{zx}} & M_{A_{zy}} & 1 \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \begin{bmatrix} S_{A_x} & 0 & 0 \\ 0 & S_{A_y} & 0 \\ 0 & 0 & S_{A_z} \end{bmatrix} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} + \begin{bmatrix} \sigma_{A_x} \\ \sigma_{A_y} \\ \sigma_{A_z} \end{bmatrix} \quad (4.1)$$

for an accelerometer. Here vector $A_{x,y,z}$ represents the *actual* acceleration applied to the respectful axes and vector u_A represents the accelerometer output for that input. The deterministic errors are described with a bias vector B_A, a misalignment matrix M_A, and a scaling factor matrix S_A. Vector σ_A refers to the noise errors of the accelerometer.

A similar model for gyroscope is

$$\begin{bmatrix} u_{G_x} \\ u_{G_y} \\ u_{G_z} \end{bmatrix} = \begin{bmatrix} B_{G_x} \\ B_{G_y} \\ B_{G_z} \end{bmatrix} + \begin{bmatrix} 1 & M_{G_{xy}} & M_{G_{xz}} \\ M_{G_{yx}} & 1 & M_{Gyz} \\ M_{G_{zx}} & M_{G_{zy}} & 1 \end{bmatrix} \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} + \begin{bmatrix} S_{G_x} & 0 & 0 \\ 0 & S_{G_y} & 0 \\ 0 & 0 & S_{G_z} \end{bmatrix} \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} + \begin{bmatrix} \sigma_{G_x} \\ \sigma_{G_y} \\ \sigma_{G_z} \end{bmatrix} \quad (4.2)$$

where vector $G_{x,y,z}$ is the angular rate acceleration applied to the respectful axes of the gyroscope, vector u_G is the gyroscope output, vectors B_G and σ_G correspond to the bias and the noise terms of the error, and matrices M_G and S_G are the misalignment and scale factor matrices for the gyroscope.

Sometimes matrices M and S are combined into one matrix T with the diagonal terms representing the scale factor error and the off-diagonal elements representing the misalignment.

Note that these models can in general be used not only for calibration but also for filtering, with respectful errors being split into deterministic and stochastic components. Because the noise errors usually don't have deterministic parts, they are not included in calibration models by most researchers [222]. Then each of the equations of system (4.1-4.2) has 12 parameters that need to be estimated.

Broadly, three groups of methods can be defined: methods that require precise testing equipment, methods that use an auxiliary sensor for estimation, and methods that don't require additional equipment.

4.1. Calibration with Auxiliary Testing Tools

Calibration with auxiliary testing tools is a group of calibration methods that use a leveled turntable (or a similar high-end device) to get precise measurements of the sensors' *input*.

One of the most commonly used basic techniques is the six-position static and rate test [269]. This method uses the assumption that the misalignment errors of the sensors are negligibly small. The idea of that test can be expressed as follows (see Figure 4.1):

1. Place the device on a turn-table in such a way that the sensor's z-axis points directly up.
2. Rotate the device with a constant angular rate ω.
3. Turn the device upside-down, so that the z-axis now points down and the xy plane stays in place.
4. Rotate the device with a constant angular rate ω.
5. Repeat steps 1-4 for axes x and y respectfully.

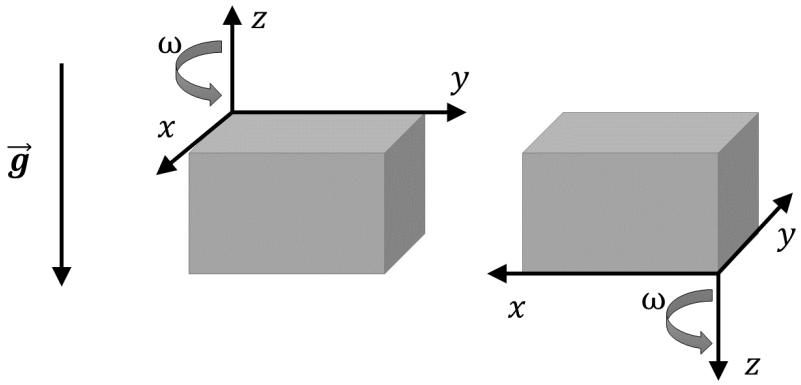


Figure 4.1.: IMU placement for z-axis in the six-position test

The six-position method usually uses a simplified sensor error model that is described by

$$\begin{aligned} u_k^{up} &= B_k + (I + S_k) \cdot G \\ u_k^{down} &= B_k - (I + S_k) \cdot G, \quad k \in \{x, y, z\}, \end{aligned} \tag{4.3}$$

where k is the axis index (x, y, or z), u^{up} and u^{down} are the respective sensor measurements when the axis is pointed up or down, B is the component of the bias, I is an identity matrix, S is the scale factor error, and G is the gravitational constant (for accelerometers) or the magnitude of the earth rotational rate (for gyroscopes) at the local latitude. This method does not allow estimation of the misalignment error; moreover, it is derived under the assumption

of this error being negligible. If in reality the sensor's axes are noticeably misaligned, the method will not give a reasonable calibration result.

An improved six-position test [202, 3, 112] takes misalignment errors into consideration. The method starts with the general model (4.1-4.2) for the error sensors (with matrix T representing both misalignment errors and scaling factors). Then, a least-square method is used to find estimates for the elements of misalignment and scaling matrix:

$$\widetilde{T} = UA^T(AA^T)^{-1}, \tag{4.4}$$

where matrix $\widetilde{T} = [T|B]$ is a concatenation of matrix T and vector B, matrix U is a concatenation of sensor outputs for the 6 test positions [263], namely

$$U = \begin{bmatrix} u_{+x} & u_{-x} & u_x & u_x & u_x & u_x \\ u_y & u_u & u_{+y} & u_{-y} & u_y & u_y \\ u_z & u_z & u_z & u_z & u_{+z} & u_{-z} \end{bmatrix}, \tag{4.5}$$

and matrix A is the matrix of sensor inputs

$$A = \begin{bmatrix} a & -a & 0 & 0 & 0 & 0 \\ 0 & 0 & a & -a & 0 & 0 \\ 0 & 0 & 0 & 0 & a & -a \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.6}$$

with $a = g$ (gravity) for accelerometer and $a = \omega$ (input angular rate) for gyroscope.

An alternative approach of estimating the model parameters was suggested in [58] with a weighted least-square method:

$$\widetilde{T} = (UPA^T)(APA^T)^{-1}, \tag{4.7}$$

where P is a 6×6 weight matrix based on sample variance of the accelerometer errors.

This model (with any of the suggested least-square estimators) allows finding reasonably good estimating values for the errors of both accelerometer and gyroscope. However, it relies heavily on the accuracy of the turntable to provide the input data for the gyroscope calibration.

A similar approach proposed in [249] and then improved in [263] suggests a larger number of testing positions for sensors' error estimation. Again, the technique relies heavily on the use of a high-precision turntable for gyroscope data acquisition. The study considers a MEMS IMU as an aiding tool for GPS to ensure the quality of navigation in the locations without a clear line of sight to the satellites. A multi-position calibration method is proposed that allows to compensate for deterministic sensor errors.

The general model that provides the base for the proposed method was proposed in [249]. Starting with this model, authors consistently complexify the approach by including non-orthogonality of the sensor axes (to treat misalignment errors), biases, and scale factors. The least-squared-error approach is then applied to the linearized model, and the following analysis is performed by collecting sensor data from 26 different attitudes of the IMU.

Verification was performed for the calibrated systems by comparing the position drifts in 3 field tests datasets. The test vehicle setup is presented in Figure 4.2.

Figure 4.2.: Test vehicle setup. Source: [263]

During the calibration stage of the experiment, the original idea of using the earth rotation rate as the reference for the gyroscope calibration model was discarded. To provide a strong reference rotation rate for the high noise sensors, a turn-table was used. The experiment was then successfully concluded. The results of the verification are presented in Figure 4.3.

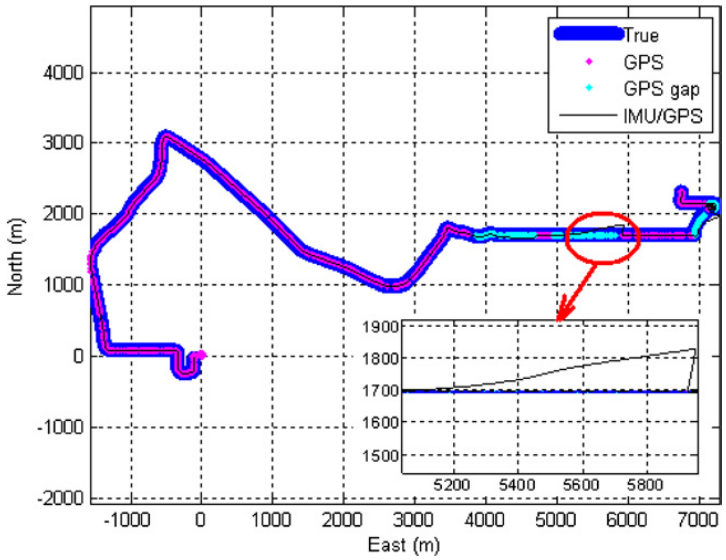


Figure 4.3.: Trajectory for run 3 along with simulated GPS outages. Source: [263]

The proposed calibration method showed good results for the suggested use of IMU calibration for aided GPS navigation (see [263] for detailed results). The use of the multiposition calibration technique allows improving the quality of the traditional six-position test. However, this method cannot be applied to the systems where the accelerometer and gyroscope are significantly misaligned. This method uses a simplification, in the sense that it calibrates both sensors to the same artificial reference frame. Then, if the actual misalignment between the triads cannot be neglected, the method presents more errors than it can eliminate.

In [251] a research group from Signal Processing Lab (Stockholm, Sweden) presented a

calibration method for a MEMS inertial measurement unit that was based on a simple rotating table for the gyroscope and no mechanical platform for the accelerometer. The authors present an in-house constructed low-cost inertial measurement unit (see Figure 4.4) and then provide a sensor model description that is later tested for error elimination.

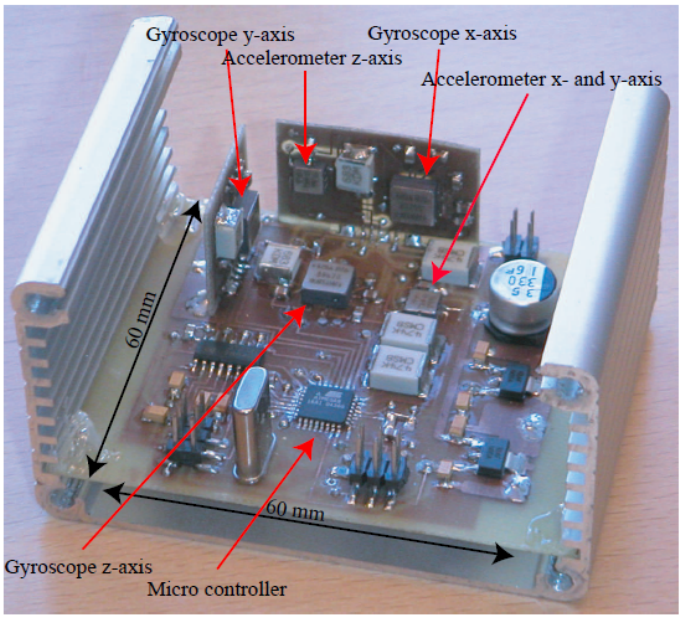


Figure 4.4.: The in-house IMU with 3 accelerometers and 3 gyroscopes. Source: [251]

The sensor model used in the paper includes misalignments, scale factors, biases, and measurement noise. Both accelerometer and gyroscope are suggested to be calibrated with this model. A method for estimating the parameters of this model is also proposed by the authors: The estimation is performed by minimizing the squared error between the squared magnitude of the input and the square magnitude of the compensated IMU output.

This approach, however, cannot provide a proper gyroscope calibration, because some of the parameters of the gyroscope model are not observable. For that reason, the authors use a mechanical platform for the IMU rotations. With the assistance of this high-end device, the model is verified and the presented results show a good correspondence with the theoretical estimation.

An alternative approach is presented in [26]. This paper describes a low-cost inertial reference unit (IRU) developed for spin-stabilized sounding rockets. The application field of the device demands high-precision calibration of sensors.

The constructed IRU (see Figure 4.5) includes 3 MEMS gyroscopes: two for angular rate in yaw and pitch and one for roll rate. The suggested use of a one-axis rate table potentially presents a new source of sensor errors: the IRU has to be remounted on the table after each calibration run and new misalignment angles can be introduced. To avoid that unnecessary effect, a special L-shaped mounting fixture was developed by the author to fix the nominal

input axis parallel to the rate table rotation axis.

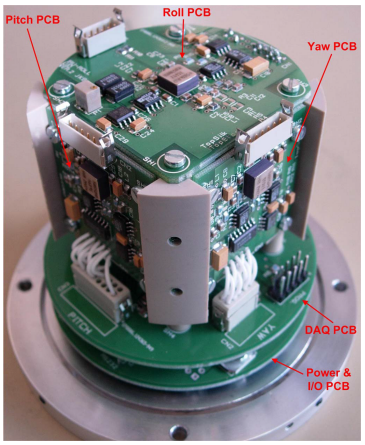


Figure 4.5.: Inertial Reference Unit. Source: [26]

To allow the use of a one-axis rate table for full calibration of a three-axis device, a sequential procedure was developed by the author. A ramp profile is first applied to the first axis, then the IRU is remounted, and the same profile is applied to the second axis and similarly to the third one. The data from the sensors is then processed with a Kalman filter and the gyroscopes' calibration parameters are obtained. Table 4.1 presents the *simulation* results from the IRU calibration in comparison with the true values. Here δ_{ij} is the misalignment angle between axes i and j, λ_i is the gyroscope scale factor error in axis i, and β_i is the gyroscope bias in axis i. With the simulation results, the software part of the method is verified.

Parameter	**True value**	**Estimated value**
δ_{xy} (deg)	1.500	1.503
δ_{xz} (deg)	0.700	0.704
δ_{yx} (deg)	-1.300	-1.300
δ_{yz} (deg)	0.800	0.799
δ_{zx} (deg)	1.300	1.300
δ_{zy} (deg)	-1.100	-1.098
λ_x	$1.920 \cdot 10^{-2}$	$1.919 \cdot 10^{-2}$
λ_y	$2.094 \cdot 10^{-2}$	$2.093 \cdot 10^{-2}$
λ_z	$1.571 \cdot 10^{-2}$	$1.573 \cdot 10^{-2}$
β_x (deg/sec)	2.000	1.994
β_y (deg/sec)	1.300	1.301
β_z (deg/sec)	1.000	1.0

Table 4.1.: Simulation results from IRU calibration. Source: [26]

The paper also presents results for a comparison of calibrated and uncalibrated systems in an experiment (see Figures 4.6 and 4.7). The results show a clear error reduction (the absolute

rate error of the calibrated IRU is reduced by more than a factor of 10), which means that the proposed calibration technique can be successfully used in practice.

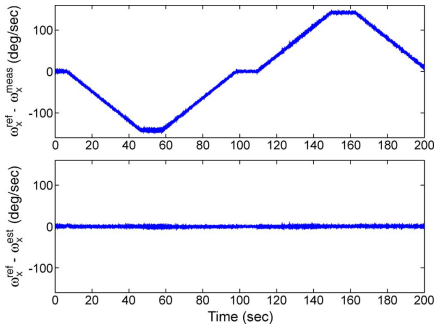


Figure 4.6.: Roll errors for the (top) uncalibrated and (bottom) calibrated IRUs. Source: [26]

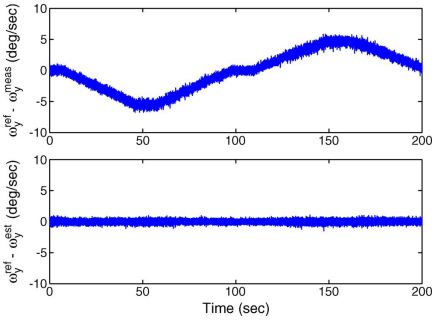


Figure 4.7.: Pitch errors for the (top) uncalibrated and (bottom) calibrated IRUs. Source: [26]

Paper [159] by a group of scientists from South Korea also proposes an approach for error parameter estimation for MEMS that only uses a single-axis rate table. The IMU that is presented in the paper (see Figure 4.8) consists of 8 single-axis sensors, namely 4 accelerometers and 4 gyroscopes. The main difference of this approach is the numerical method: the authors propose the use of the Fourier transform method.

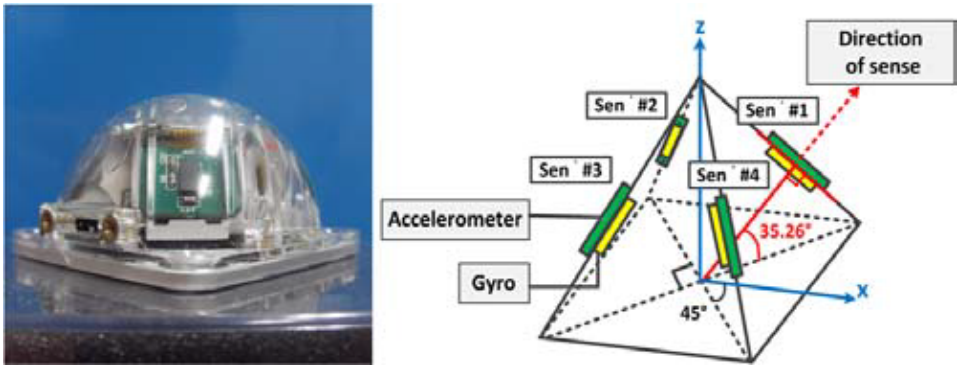


Figure 4.8.: Inertial Measurement Unit. Source: [159]

The authors compare a static calibration procedure (the data is collected from sensors when IMU is static) with a dynamic procedure with attitude change mounts (the rate table rotates at a constant angular rate). The acquired data are then processed with a sensor parameter estimation algorithm. Authors compare the recursive least squares method (RLS), which is commonly used for parameter estimation problems, with a Fourier transform (FT) method. The root mean square error (RMSE) of estimated values (in reference to the true values) computed with RLS and FT for static and dynamic tests are provided in the paper (see Table 4.2).

Setup	Sensor / Method	Accelerometer [g]			Angular velocity [°/s]		
		X	Y	Z	X	Y	Z
Static	RLS	0.0099	0.0128	0.0474	0.0122	0.0179	0.0372
	FT	0.0099	0.0128	0.0475	0.0116	0.0171	0.0371
Dynamic	RLS	0.0098	0.0109	0.0476	0.0116	0.0091	0.0162
	FT	0.0088	0.0117	0.0432	0.0112	0.0084	0.0163

Table 4.2.: RMSE of calibrated IMU parameters. Source: [159]

Judging by the presented data, both RLS and FT methods provide a similarly good quality of parameter estimation. The dynamic calibration procedure proposed in the paper, in principle, gives better quality of estimation than the classical static procedure. Since the dynamic test is usually performed with a multi-axis rate table, the dynamic calibration procedure for a single-axis table with a corresponding model description presents a basis for a lower-cost calibration system that can perform high-precision calibration.

Alternatively to turntables other auxiliary testing tools have been studied by the researchers. In [49] a calibrating technique based on pendulum motion is presented (see Figure 4.9).

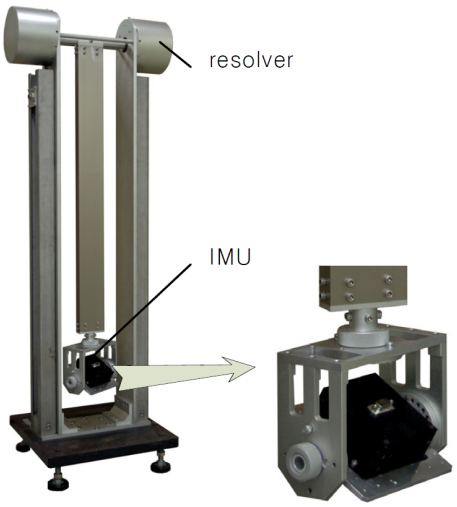


Figure 4.9.: Pendulum calibration system. Source: [49]

In this study, angle sensors are employed for measuring the movement of the pendulum with the attached IMU. The nonlinear equation for pendulum motion is derived in the paper and the calibration parameters are then obtained with the use of a neural network.

In [234] a robotic arm was used for calibration. The robot arm consists of six joints, and the IMU is attached to the last of them. The sensor data are collected from three of the joints, performing movements in roll, pitch, and yaw. A specific optimization function was derived accordingly based on a standard least mean square procedure.

The techniques presented in this subsection rely on specific high-end testing tools that are

generally employed to collect the precise data for gyroscope input.

4.2. Calibration with Auxiliary Sensors

Alternatively, some researchers suggest methods that use auxiliary sensors to collect the necessary data or improve the general calibration performance.

In [20], a study of the misalignment angle of an IMU is presented for a case when the IMU is mounted on a vehicle. The research is only focused on misalignment error; the authors suggest using preexisting techniques like Kalman filter for the other types of sensor errors. The main idea of the suggested calibration technique is to use the data obtained from a GPS device of the vehicle as a reference. The results of the calibration provided in the paper show a reduction in acceleration error and an improvement of the position estimates.

Paper [290] considers a Multi-IMU system, where all of the components are calibrated simultaneously. A scheme of the setting is provided in Figure 4.10. Information obtained from neighboring IMUs is used for calibration.

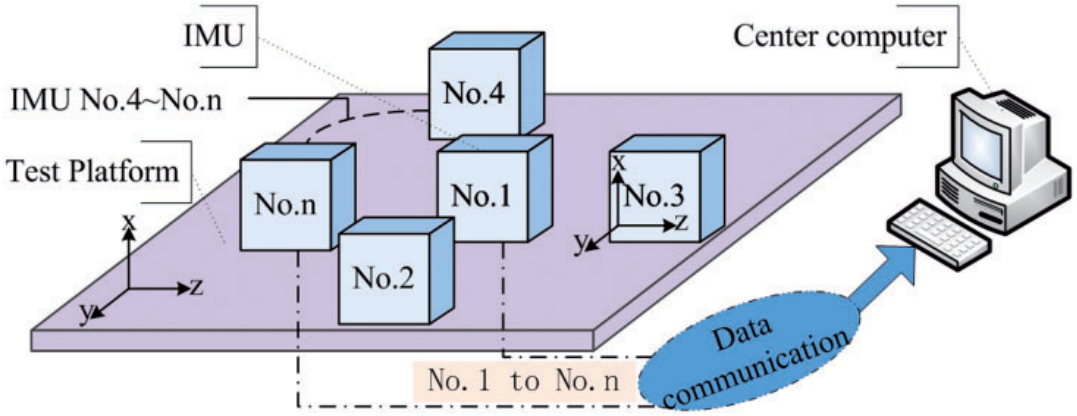


Figure 4.10.: Test platform with installed IMUs. Source: [290]

The presented approach is based on a dual-loop Kalman filter (see Figure 4.11) that consists of an inner filter and an outer filter. First, the state vector (i.e., the vector of sensor errors) is roughly estimated in the inner filter for each IMU separately. Then, the outer filter fuses the obtained data to improve the estimates.

The paper provides a detailed description of the calibration procedure and an extensive study of simulated and practical experiments. Although in the paper a turntable is used to ensure precision in turning, the presented results show the effectiveness of the method and its potential use for the simultaneous calibration of several IMUs.

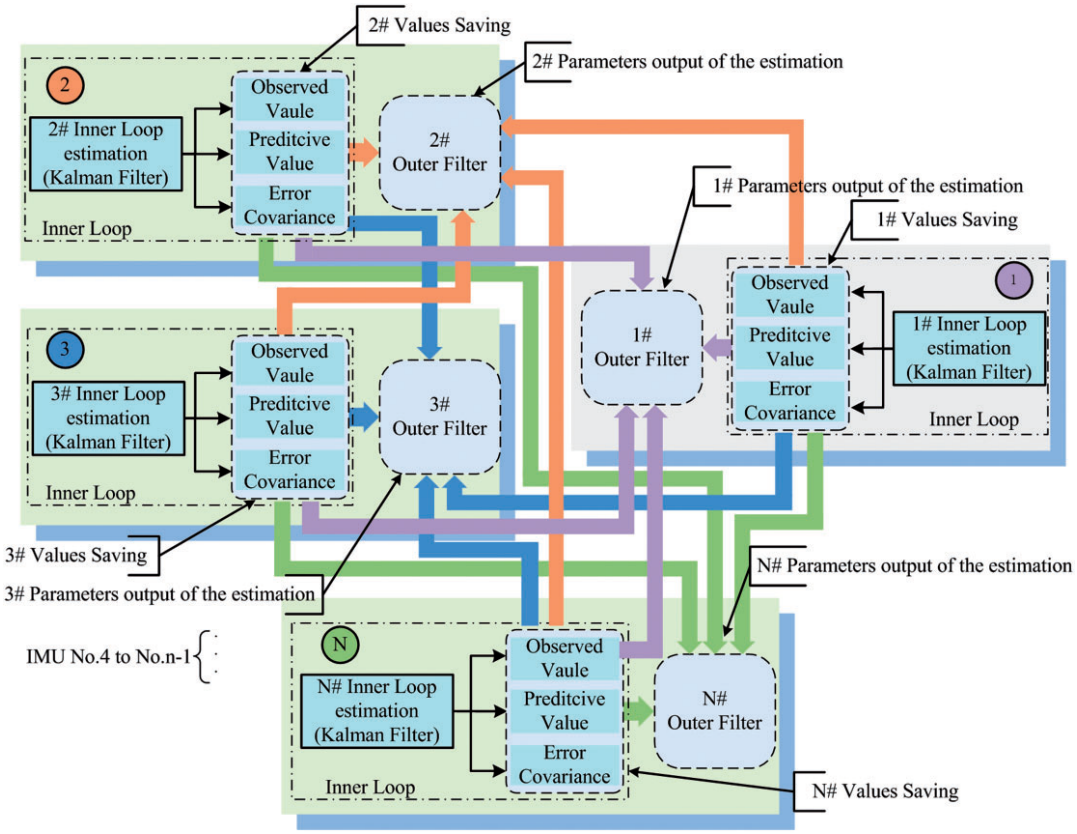


Figure 4.11.: Proposed dual-loop filter data flow. Source: [290]

Another group of methods uses cameras for IMU calibration. Paper [277] describes a method that allows estimating calibration parameters of an inertial sensor based on a "shape-from-motion" approach. This approach, in turn, is based on the factorization of raw sensor output data and produces the matrices that describe the applied load and the calibration matrix. The authors provide a comparison with the conventional least-squares method and the results show a similar accuracy level. Unfortunately, this approach doesn't estimate all of the sensor errors, and a separate investigation of bias errors is needed.

This shortcoming was eliminated in [124] when the factorization approach was extended to redundant (non-triad) configurations, which allowed the authors to also estimate the bias of the sensors in the same run.

A vision-based 3D optical tracking system that gets a true measurement estimate from an optical sensor was proposed for aided IMU calibration in [61]. The presented approach allows to estimate bias, scale factor, and misalignment of the IMU accelerometer with a standard least-square technique; gyroscope calibration, however, is not considered.

Some authors also present approaches for the simultaneous calibration of an IMU and a camera.

In [8] a joint IMU-camera calibration was attempted with the sensors attached to a pendulum (see Figures 4.12 and 4.13). The paper estimates the relative orientation between the optical and inertial sensors taking into consideration the scaling factor and misalignment of IMU.

Figure 4.12.: Setup with the pendulum. Source: [8]

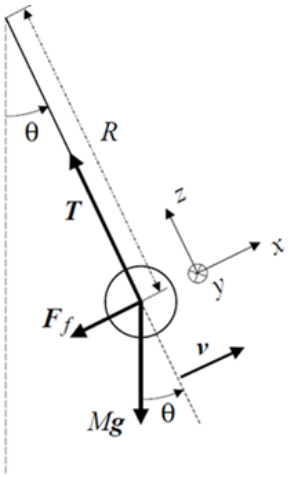


Figure 4.13.: Forces acting on a moving pendulum. Source: [8]

Research [174] continues on the same topic proposing a two-step approach: first, the relative *orientation* of the IMU and the camera is determined, and then a turntable is used to determine their relative *position*. The second part of the approach, however, is a very difficultly performed operation because the IMU has to be set into the exact center of the turntable (to get the lever arm vector).

Research [295] presents a calibration procedure for an IMU and a monocular camera that only requires a visual pattern (see Figure 4.14) as a calibration tool. The approach is based on recursive Unscented Kalman Filter; bias, scale factor, and misalignment IMU errors are estimated together with the IMU-camera coordinate transformation parameters, i.e. the registration of the IMU and the camera.

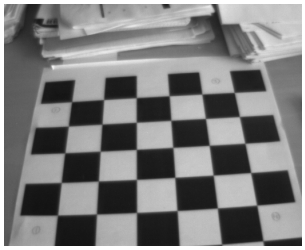


Figure 4.14.: Visual pattern used for joint Camera-IMU calibration. Source: [295]

4.3. Calibration without Tools

As shown in the previous sections, IMU sensors calibration can be successfully performed with either auxiliary testing tools, like rate tables, or augmenting tracking devices, like GPS or cameras. Note, however, that the standard 6-position static test [269] or its improved versions [202, 3, 112] allow unaided accelerometer calibration: all the deterministic accelerometer errors can be estimated using the gravity vector as a reliable known input. So, this section presents a review of calibration techniques that don't require any additional tools, the focus being on unaided gyroscope calibration.

Paper [78] presents a calibration technique for a three-axis accelerometer and a three-axis gyroscope without any auxiliary tools. The setup (see Figure 4.15) is a strap-down platform that consists of an accelerometer and a gyroscope enclosed in a case (with a specified reference system). The case is placed on a base furnished with a reference block. This setup provides a fixed rotation axis for gyroscope calibration (as the normal direction for the support base).

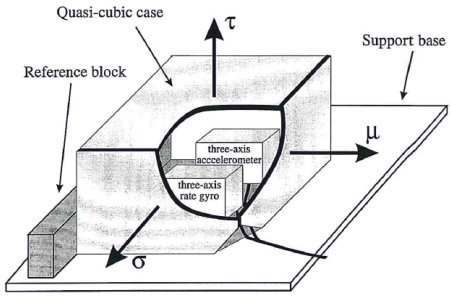


Figure 4.15.: Experimental setup. Source: [78]

The proposed method allows estimating the biases and scale factors of both sensors but not the misalignments. Nevertheless, the results obtained for the estimated parameters prove to be accurate and the procedure itself is simple to perform.

An improved version of the method was proposed in [137]: misalignment error, as well as bias and scale factor, are estimated. The IMU considered in the paper includes an accelerometer, a gyroscope, and a magnetometer. The deterministic errors of the latter are also taken into consideration in the research. The observations for the gyroscope error model parameters are collected (following and improving the procedure suggested in [78]) by the full rotation of the IMU base about its normal axis. The authors propose an iterative scheme for error model parameters estimation and the results are compared with the experimental data obtained with an optical kinematic measurement system (see Figure 4.16).

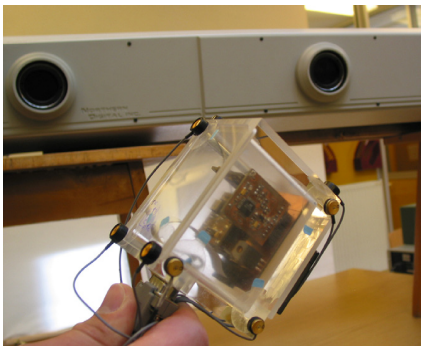


Figure 4.16.: Measurement setup. Source: [137]

The results presented in the paper allow concluding that the schemes provide a good estimation of deterministic errors of IMU sensors; however, the computational scheme requires massive calculations due to its iterative nature. It should also be noted that both accelerometer and magnetometer data were used to enhance and assist gyroscope calibration.

Another tool-free technique for gyroscope calibration that relies on the data from the accelerometer and magnetometer is presented in [48]. The sensor suite of the IMU considered in the work is presented in Figure 4.17.

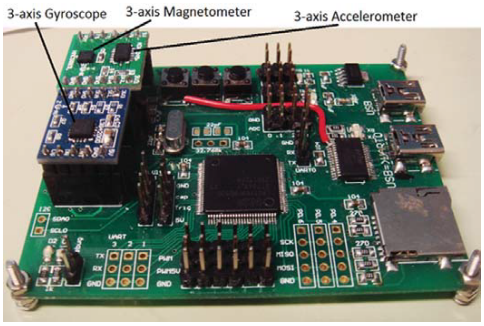


Figure 4.17.: Sensor suite of the IMU composed of a gyroscope, a magnetometer, and an accelerometer. Source: [48]

The authors suggest a set of static and quasi-static tests for calibration and a quasi-static detector technique for the static state condition. This technique allows estimating the gyroscope error model parameters by only performing several rotations by hand. The suggested tests are presented in Figure 4.18. The top row shows the suggested static placements for accelerometer calibrations: six faces placed horizontally (top left) and twelve edges placed at 45° (top right). The bottom row shows the suggested rotational motions for gyroscope calibration: rotations about the three major axes (bottom left) and rotations about cross-axes in the xy plane (bottom right).

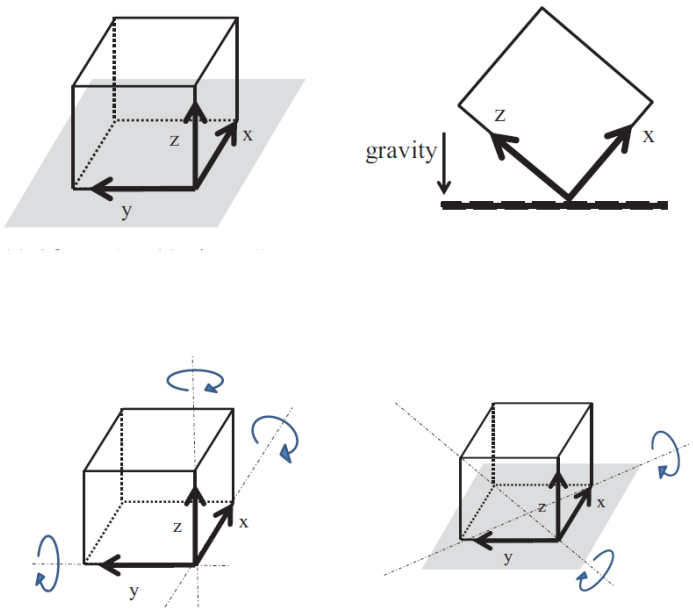


Figure 4.18.: Suggested static placements for accelerometer calibration (top row) and rotational motions for gyroscope calibration (bottom row). Source: [48]

The scheme shows promising results for error estimations; however, the underlying error model doesn't include inter-sensor misalignment. Moreover, the authors conclude that the results may depend on the variations in the magnetic field since the calibration procedure relies on magnetometer data.

A somewhat similar approach is described in [265]. The authors also calibrate the gyroscope by comparing the gravity vector measured by the calibrated accelerometer with the gravity computed from the gyroscope data. As opposed to the previous approach, however, the authors don't rely on the magnetometer data for calibration, which allows for the use of their method with IMUs that are not equipped with a sensor of this type.

The protocol for the calibration procedure proposed in the paper is presented in Figure 4.19.

The authors also focus on an important practical issue, namely, on the automation of the calibration procedure. The proposed approach relies on the fact that after each rotation comes a static period. Because the proposed approach doesn't include any additional equipment

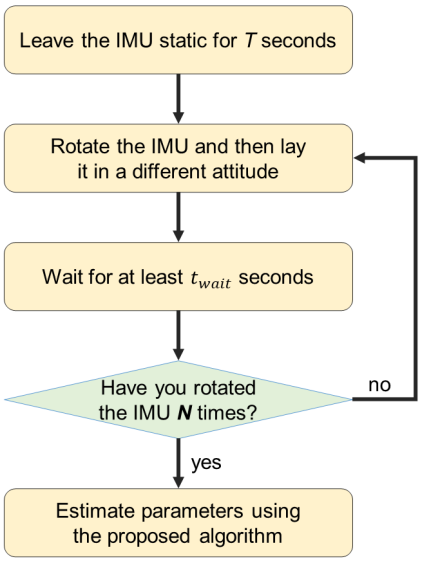


Figure 4.19.: Diagram of the calibration protocol. Adapted from [265]

that could record the performed operations, the automation is achieved through the use of a static detector, that selects the static intervals from the obtained data. An example of the application of the detector is presented in Figure 4.20.

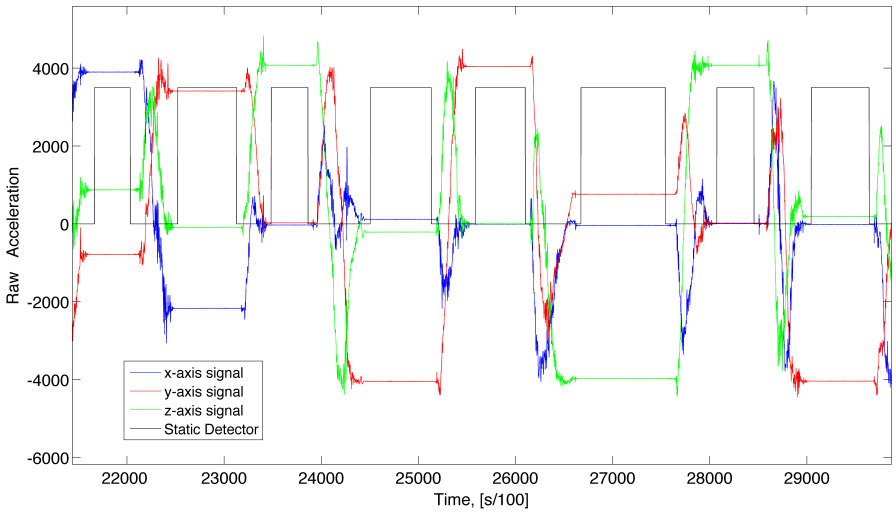


Figure 4.20.: Application of static detector to accelerometer data. High level of detector curve represents static intervals. Source: [265]

The estimation of calibration parameters is achieved through the application of the Levenberg-Marquardt algorithm to two quadratic objective functions. For the accelerometer,

the objective is defined as

$$L(\theta^{acc}) = \sum_{k=1}^{M} \left(||g||^2 - ||h(a_k^s,\ \theta^{acc})||^2 \right)^2, \tag{4.8}$$

where vector θ^{acc} contains the calibration parameters of the accelerometer, g is the local gravity vector, the magnitude of which can be recovered from table data, and a_k^S is the vector of accelerometer measurements. Constant M is the number of static intervals that are included in the calibration procedure. Function $h(a^s, \theta^{acc})$ represents the error model for accelerometer data, combining the readings from the sensor with the error parameters.

The objective function for gyroscope calibration is

$$L(\theta^{gyro}) = \sum_{k=2}^{M} ||u_{a,k} - u_{g,k}||^2, \tag{4.9}$$

where θ^{gyro} is the vector of gyroscope calibration parameters, vector $u_{a,k}$ is the gravity versor obtained from the accelerometer and averaged through the entire static window, and $u_{g,k}$ is the gravity versor computed from the gyroscope data as

$$u_{g,k} = \Psi \left(\{\omega_i^S\}_{i=1}^n, u_{a,k-1} \right) \tag{4.10}$$

through an integration algorithm Ψ, that takes as input the gravity versor from the accelerometer $u_{a,k-1}$ and a sequence of gyroscope readings ω_i^S.

It should be noted that the proposed IMU calibration method requires a good calibration of the accelerometer because the gyroscope calibration algorithm is sensitive to the errors in accelerometer data. For the same reason, an accurate integration method should be used, otherwise, the integration error will seriously affect the calibration quality. Although according to the paper, the normalized 4-order Runge-Kutta method provides a reasonable level of accuracy, a predictor-corrector 4-5-order Runge-Kutta or 4-order Crouch-Grossman methods can be expected to be more accurate in these types of problems [10].

An alternative approach was proposed in [89] and later improved in [90]. In this research, an approach based on Artificial Fish Swarm Algorithm (AFSA) is presented that allows estimating accelerometer and gyroscope error model parameters. The AFSA method is a global optimization method based on the application of behaviorism in artificial intelligence [168]. The flow-chart of optimal modification of AFSA (OAFSA) from [89] is presented in Figure 4.21. The main difference from the standard AFSA is an improved balance between exploration and exploitation in the method.

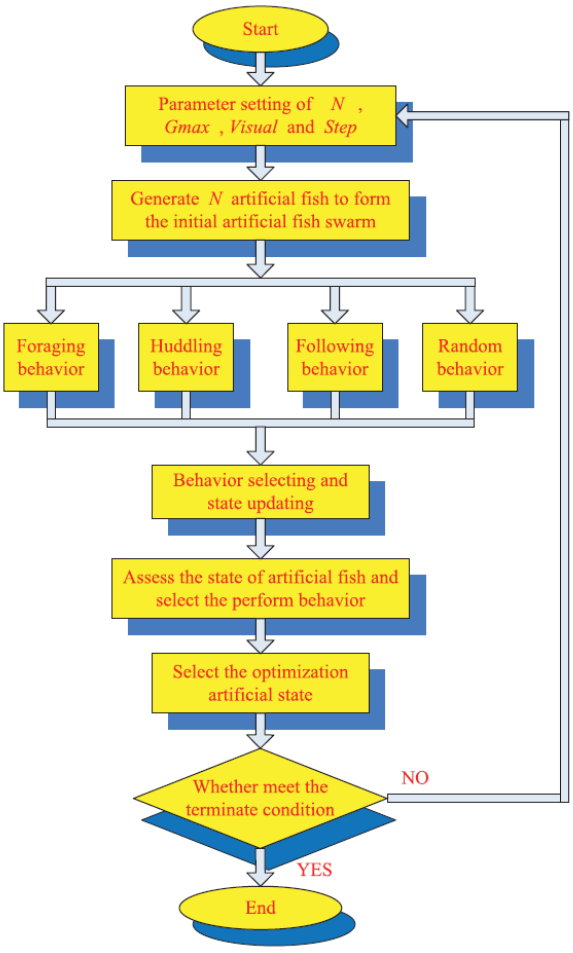


Figure 4.21.: The flow-chart of OAFSA. Source: [89]

By adding specific parameters and eliminating others, another modification of AFSA called novel AFSA (NAFSA) can be obtained. The main feature of this modification is the reduced structural and computational complexity. NAFSA is used in [90] to estimate both accelerometer and gyroscope deterministic errors. The results are compared with the ones obtained through a conventional method (static and dynamic placement test with a rate table) through a series of experiments. Trajectory comparison between the reference solution, a solution obtained after a conventional calibration, and two solutions with AFSA modifications are compared in Figure 4.22.

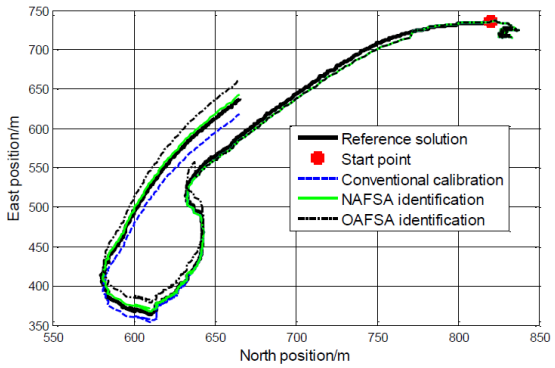


Figure 4.22.: Trajectory comparison. Source: [90]

The presented results suggest that the NAFSA-based approach outperforms other presented calibration methods in terms of accuracy; moreover, it has low complexity and doesn't require any additional tools.

Another set of tool-free approaches is based on Kalman filters. This technique was first suggested for calibration in the 1970s in [115] and [147]. Models presented in these works combine calibration and fusion in iterative filtering. Both of the models require the use of additional high-end equipment, namely, a centrifuge in the former work and a rate table in the latter. Nevertheless, these works provided a reliable basis for further development of the methodology.

In [169] no additional equipment was used to perform Kalman filter-based calibration. The approach was tested in two IMUs (see Figure 4.23): a high-end one and a low-end one.

Figure 4.23.: Tested IMUs. Source: [169]

The high-end IMU was first calibrated with a standard six-position method and then, additionally, with a proposed technique, so that any error value estimated by the technique can be treated as a technique error. Furthermore, artificial errors were added to the output of this high-end device to simulate a low-end sensor. The technique was then tested on the anti-enhanced IMU. The high-end device was also used as an oracle for experiments with the low-end IMU.

The approach itself only requires static tests or hand motion-driven ones. The error estimation results obtained for all the experiments are in good correspondence with the true values. However, this approach only considers sensor biases and scale factor errors, and not misalignment or cross-sensor calibration.

A further improvement of the Kalman filter-based calibration approach was presented in [60]. The authors propose a method that allows estimating error parameters of accelerometers and gyroscopes without additional equipment. The error models for both sensors include biases, scale factor errors, and misalignments.

The model of gyroscope error estimation is based on the comparison between the acceleration from the gravity measured by the calibrated accelerometer and the gravity computed from gyroscope outputs. This model is then embedded into the Kalman filter and the parameters are estimated via a calibration procedure. For the accelerometer, the authors propose a set of static position tests and for the gyroscope, a conventional rotation scheme is presented which doesn't rely heavily on the precision of rotation.

The results of the calibration were compared with the results obtained from a standard six-position test, which was performed with the use of a rate table. The data from the IMU calibrated with two different methods were then used for pose estimation and the position drifts are presented in Figure 4.24 along with the drift induced by uncalibrated sensors.

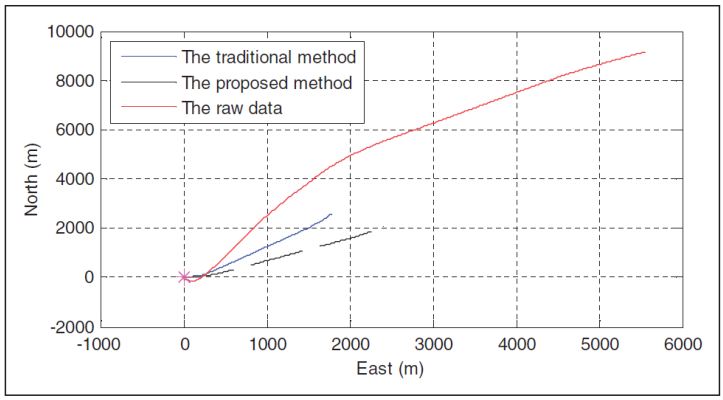


Figure 4.24.: Position drifts in a performed experiment. Source: [60]

The comparison shows that the proposed method allows estimation of error parameters for both accelerometer and gyroscope at least at the same general level of accuracy as the classic approach.

4.4. Calibration of Multi-IMU

This section presents a practical use of calibration. Practical results concerning a comparison between a system with and without calibration are presented, analyzed, and discussed.

The problem presented in this section comes from practice. The market of IMUs nowadays provides devices at a very wide range of prices. Obviously, the devices of different costs also differ in performance: usually, high-cost IMUs give better results than low-cost ones. Nevertheless, high-cost IMUs also produce sensor errors and mitigate the accuracy of the tracking system. Thus the problem investigated in this section can be formulated as follows: can a system of 3 low-cost IMUs reach the same performance in terms of accuracy as one high-cost IMU.

In the experiment, published by the author et. al in [131, 130], two different fused systems of 3 independently calibrated low-cost IMUs on a planar grid and on a non-planar grid are presented, and their performance is compared to that of a single high-cost IMU. For the comparison, several 1 DoF measurement setups for mechanical rotation and translation movements are provided.

The main focus of this experiment was on the calibration of the IMUs. One of the objectives was to create a low-cost tracker that can provide accurate measurements. For that reason, a method of non-aided calibration described in [265] was chosen. As reviewed earlier, the method allows estimating most of the calibration parameters of both accelerometer and gyroscope without additional tools or trackers.

In practice, however, an observability problem arose during the application of the method. The accelerometer calibration, performed according to the method via a series of static tests,

was successful. The gyroscope calibration, however, suffered immensely. The authors of [265] propose to excite the sensor by hand movements. But the input obtained by gyroscope when moved by hand could not be measured accurately and also could not provide high angular velocity, which drastically decreased the calibration quality. To improve the quality, a controlled environment was created with the use of a tripod, similar to the one presented in Figure 4.25.

Figure 4.25.: Smartphone tripod. Source: [232]

With the use of the tripod, rotation of over 90° could be achieved within a reasonable time interval of ~ 5 seconds, thus giving the gyroscope an input of $\sim 18°/sec$. The calibration enhanced with this additional low-cost physical device showed better results than the original totally non-aided one.

Table 4.3 summarizes the main results of the research described in this section. This table was published in [131]. The results, presented in this table, confirm that by combining three low-cost IMUs into a specifically designed system one can reduce the error beyond the performance level of a high-cost IMU.

		Rotation, °/s		Slider, m/s^2	
single	raw	7.9871		0.6978	
	calibrated	7.8708		0.4951	
		plate	*cube*	*plate*	*cube*
fused	raw	4.6392	8.6335	0.7962	0.4603
	calibrated	4.2951	6.5493	0.5901	0.4559
	high-cost	6.7723		0.6177	

Table 4.3.: RMSE values for all configurations

The research that was presented in this section shows that a specifically designed fusion pipeline can provide a chance of improving existing setups by adding new IMUs instead of replacing the installed sensors with higher-cost devices. For a particular set of 3 identical low-cost IMUs, the performance equaled or exceeded that of a single high-cost IMU.

"Even the word 'silence' makes noise."

— Georges Bataille

5. Filtering

As discussed in Chapter 3, IMU sensor errors consist of deterministic and stochastic components. To eliminate the latter, a procedure called *filtering* is applied. This chapter is devoted to a discussion on various filtering techniques. Filtering is a procedure that extracts information about a particular quantity of interest at a fixed time-frame by using the data observed up to this time frame [47].

There exist various filtering techniques. The most commonly applied techniques in IMU sensor data filtering belong to the classes of frequency-based filters and Bayesian filters. The filters belonging to the mentioned classes are reviewed respectively in Sections 5.2 and 5.3. Section 5.1, preceding the two filtering sections, is focused on Exploratory Data Analysis, which provides insight into the data series and helps select correct filtering techniques and successfully implement them.

It should also be noted that the parameters of the models that form the base of Bayesian filters are usually estimated conjointly, sometimes bringing together data from multiple sensors. That means that while the errors are getting estimated, data fusion can also be performed simultaneously without additional effort. For this reason a lot of *fusion* techniques (see Chapter 8) are based on (modified) Bayesian filters.

5.1. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to data analysis that uses both graphical and quantitative methods to analyze the data, that is to get insight into the data set, uncover the underlying structure, detect anomalies, construct and test assumptions, develop appropriate models and their parameter settings [79].

As this chapter deals with noise elimination, the main focus is on stochastic processes analysis. The model of a stochastic process is built through three steps:

1. Model selection:

   An appropriate model needs to be selected. The methods that allow for choosing a model are described in Subsection 5.1.1, and the models are reviewed in Subsection 5.1.2.

2. Model fitting:

   Once a model is chosen, the parameters of the model need to be estimated from the particular observed data. Different techniques can be used at this step, such as

maximum likelihood (Subsection 5.1.3.1), least squares (Subsection 5.1.3.2), or explicitly Bayesian filtering (Section 5.3). In general, there exist infinite possibilities for model fitting depending on the choices of the optimized objective. Nevertheless, these and their variations are most commonly used in practice because the estimators they provide have a set of proved desirable qualities.

3. Model validation:

   When the parameters are found for the chosen model, validation is performed to assess if the model can provide reliable and plausible results for analysis purposes. Generally, a residual analysis is used for model validation, but other techniques, e.g., lack-of-fit-test can also be used at this stage [105]. Some advanced filters include this stage into their body and perform filtering recursively to reestimate the state vector in such a way that the designed tests are passed, see for example Adaptive Fading Unscented Kalman Filter in Subsection 5.3.1.4.

5.1.1. Time Series Analysis

The tools developed for time series analysis allow to estimate and classify stochastic components of IMU errors. The approaches that are of main interest in this perspective are the ones that allow understanding or estimating the model of the process from the measured data. Four main approaches are presented in this subsection: lag plots, autocorrelation, power spectral density, and Allan variance.

5.1.1.1. Lag plots

A lag is defined as a fixed time displacement. In principle, lag plots can be generated for any lag, but in practice, most of them are plotted for lag$=1$. A 1-lag plot is a scatter plot where all of the measured data on time-step k are presented in the Y-axis, and all of the data on step $k-1$ are in the X-axis. Lag plots allow checking whether a data set (or a time series) has a random nature at all and whether the series exhibits an autocorrelation (for a formal definition of autocorrelation see Section 5.1.1.2). They also clearly show outliers (if present) and in some cases can provide a basis for suggesting a stochastic model for the process. To clarify the usage of lag plots, three examples are presented.

Figure 5.1 shows a lag plot of 200 normally distributed random numbers ($z_k \sim N(0,1)$). The following conclusions can be made from this plot:

1. The data contain no outliers.

   Although the plot shows several points that are located further away from the center than the others, the relative distance is too small to deem them as outliers.

2. The data are random and have no autocorrelation.

   The randomness of the data can be stated from the absence of any structure on the plot. The current value z_k cannot be predicted from the previous value z_{k-1}.

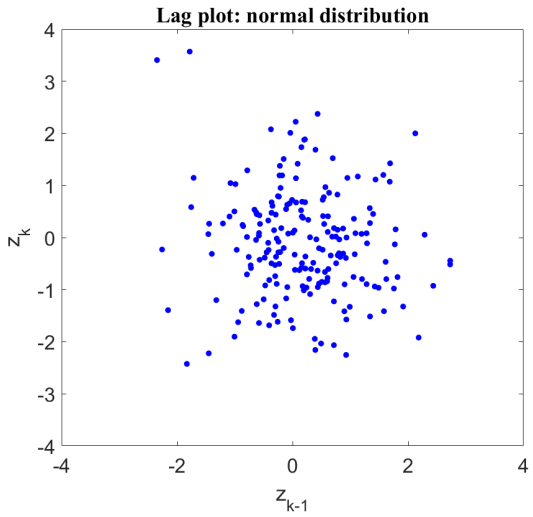


Figure 5.1.: Lag plot: normal distribution. Adapted from [79]

Next, consider a random walk – a process that describes a path consisting of a succession of random steps. In this case, the steps are defined via a centralized uniform distribution: $z_k = \sum_{i=j}^{k+j} \hat{U}_i$. The lag plot of the data is presented in Figure 5.2.

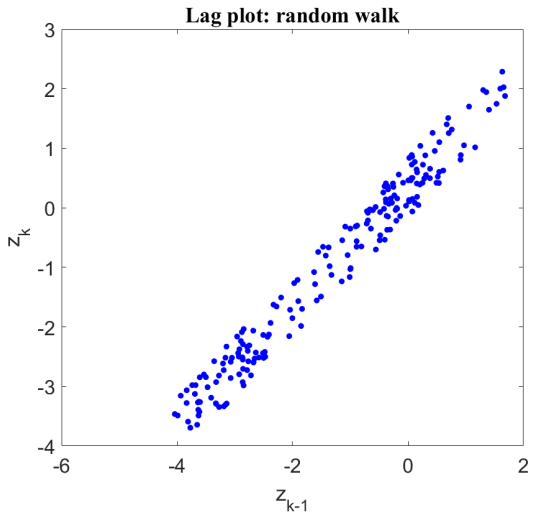


Figure 5.2.: Lag plot: random walk. Adapted from [79]

This lag plot allows concluding the following:

1. The data contain no outliers.
2. The data show strong autocorrelation.

The points in the plot are aligned diagonally, which corresponds to the strong positive autocorrelation of the data. Note, that nevertheless, the process is still not deterministic: the knowledge of a particular value of z_{k-1} does not provide the precise value of z_k. Unlike in the previous case, however, this knowledge provides a rather narrow corridor of possible values of z_k.

3. The data can be modelled with a linear model.

   In this case, a model of linear regression can be proposed: $z_k = a_0 + a_1 \cdot z_{k-1} + n_k$.

So, a simple tool such as a lag plot can provide useful information about the nature of the data series and even suggest a model or a class of models for the representation of the data.

5.1.1.2. Autocorrelation function

The autocorrelation function describes the correlation of a signal with the lagged signal (i.e., with a delayed copy of the original signal) as a function of delay.

The autocorrelation function is formally defined as

$$R_X(\tau) = E[X_t {X^*}_{t+\tau}], \tag{5.1}$$

where $E[\cdot]$ is the expectation operator, τ is the lag (the delay), and $\cdot^*$ refers to the complex conjugate of the process.

A related term is *autocovariance* function, which is defined as autocorrelation of the centralized processes:

$$K_X(\tau) = E[(X_t - \mu)(X_{t+\tau} - \mu)^*] = R_X(\tau) - \mu\mu^*. \tag{5.2}$$

Sometimes, especially in the context of exploratory data analysis, the term "autocorrelation" refers to autocovariance instead. Also, a *normalized* function is usually considered:

$$R_X(\tau) = \frac{K_X(\tau)}{\sigma^2} = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)^*]}{\sigma^2}, \tag{5.3}$$

where $\sigma^2 = K_X(0)$.

Following the notation established in the field, the function defined in (5.3) will be referred to as the autocorrelation function.

In the context of mathematical statistics and data analysis, the operations are performed over realizations (samples) of the stochastic process. Stochastic process $X(t)$ is represented by an observed realization x_k consisting of a finite number of components $k = \overline{1, N}$ measured at times t_k. The expectation operator $E[X(t)]$ corresponds to the averaging operator. Thus the corresponding sample autocorrelation function (assuming the process is real-valued) can be computed as

$$r_x(k) = \frac{\sum_{i=1}^{N-k} (x_k - \overline{x})\,(x_{i+k} - \overline{x})}{\sum_{i=1}^{N} (x_i - \overline{x})^2}, \tag{5.4}$$

where $\bar{\cdot}$ refers to averaging.

Note that this function (if the components are well-defined) produces values in the interval $[-1;1]$. The case $R = 1$ is usually referred to as *perfect correlation* and $R = -1$ as *perfect anti-correlation*.

To assess the characteristics of the original data series, the autocorrelation function can be computed, plotted and analyzed. To demonstrate this process, autocorrelation functions for the normal distribution and random walk (introduced in Subsection 5.1.1.1) are presented in Figures 5.3 and 5.4 respectively. The blue horizontal lines show the 95% *confidence bounds* or *confidence bands*. The plots of autocorrelation function are called *correlograms*.

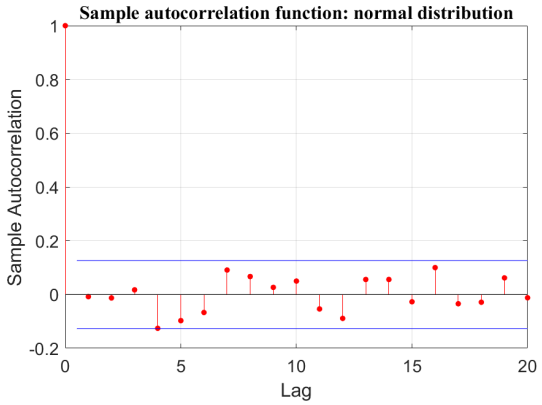


Figure 5.3.: Sample autocorrelation function: normal distribution

The sample autocorrelation function of the normally distributed process in Figure 5.3 can be described as not having any specific shape and (excepting, naturally, point with lag= 0) all fit within the 95% confidence bounds. This type of autocorrelation plot corresponds to random processes that don't show correlation for any value of lag.

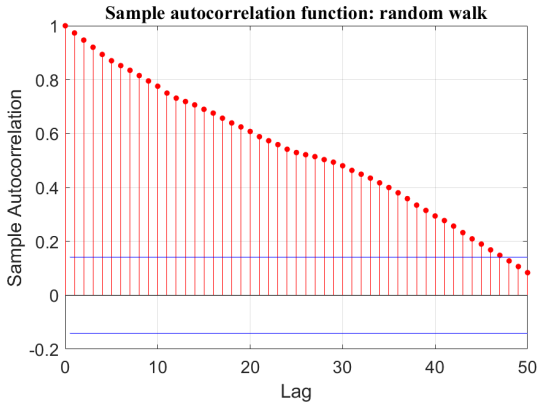


Figure 5.4.: Sample autocorrelation function: random walk

The sample autocorrelation function of a random walk in Figure 5.4 starts with $r = 1$ and then decays to 0. This behavior corresponds to a highly correlated process. Again, based

on this analysis a linear model can be proposed for the data because the plot shows a high correlation between the data on subsequent steps.

In [79, 223] the following choices of models are suggested based on correlograms or autocorrelation function values analysis:

- White noise model

  No recognizable shape/pattern; all values for lag>0 are within 95% confidence bands.

- Autoregressive model

  High correlation on all subsequent steps: values are decaying to 0 exponentially (possibly with alternating positive-negative values).

- Moving average model

  High values on several first lags, zero values afterward – high correlation only in few lags.

- Autoregressive moving average model

  A combination of the previous two observations, i.e., after several high peaks the values reduce gradually to 0, calls for a model that combines autoregressive and moving average models.

- Non-stationary model

  If the data does not decay to 0 with time, it cannot be represented with a stationary model.

The mentioned models are discussed in detail in Subsection 5.1.2.

Autocorrelation functions are often used in filtering to characterize correlated slowly drifting noises. Autocorrelation analysis helps determine the degree of autocorrelation of the studied random process that affects the IMU sensors and choose a respective model that can be used to filter the sensor signal.

5.1.1.3. Power spectral density

Power Spectral Density (PSD) is another powerful tool for analyzing a signal or time series. By applying a discrete Fourier transform to the series and obtaining Fourier coefficients, the distribution of energy in the frequency domain can be analyzed. For a finite-energy signal $X(t)$, the (two-sided) power spectral density is defined as:

$$S_X(\omega) = \left| \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} X(t) e^{-i\omega t} dt \right|^2 = \frac{\mathcal{X}(\omega)\mathcal{X}^*(\omega)}{2\pi}, \tag{5.5}$$

where ω is the frequency, $\mathcal{X}(\omega)$ and $\mathcal{X}^*(\omega)$ are the Fourier transform of $X(t)$ and its complex conjugate respectively. As can be seen directly from definitions, the two-sided PSD $S_X(\omega)$

and autocorrelation function $R_X(\tau)$ are related to each other by the Fourier transform (if the signal can be considered a wide-sense stationary random process [127]):

$$S_X(\omega) = \int_{-\infty}^{+\infty} e^{-i\omega\tau} R_X(\tau) d\tau, \tag{5.6a}$$

$$R_X(\tau) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i\omega\tau} S_X(\omega) d\omega. \tag{5.6b}$$

By using this relation between the two functions, the sample PSD can be easily computed by Fourier transform from the obtained sample autocorrelation function $r_x(k)$.

The two-sided PSD is a positive symmetric function in frequency ω. The single-sided PSD also often used in exploratory data analysis [79, 182] is defined as

$$S_X^{1S}(\omega) = \begin{cases} 2 \cdot S_X(\omega) & \text{if } \omega \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

Different noise types can be estimated in the power spectrum by comparing the PSD with a power function

$$S_X(\omega) \approx h \cdot \omega^{\alpha}, \tag{5.8}$$

and estimating the parameters α and h [23, 239]. Here the exponent α provides the information about the type of noise and the constant h shows the noise level.

As a tool for sensor data analysis, *log-log* plots of PSD are very often used in practice, because the visual representation helps identify various regions that correspond potentially to different noises. A plot of single-sided PSD provided by one of the IEEE Standards for accelerometers [126] is presented in Figure 5.5. Here the noise types/sources are specified in corresponding regions. It should however be noted that in real applications the slopes might not be equal to the provided typical values, and the transitions between the regions might be gradual rather than sharp [126].

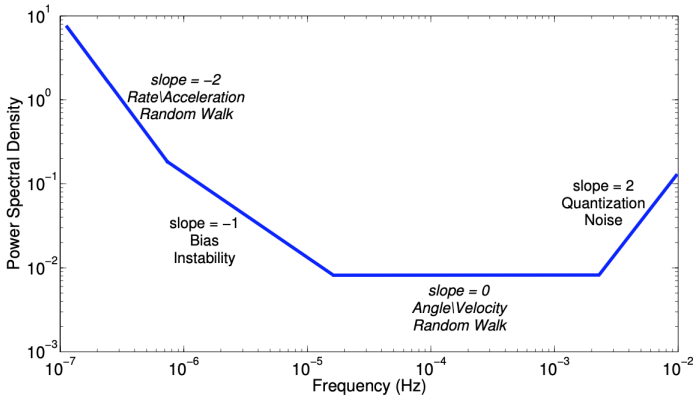


Figure 5.5.: Power spectral density: Noise types. Source: [126]

5.1.1.4. Allan variance

Allan variance test is another method that helps determine and characterize the noise terms in sensor data [127, 248].

For "truly" random processes the classical variance, usually used to estimate the divergence, is expected to decrease with the increase in the size of the sample set. In practice, though, the data obtained from sensors is often autocorrelated, meaning that the estimated variance can diverge with the increase of the number of data points [7]. To avoid that undesirable effect, Allan variance can be used instead.

The idea of the method is to use a high-pass filter to remove possible trends or fast fluctuations. To implement that, a given input $X(t)$ is divided into a set of $\{0 \dots K-1\}$ local windows of width τ. Note, that the method doesn't require the windows not to overlap. Then, for each window k the average $\bar{X}(t,\tau)_k$ is computed. Next, the mean squared difference across all chosen windows is computed $(\bar{X}(t,\tau)_k - \bar{X}(t,\tau)_{k-1})^2$ for $k \in \{0 \dots K-1\}$. As a function of window width τ the resulting variance is

$$\sigma^2(\tau) = \frac{1}{K} \sum_{k=0}^{K-1} \left(\bar{X}(t,\tau)_k - \bar{X}(t,\tau)_{k-1}\right)^2 . \tag{5.9}$$

So, the Allan variance is another tool that helps analyze the noise types, but this function (as compared to the frequency-space PSD) operates in the time domain: the window sizes represent the correlation times and the variance characterizes the noise with respect to them.

Note also, that the Allan variance is related to two-sided PSD by the following equation [259]:

$$\sigma^2(\tau) = 2 \int_0^{+\infty} S_X(\omega) \cdot \frac{\sin^4(\pi\omega\tau)}{(\pi\omega\tau)^2} d\omega. \tag{5.10}$$

As with the power spectral density, log-log plots of Allan variance can be used to classify various random errors [229, 164, 282]. A sample plot of $\sigma(\tau)$ is presented in Figure 5.6.

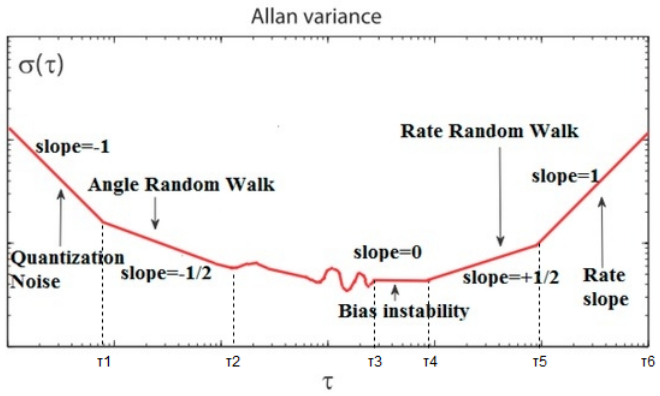


Figure 5.6.: Allan variance: noise types. Adapted from [282]

Relationship (5.10) is used to identify the noise types from the plot (or processed data series) as follows [23]:

- Quantization noise: slope $= -1$;
- Angle/Velocity random walk: slope $= -0.5$;
- Bias instability: slope $= 0$;
- Rate/Acceleration random walk: slope $= 0.5$;
- Rate slope: slope $= 1$.

5.1.2. Stochastic Models

As clarified before, random errors cannot be eliminated by calibration procedure, they need to be modelled with stochastic process models. To choose a specific model, different methods explained in the previous subsection can be used. This subsection is devoted to the description of the models themselves.

A common assumption for many time-series techniques is the *stationarity* of the data, i.e., the mean, variance, and autocorrelation structure do not change with time. If the original time series is not stationary, different techniques can be used to transform it to a stationary form [223], such as differencing the data, removing constant trends, stabilizing variance (by taking a logarithm or square root of the series), or excluding seasonality of the data. The series can also be transformed into the frequency domain for further analysis and modelling [45].

5.1.2.1. Autoregressive model

A common approach for modelling time series is the autoregressive model:

$$X_t = \delta + \alpha_1 X_{t-1} + \ldots + \alpha_p X_{t-p} + A_t, \tag{5.11}$$

where X_t is the time series, A_t a purely random process with 0 mean (white noise), and

$$\delta = \left(1 - \sum_{i=1}^{p} \alpha_i\right) \mu_i, \tag{5.12}$$

with μ_i denoting the process mean.

An autoregressive process represents a linear regression of the current value of the series X_t against prior values of the series $X_{t-1}, X_{t-2}, \ldots X_{t-p}$. Parameters $\alpha_1, \ldots, \alpha_p$ are the coefficients, also called lag-coefficients. Before these parameters can be estimated, the order of the autoregressive model, p, has to be determined. General techniques described in the previous section can be applied, or some specific tools like Bayesian information criterion, Akaike information criterion, or their implications/combinations can be used [59]. Then, for

estimation of parameters α_i, such techniques as least-square fitting, Yule-Walker equations [70], or Burg's method [35] can be implemented.

Equation 5.11 can be rewritten as

$$X_t - \alpha_1 X_{t-1} - \ldots - \alpha_p X_{t-p} = \delta + A_t. \tag{5.13}$$

With the use of a backshift operator $B: \ BX_t = X_{t-1}$, this equation can be represented as

$$(1 - \alpha_1 B - \alpha_2 B^2 - \ldots - \alpha_p B^p) X_t = \delta + A_t. \tag{5.14}$$

By introducing a characteristic polynomial $\phi(B) = 1 - \alpha_1 B - \alpha_2 B^2 - \ldots - \alpha_p B^p$, it can be further rewritten as

$$\phi(B) X_t = \delta + A_t. \tag{5.15}$$

The characteristic polynomial is an important tool that can be used for determining the stationarity of the process.

Two important special cases of autoregressive processes that are often used for IMU sensor error modelling are random walk and first-order Gauss-Markov process.

Random walk A random walk as briefly mentioned earlier is a process that is produced by an integration of uncorrelated signals, e.g., when the white noise is integrated:

$$X_t = X_{t-1} + A_t, \tag{5.16}$$

where X_t is the process at time step t and A_t is the white noise. Generally, random walks are non-stationary, because their covariance can increase with time. However, they can be transformed into stationary processes or just be considered within small time intervals, where their behavior is stationary.

As shown in Subsection 5.1.1, random walks are often present in IMU sensor data, because the original obtained data are integrated to get the object's pose.

Gauss-Markov process Generally, a Gauss-Markov process is a process that satisfies both Gauss and Markov requirements, namely, or a Gauss-Markov process $X(t)$ the following statements are true [185]:

- If $f(t)$ is a (non-zero) scalar function, then the product $f(t)X(t)$ is a Gauss-Markov process;
- If $g(t)$ is a non-decreasing scalar function, then the composition $X(g(t))$ is a Gauss-Markov process.
- If the process $X(t)$ is a non-degenerate and mean-square continuous, then it can be represented from a standard Wiener process $W(t)$ as follows: There exist a (non-zero) scalar function $f(t)$ and a strictly increasing scalar function $g(t)$ such that $X(t)$ can be represented as $X(t) = f(t)W(g(t))$.

A stationary Gauss-Markov process is called Ornstein-Uhlenbeck process. The first-order process with variance σ^2 and time constant β^{-1} has exponential autocorrelation

$$R_X(\tau) = \sigma^2 e^{-\beta|\tau|} \tag{5.17}$$

and power spectral density

$$S_X(\omega) = \frac{2\sigma^2\beta}{\omega^2 + \beta^2}. \tag{5.18}$$

Parameter β is the reciprocal of the process correlation time, i.e., for $\tau = \tau_c$ correlation function is $R_X(\tau) = \sigma^2/e$.

The plot of the autocorrelation function of first order Gauss-Markov process in presented in Figure 5.7. The function has a peak at $\tau = 0$ and it decays to 0 with $|\tau| \to +\infty$.

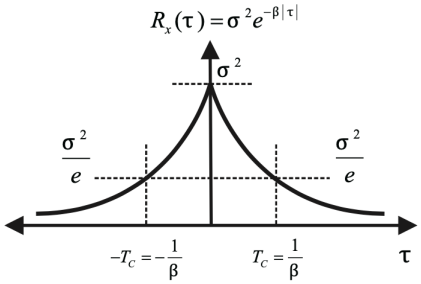


Figure 5.7.: Autocorrelation function of a first-order Gauss-Markov process. Source: [198]

For a discrete Markov process of the first order a finite difference equation can be written:

$$X_t = e^{-\beta|\tau|} X_{t-1} + A_t; \tag{5.19}$$

if the noise term A_t is Gaussian, then the equation above describes a Gauss-Markov first=order process.

A first-order Gauss-Markov process is often used for modelling the stochastic errors in IMU sensor data, especially for gyroscope and accelerometer bias instability. The choice of this model as compared to the general case of a first autoregressive process is that the coefficient α_1 is of a pre-known form, and parameter β can easily be estimated from the observed data.

Although in many applications this model provides an adequate representation of the noise, it should be noted that sometimes models of higher order need to be employed. For example, the sample autocorrelation functions computed from sensor data of three different IMUs in [198], presented in Figure 5.8, show clearly that a first-order Gauss-Markov process cannot provide a basis for proper noise estimation.

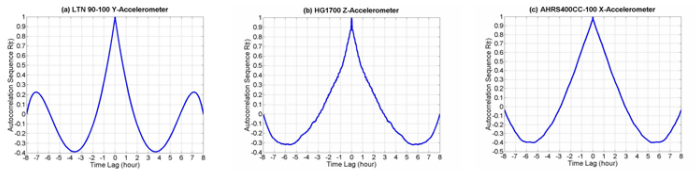


Figure 5.8.: Sample autocorrelation function for accelerometer data from 3 different IMUs. Source: [198]

5.1.2.2. Moving average model

Another common approach for time series modelling is the moving average model. The moving average model is a linear regression of the current value of the time series against the white noise applied to previous values of the series:

$$X_t = \mu + A_t - \lambda_1 A_{t-1} - \ldots - \lambda_q A_{t-q}, \tag{5.20}$$

where X_t is the time series, μ is the mean, A_i are white noise terms, and $\lambda_1, \ldots, \lambda_q$ are the parameters of the model. The value of q is the order of the moving average model.

With the use of the backshift operator, the characteristic polynomial of the model can be defined, which can be used for additional analysis of the series:

$$\theta(B) = 1 - \lambda_1 B - \ldots - \lambda_q B^q. \tag{5.21}$$

To determine the order of the model, autocorrelation plots and partial autocorrelation plots are usually used, although other graphical or analytical techniques also exist [79]. Then the parameters of the model of fixed order can be estimated from the observed data series. As the error terms in the moving average model are not observable, iterative non-linear fitting procedures have to be used to estimate the parameters, because regular linear list squares cannot provide a good estimation in this case.

5.1.2.3. Autoregressive moving average model

An approach proposed in [286] combines the autoregressive model and the moving average model to provide a description of the process through two polynomials, one for the autoregression and one or the moving average, respectively:

$$X_t = \delta + \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \ldots + \alpha_p X_{t-p} + A_t - \lambda_1 A_{t-1} - \lambda_2 A_{t-2} - \ldots - \lambda_q A_{t-q}, \tag{5.22}$$

where the terms of this equation are defined respectively in equations (5.11) and (5.20). The combined model is called Autoregressive Moving Average or ARMA. It is also sometimes referred to as Box-Jenkins Model, after the authors of the book [37], which popularized the approach. If the original data series is non-stationary, differencing can be performed within the ARMA model, and that produces the autoregressive integrated moving average model, ARIMA.

Since the ARMA model is a combination of two models, its order is defined by two values, (p, q). Both orders need to be estimated, the techniques mentioned in the respective sections can be used or specifically designed criteria like Bayesian or Akaike Information criteria. Note, that since the model is complex, it is generally harder to estimate the orders using graphical methods. Extended autocorrelation function can be constructed that allows determining both p and q simultaneously [39].

When the orders of the underlying models are chosen, the parameters of the models need to be estimated. As it is a complex model, the parameter estimation requires a moderately long observation series [46]. A modified Yule-Walker method can be used to estimate the ARMA model parameters [84]. Other approaches exist, e.g., an artificial neural network can be used for that purpose [50].

Once an ARMA model is constructed and fitted, a filter can then be developed, which includes this model to help remove random errors from the data.

5.1.3. Numerical Methods of Regression Analysis

Regression analysis is a branch of statistical modelling that focuses on estimating the relationships between dependent (outcome) variables and independent variables (also called predictors or features). A general regression problem can be formulated as follows: estimate the relationship between the dependent and independent variables that most closely fits the observed data. Most regression models assume that the dependent variable y_i is a function of the independent variables x_i and some unknown parameters β with additional error term e_i that can represent unmodelled determinants of the dependent variable or random noise:

$$y_i = f(x_i, \beta) + e_i. \tag{5.23}$$

Regression analysis started with the discovery of the least squares technique, published by A.M. Legendre [162] and several years later by C.F. Gauss [91]. In both works, the method was applied to a problem of celestial mechanics, namely, to the problem of determining the orbits of the comets and minor planets about the Sun. A decade later Gauss published another work on the topic [92] with further development of the theory that included assumptions which later became the base of the Gauss-Markov theorem.

The term "regression" itself was proposed by F. Galton in [87], where the author used this term to describe a biological phenomenon where the heights of descendants of tall ancestors tend to regress down towards a normal average. G.U. Yule [294] and K. Pearson [219] later extended the term to a more general statistical context.

In general, regression computations are only based on observed data and cannot be guaranteed to describe the stochastic process. However, the computed characteristics can be meaningful if some specific assumptions are satisfied, e.g.:

- The observation is representative of the potential population.
- Independent variables are measured precisely (i.e., with no error).

- Deviations from the model have an expected value of zero: $E[e_i|x_i] = 0$.
- The variance of error terms e_i is constant across all observations.
- The error terms e_i are uncorrelated.

By choosing appropriate assumptions, the estimators can be proven to have some desirable qualities. For example, the Gauss–Markov assumptions (see in the discussion later) guarantee that the estimates are unbiased, consistent, and efficient in the class of linear unbiased estimators.

5.1.3.1. Maximum Likelihood

Maximum likelihood estimation is a regression method that is based on the maximization of a likelihood function. The values of parameters of the model that maximize the sample likelihood are called maximum likelihood estimates.

Generally, the likelihood function $L_n(\beta)$ is defined for a set of the observed data samples $x = (x_1, \ldots, x_n)$ and a vector of parameters $\beta = (\beta_1, \ldots, \beta_k)$ that index the probability distribution within a parametric family $\{f(\cdot, \beta) \mid \beta \in B\}$ in parameter space B as follows:

$$L_n(\beta) = f_n(y, \beta). \tag{5.24}$$

The maximum likelihood estimate is therefore

$$\hat{\beta} = \arg\max_{\beta \in B} \hat{L}_n(\beta). \tag{5.25}$$

In practice, the logarithm of the likelihood function is often used (logarithm is a monotonic function, so the maxima of both functions are given by the same value of β):

$$l_n(\beta) = \ln L_n(\beta). \tag{5.26}$$

Both L_n and l_n are often called likelihood function, l_n is sometimes called log-likelihood. The advantage of using the logarithm is that it converts multiplications to sums. For independent identically distributed random variables $f_n(y, \beta)$ is the product of univariate density functions, and log-transformation allows for explicit computations in this case.

The necessary conditions for the extrema of the likelihood function (in case it is differentiable in β) are called likelihood equations:

$$\frac{\partial L_n}{\partial \beta_1} = 0, \quad \frac{\partial L_n}{\partial \beta_2} = 0, \quad \ldots, \quad \frac{\partial L_n}{\partial \beta_k} = 0. \tag{5.27}$$

Obviously, similar equations can be written down for the log-likelihood.

For some models, the likelihood equations can be solved explicitly to obtain $\hat{\beta}$, but in general, the closed-form solution can not be obtained and numerical optimization methods are used.

Maximum likelihood is a very general method that provides a consistent approach to regression and parameter estimation. The estimates are efficient, i.e., with the increase in sample size, maximum likelihood estimators become minimum variance unbiased estimators. The approach gives consistent results so that the sequence of estimates converges in probability to the estimated value [237].

On the other hand, as mentioned above, the likelihood equations have to be formulated specifically for every given estimation case (which is not a trivial problem in itself) and then the equations need to be solved, usually - numerically. And the numerical estimation, in general, is also non-trivial, requiring specialized high-precision numerical optimization methods. Another disadvantage is a highly biased behavior for samples of small length: the desirable qualities are only proven for the limit in the number of samples, and in practice, the method often shows bad results for small samples. Lastly, most of the numerical algorithms for the maximum likelihood approach are sensitive to the initial guesses, thus requiring specific methods for obtaining starting values.

All said, the maximum likelihood method remains a valid and commonly used approach, and some other regression analysis methods can be seen as subcases of it.

5.1.3.2. Least Squares

The least squares approach is a popular regression analysis method. The basic version, considered in this subsection, provides estimates for linear regression problems.

To put the problem in mathematical terms, suppose β is a constant but unknown n-element vector, and y is a k-element noisy measurement vector. Error in t^{th} reading is denoted as ϵ_t. The linear problem can then be formulated as

$$y_t = \sum_{i=0}^{n-1} h_{ti}\beta_i + e_t, \quad t \in [1,k]\,. \tag{5.28}$$

Equation (5.28) can be written as follows:

$$e_t^2 = \left(y_t - \sum_{i=0}^{n-1} h_{ti}\beta_i\right)^2, \quad t \in [1,k]\,. \tag{5.29}$$

The least squares estimator then is

$$\hat{\beta}_{LS} = \arg\min_{\beta} \sum_{t=1}^{k} e_t^2, \tag{5.30}$$

achieved by minimization of the respective least squares function

$$L_{LS}(\beta) = \sum_{t=1}^{k} e_t^2. \tag{5.31}$$

The errors that constitute the target function are equally weighted, so the error can be written as:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix} = y - H\beta, \tag{5.32}$$

where y is the vector of dependent variables and H is the matrix constructed from h_{ti}.

Then the least squares function is

$$L_{LS}(\hat{\beta}) = e^T e = \left(y - H\hat{\beta}\right)^\intercal \left(y - H\hat{\beta}\right) = y^\intercal y - \hat{\beta}^\intercal H y - y^\intercal H\hat{\beta} + \hat{\beta}^\intercal H^\intercal H\hat{\beta}. \tag{5.33}$$

Then the necessary optimality conditions are

$$\frac{\partial L_{LS}}{\partial \hat{\beta}} = -2 \cdot y^\intercal H + 2 \cdot \hat{\beta}^\intercal H^\intercal H = 0 \tag{5.34}$$

Then the optimizer is

$$\hat{\beta} = (H^\intercal H)^{-1} H^T y. \tag{5.35}$$

It should be noted that the least squares method can be seen as a subcase of the maximum likelihood approach. Indeed, the errors are considered to be independently distributed, and let's assume each of them has a probability density of $h(y)$. By assumption, the dependent variables y_i depend on the errors linearly, so they have the same distribution albeit with an offset: $h(y - f(x_i))$. Then the maximum likelihood function for y is the product of all the probability densities of its components, i.e.

$$L_n(y, \beta) = h(y_1 - f(x_1)) \cdot \ldots \cdot h(y_n - f(x_n)) = h(e_1) \cdot \ldots \cdot h(e_n). \tag{5.36}$$

Next, assume that the errors are distributed normally, i.e., $e_i \sim N(0, \sigma^2)$, then the likelihood function is

$$L_n(y, \beta) = \frac{1}{\sigma^n (2\pi)^{n/2}} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 \right\}, \tag{5.37}$$

and for any fixed σ the maximum of this function is obtained when the exponent's power is minimized:

$$\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 = \sum_{i=1}^{n} e_i^2, \tag{5.38}$$

which leads to the least squares approach.

5.1.3.3. Weighted Least Squares

A modification of the basic least squares algorithm assigns specific weights to the errors:

$$L_{WLS}(\hat{\beta}) = \sum_{t=1}^{k} \frac{e_t^2}{\sigma_t^2}. \tag{5.39}$$

Let R denote the covariance matrix of the noise:

$$R = E[e \cdot e^\intercal] = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k \end{bmatrix}, \tag{5.40}$$

then the objective can be rewritten as

$$L_{WLS}(\hat{\beta}) = e^\intercal R^{-1} e = (y - H\hat{\beta})^\intercal R^{-1}(y - H\hat{\beta}) = \\ = y^\intercal R^{-1} y - \hat{\beta}^\intercal H R^{-1} y - y^\intercal R^{-1} H\hat{\beta} + \hat{\beta} H R^{-\intercal} H\hat{\beta}. \tag{5.41}$$

The necessary optimality conditions:

$$\frac{\partial L_{WLS}}{\partial \hat{\beta}} = -2 \cdot y^\intercal R^{-1} H + 2 \cdot \hat{\beta}^\intercal H^\intercal R^{-1} H = 0. \tag{5.42}$$

Then the optimizer is

$$\hat{\beta} = \left(H^\intercal R^{-1} H\right)^{-1} H^\intercal R^{-1} y. \tag{5.43}$$

Note that the expression is similar to the regular least squares optimizer (Equation 5.35) but for the presence of the covariance matrix.

5.1.3.4. Recursive Least Squares

Although the approaches described above are powerful and are often used in practice, they have limitations when dealing with online computations, i.e. when the new measurements are obtained the least squares estimators and maximum likelihood estimators have to be reestimated from scratch. The recursive least squares technique allows estimating the new best value of $\hat{\beta}$ without recomputations.

Again, consider the linear model from Equation (5.28). Then, a linear recursive estimator can be written as

$$y_i = H_i \beta_i + e_i, \tag{5.44a}$$
$$\hat{\beta}_i = \hat{\beta}_{i-1} + K_i(y_i - H_i \hat{\beta}_{i-1}), \tag{5.44b}$$

where K_i is called the estimator gain matrix.

The principal goal is to minimize the residual:

$$r_i = \beta - \hat{\beta}_i = \beta - \hat{\beta}_{i-1} - K_i(y_i - H_i\hat{\beta}_{i-1}) = \\ = r_{i-1} - K_i(H_i\hat{\beta}_i + e_i - H_i\hat{\beta}i - 1) = r_{i-1} - K_i H_i(\hat{\beta}_i - \hat{\beta}_{i-1}) - K_i e_i = \\ = (I - K_i H_i) r_{i-1} - K_i e_i, \tag{5.45}$$

where I is an identity matrix.

Then the least recursive square target function is

$$L_{RLS,i}(\hat{\beta}) = E[||\beta - \hat{\beta}_i||^2] = E[r_i^\intercal r_i] = E[Tr(r_i r_i^\intercal)] = Tr(P_i), \tag{5.46}$$

where $Tr(\cdot)$ denotes the trace of the matrix and P_i is the estimation error covariance matrix:

$$P_i = (I - K_i H_i) P_{i-1} (I - K_i H_i)^\intercal + K_i R_i K - i^\intercal. \tag{5.47}$$

The optimality conditions:

$$\frac{L_{RLS,i}}{K_i} = -2 \cdot P_{i-1} H_i^\intercal + 2 \cdot K_i (H_i P_{i-1} H_i^\intercal + R_i) = 0. \tag{5.48}$$

The estimator gain matrix is

$$K_i = P_{i-1} H_i^\intercal (H_i P_{i-1} H_i^\intercal + R_i)^{-1}. \tag{5.49}$$

By some more massaging, the formulas for optimal K_i and respective P_i can be rewritten as follows [134]:

$$P_i = (P_{i-1}^{-1} + H_i^\intercal R_i^{-1} H_i)^{-1}, \tag{5.50a}$$

$$K_i = P_i H_i^\intercal R_i^{-1}. \tag{5.50b}$$

Note that these two equations require first computing the error covariance matrix P_i and then using it to update the estimator gain matrix K_i.

The algorithm for recursive least squares estimation can be written as follows:

1. Initialize the estimator:

$$\hat{\beta}_0 = E[\beta], \tag{5.51a}$$

$$P_0 = E[(\beta - \hat{\beta}_0)(\beta - \hat{\beta}_0)^\intercal]. \tag{5.51b}$$

   If there is no reliable information on β, assume $P_0 = [+\infty]$.

   If the value of β is known precisely, let $P_0 = [0]$.

2. For $i = 1, 2, ...$ iterate the following two substeps:
   a) Obtain a new measurement y_i, assuming that it follows Equation 5.44a, where the measurement noise terms are independent from step to step, have zero means and covariances R_i.
   b) Update the estimate of $\hat{beta}_i$ by Equation (5.44b) and the estimator gain K_i and the estimation-error covariance P_i by the pair of equations (5.49,5.47) or (5.50a,5.50b).

After providing a short review of the appropriate EDA topics, the next sections focused on filtering techniques can now be presented.

5.2. Frequency-based Filtering

This section is devoted to a short review of frequency-based filtering techniques. It should be noted that these filters alone often cannot provide a proper quality of filtering for sensor data, but they are commonly used as components of a larger advanced filter.

5.2.1. Low-pass filter

A low-pass filter is a linear operator that only lets *long-term* changes pass while filtering any short-term fluctuations. Consider a series of observations z_k measured respectively at times t_k, $k = 0, 1, ...$ and a respective sequence of filtered values w_k. To force changes to build up slowly (i.e., to implement the low-pass filter), every next filtered value can be defined as:

$$w_k = \alpha \cdot w_{k-1} + \beta \cdot z_k, \tag{5.52}$$

so that the new filtered output w_k is a linear combination of the previous filtered output w_{k-1} and the new non-filtered (measured) output z_k. It is also necessary to impose the condition $\alpha + \beta = 1$ so that the filtered value doesn't exceed the original one. Then the filter equation can be rewritten as

$$w_k = \alpha \cdot w_{k-1} + (1 - \alpha) \cdot z_k. \tag{5.53}$$

Parameter α in this formula defines the relationship between the "trust" the user puts into the filtered value as compared to the measured one. If the measurements are taken with a constant sampling time, a time constant of the signal τ which shows the relative duration of the signal can be defined as follows:

$$\tau = \frac{\alpha \cdot dt}{1 - \alpha}, \tag{5.54}$$

where dt is the (constant) time step of the sampling time sequence. For low-pass filters signals that are longer than the time constant pass through, while the signals shorter than the constant are eliminated by the filter. So, if the sample rate dt and the desired time constant τ are known, the low-pass filter coefficient α can be computed as

$$\alpha = \frac{\tau}{\tau + dt}. \tag{5.55}$$

Generally, accelerometers tend to suffer from high-frequency inputs. Therefore, low-pass filters are often suggested to stabilize accelerometer output by eliminating the high-frequency components. The disadvantage of this approach however is that low-pass filtering introduces lag into the sensor output sequence, which is an undesirable effect.

5.2.2. High-pass filter

A similar idea for eliminating low-frequency components of the signal is implemented in the high-pass filter. The filtering can be performed by subtracting the low-pass filtered series w_k from the original measured series z_k. By introducing h_k as a notation for the elements of the detrended sequence $h_k = z_k - w_k$ from the definition given in the previous subsection:

$$h_k = z_k - w_k \overset{(5.53)}{=} z_k - \alpha \cdot w_{k-1} - z_k + \alpha \cdot z_k. \tag{5.56}$$

Now, to get h_{k-1} in the right part of the equation, add and subtract $\alpha \cdot z_{k-1}$:

$$h_k = \alpha \cdot (h_{k-1} + z_k - z_{k-1}). \tag{5.57}$$

A large value of the filter parameter α implies a slow decay of the output and a strong influence from high-frequency components. A small value of the parameter implies, conversely, a quick decay, and only large changes in the measured data lead to changes in the filtered output. It should also be noted that in any scenario a constant input will decay to 0 when passed through a high-pass filter.

Generally, gyroscopes tend to suffer from time-dependent drift. High-pass filters can be used to eliminate that effect.

5.2.3. Complementary filter

The idea of complementary filters is to apply two "inverse" filters to the measured data and then bring the resulted filtered sequences into a single output. Let the input z_k be modelled as a sum of the true signal s_k and noise ν_k:

$$z_k = s_k + \nu_k. \tag{5.58}$$

Let there be two (independent) measured sequences of the same signal:

$$\begin{aligned} z_k^{(1)} &= s_k + \nu_k^{(1)}, \\ z_k^{(2)} &= s_k + \nu_k^{(2)}. \end{aligned} \tag{5.59}$$

The measured sequences are then passed *separately* to the filter, as shown in Figure 5.9.

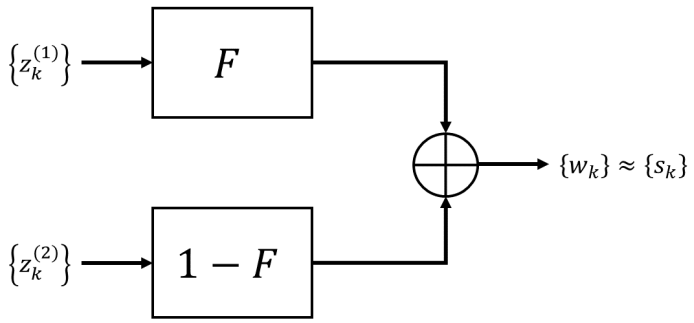


Figure 5.9.: Complementary filter for 2 measurements. Adapted from [21]

Note that regardless of the particular choice of inner filter F, the original (true) signal s passes the complementary filter without being modified.

The complementary filter can be considered as a very restricted Kalman filter (see Subsection 5.3.1). Although based on a very simple underlying idea, it is successfully used for inertial navigation systems and GPS location sensing [21].

5.2.4. Wavelet filter

A wavelet-based filter is a technique where a specifically designed low-pass filter is applied recursively to the input signal to get rid of the high-frequency components and highlight the low-frequency ones.

Since the main application considered in this work is sensor signal filtering, only discrete wavelet filters are considered. The underlying transform is, correspondingly, called *Discrete Wavelet Transform* or DWT. The theoretical foundations of wavelet analysis and specifically of its application to non-stationary processes were presented in [55, 183, 260].

Briefly, the technique consists of 3 stages:

1. Decomposition: transform the original signal into an object in wavelet space;
2. Filtering: apply a threshold in wavelet space;
3. Reconstruction: invert the filtered transform to obtain the filtered signal.

In the first stage, the original signal is decomposed into a basis of wavelet functions. To perform the decomposition, a set of wavelet functions is chosen, and convolution of these functions and the signal is computed. The result of this procedure is a set of coefficients, called wavelet coefficients. Then, the original signal can be represented as a linear combination of the wavelet functions weighted with the wavelet coefficients. The set of wavelet functions usually consists of scaled and shifted versions of one function, called the *mother wavelet*. If the mother wavelet $\psi(t)$ is chosen, the operations of translation and dilation that provide the wavelet basis are defined as follows:

$$\psi_{jk}(t) = a^{-0.5j}\psi\left(a^{-j}t - kb\right), \tag{5.60}$$

where $a > 1$ and $b > 0$. The discrete subset of the upper half-plane given by these operations is all the points $(a^j, k\, a^j b)$ for integer j, k.

If this set of wavelet functions (sometimes called *baby wavelets*) forms a tight frame on $L^2(\mathbb{R})$, then any signal finite energy $x(t)$ can be reconstructed as

$$x(t) = \sum_j \sum_k \alpha_{jk}\psi_j k(t), \tag{5.61}$$

where α_{jk} are the wavelet transform coefficients of $x(t)$, that can be computed as

$$\alpha_{jk} = \langle x, \psi_{jk} \rangle. \tag{5.62}$$

In signal analysis, a multi-resolution discrete wavelet transform is usually used. Typically, coefficients are chosen as $a = 2$ and $b = 1$. Additionally to the mother wavelet a scaling function $\phi(t)$ should be chosen, sometimes referred to as *father wavelet*.

Then, from the father wavelet, a subspace V_j is constructed as

$$V_j = \text{span}\left\{\phi_{jk} \;:\; k \in \mathbb{Z}\right\}, \quad \text{where } \phi_{jk}(t) = 2^{-0.5j}\phi(2^{-j}t - k), \tag{5.63}$$

and similarly, a subspace W_j is constructed from the mother wavelet:

$$W_j = \text{span}\left\{\psi_{jk} \ : \ k \in \mathbb{Z}\right\}, \quad \text{where } \psi_{jk}(t) = 2^{-0.5j}\psi(2^{-j}t - k), \tag{5.64}$$

such that the subspaces $\{W_j\}_{j\in\mathbb{Z}}$ are the orthogonal differences of $\{V_j\}_{j\in\mathbb{Z}}$, i.e., for every j the subspace W_j is the orthogonal complement of V_j inside V_{j-1}. It is also required that the sequence V_j forms a multi-resolution analysis of L^2.

So, the space V_j with sampling distance 2^j covers (approximately) the frequencies $[0; 2^{-j-1}]$ and its orthogonal complement, W_j covers $[2^{-j-1}; 2^{-j}]$.

Based on these sequences of subspaces, a sequence of quadrature mirror low-pass (h_k) and high-pass (g_k) filters can be defined as

$$h_k = \langle \phi_{0,0}, \phi_{1,k} \rangle, \tag{5.65a}$$
$$g_k = \langle \psi_{0,0}, \phi_{1,k} \rangle, \tag{5.65b}$$

that satisfies the identities

$$\phi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \phi(2t - k) \tag{5.66a}$$
$$\psi(t) = \sqrt{2}\sum_{k\in\mathbb{Z}} g_k \phi(2t - k). \tag{5.66b}$$

These identities form the basis for the fast wavelet transform algorithm that is usually used in signal filtering.

So, to construct a wavelet filter that suits the needs of sensor signal filtering, a pair of high-pass and low-pass filters are chosen as described before. Then these base-filters are applied to the signal, and the signal is "split" into approximation (from low-pass) and detail (from high-pass) parts, as shown in Figure 5.10.

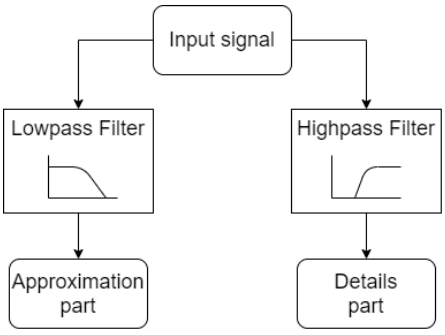


Figure 5.10.: Signal decomposition by discrete wavelet transform

Then, the approximation part can again be passed through the filter, and so on (as shown in Figure 5.11). This process is usually called *Wavelet Decomposition Tree*, and each layer is called *Level of Decomposition*.

As a result of applying a multi-level wavelet filter, a complex sensor input is decomposed at several levels of approximations, with wavelet transforms performed at each level.

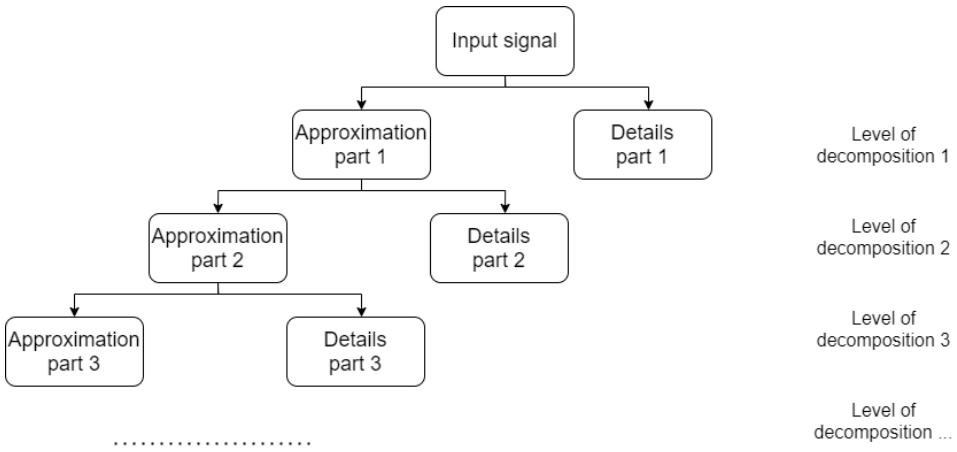


Figure 5.11.: Wavelet Multiple Level of Decomposition (Wavelet Decomposition Tree)

The idea of wavelet transforms is somewhat similar to Fourier transforms. The advantage of wavelets, however, is that they allow for variable time-frequency resolution (translation corresponds to time resolution, dilation corresponds to frequency), while Fourier transform is localized only in frequency. Therefore, wavelet filters are often used for non-stationary signals, where only high-frequency noise needs to be removed and the general behavior of the series needs to be preserved, including the high-frequency transitions at peaks [177]. It should also be noted that some classes of functions, including functions with sharp spikes, use substantially smaller wavelet bases as compared to the sine-cosine basis of Fourier transform.

5.3. Bayesian Filtering

Bayesian theory, first introduced by Thomas Bayes in [25], gained popularity with the work of Laplace [157]. After that, Bayesian statistics became one of the most important branches of statistics in general with applications in statistical decision making, pattern recognition, estimation, and machine learning [47]. In [117], the principles of Bayesian filtering are introduced and explored.

Consider a general continuous formulation of a stochastic filtering problem for a dynamic process [133]:

$$\dot{x}_t = f(t, x_t, u_t, w_t), \tag{5.67}$$

$$z_t = g(t, x_t, u_t, v_t). \tag{5.68}$$

Equation (5.67) is the *state* equation, with x_t representing the state vector. Vector u_t is the vector of system input, w_t represents the process noise. Equation (5.68) is the *measurement* equation, where z_t is the measurement (observation) vector and v_t is the measurement noise. Functions f and g are vector-valued function that can (potentially) depend on time.

The filtering problem is to get the "optimal" estimate of the true state x_t based on observations z_s, with $0 \leq s \leq t$ and controlled input u_t. Every continuous distribution is completely

described by its probability density function (pdf), therefore the goal is to find the posterior pdf. As for optimality of the filter, different criteria can be proposed to select the optimal filter from a considered class, e.g., minimum mean-squared error, maximum likelihood, minimum free energy, etc [47]. As the class of filters considered in this section is Bayesian filters, the criterion is that the posterior distribution, i.e., distribution obtained by filtering, integrates and uses all of the information provided by input probabilities.

Similarly to the continuous filtering problem (5.67-5.68), a discrete-time filtering problem can be defined as

$$x_{n+1} = f(x_n, u_n, w_n), \tag{5.69}$$

$$z_n = g(x_n, u_n, v_n). \tag{5.70}$$

The schematic of the generic discrete state-space model is presented in Figure 5.12.

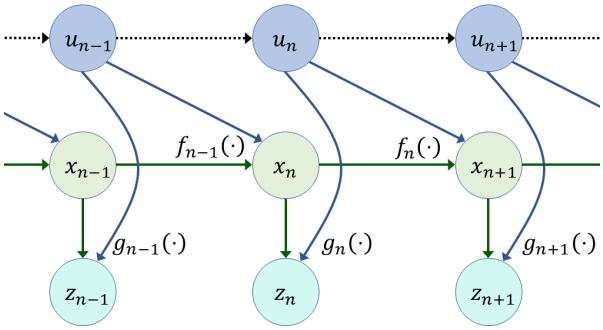


Figure 5.12.: Discrete state-space model. Adapted from [47]

It is possible to formulate a semi-discrete model, where the state equation is considered in continuous time, while the observations are provided in discrete time frames [153]. This model corresponds to many practical applications, where a sensor or similar device can only measure the process in discrete intervals, while the process itself has a continuous nature.

The solution of a generic nonlinear problem is a conditional probability distribution of the state, that is found using the observations (and, potentially, the controlled input). To get a *global* solution of a nonlinear filtering problem, a partial differential equation that describes the propagation of probability should be solved. Because of the complexity of the equations in the general case, an analytical solution is virtually impossible [47]. Therefore, the equations are usually only solved analytically for specific cases or under strict assumptions.

For example, if the problem is linear and the noise is Gaussian, then the (exact) optimal filtering solution is given by the (basic) Kalman filter (see Section 5.3.1.1).

Alternatively, the partial differential equations can be solved numerically.

Some techniques exist that approximate the posterior density with finite sums of the base function. For example, the Gaussian sum filter uses a weighted sum of Gaussian densities. This approach is based on two main ideas. First, any density can be reasonably approximated

by a (sufficient) number of Gaussian densities. Second, the individual expectations and covariances can be calculated, which makes the solution trackable.

Another group of approaches that is loosely based on the same idea of only computing some characteristics of the processes is the family of Kalman filters. The particular methods of this family differ by the approach they use to obtain and propagate the desired values.

The particle filter is another nonlinear filtering method (or rather a group of methods), which also avoids Jacobian matrix calculation. The underlying principle of this filter that distinguishes it from other methods is Monte-Carlo sampling that allows estimating the posterior distribution empirically from a set of samples.

In the next subsections, Bayesian-based filters are reviewed. The first group of methods, which are based on the Kalman filtering technique, are presented in Subsection 5.3.1. Subsection 5.3.2 is devoted to the description of the particle filter.

5.3.1. Kalman filters

Kalman filters family started with the invention of the basic linear discrete Kalman filter. Although named after Rudolf Kalman, the filter was first proposed by Swerling in [262] and only later independently formulated by Kalman in [139] and (with modification for continuous processes) by Kalman and Bucy in [138].

As mentioned earlier, the basic Kalman filter is used for linear problems with Gaussian distributed noise. As practical applications usually call for non-linear models, various methods based on the Kalman principle have been developed that allow to approximate and resolve the non-linear filtering problems.

Extended Kalman filter (see 5.3.1.3) linearizes state-space equations (5.69-5.70), and then employs the basic Kalman filter (assuming the posterior is Gaussian). This approach allows to tackle nonlinear problems but gives poor results if the process is highly non-Gaussian (or highly non-linear).

Another approach is to approximate the propagated densities using some characteristics of a set of propagated points. A class of approaches called Sigma-Point Kalman Filters exploits that idea while propagating the values from a small amount of specifically selected points [247]. This class includes the Unscented Kalman filter and its modifications, Central Difference Kalman filter, Quadrature Kalman Filter, Cubature Kalman Filter, and others. The filters that belong to this family differ in the particular choice of propagated sigma-points, their number, and assigned weights for them as well as for algorithms of covariance computations. The general idea, nevertheless, is the same in all of the methods of this class, so only one of them, the Unscented Kalman Filter is considered here (see 5.3.1.4).

5.3.1.1. Basic Kalman filter

As mentioned earlier, the (basic) Kalman filter is developed for linear discrete filtering problems with Gaussian noise, that is

$$x_{n+1} = Ax_n + Bu_n + w_n \tag{5.71}$$
$$z_n = Cx_n + v_n. \tag{5.72}$$

Here the notation corresponds to the one introduced above in equations (5.69-5.70), that is x_n is the current state of the process, u_n is the applied control, w_n is the system noise, z_n is the available measurement (observation), and v_n is the measurement noise. Both w_n and v_n are assumed to follow normal distributions with 0 mean and variances Q and R respectively. Processes w_n and v_n are assumed to be independent; in case that assumption doesn't hold, the system can be rewritten in such a way that the term corresponding to their covariance disappears [95].

Kalman filter is a predictor-corrector scheme, i.e., every step of the algorithm consists of 2 substeps, usually called *prediction* or *propagation* and *update* or *correction*.

The prediction state estimate is

$$\hat{x}_{n+1}^{-} = A\hat{x}_n^{+} + Bu_n, \tag{5.73}$$

where $\hat{\cdot}$ refers to the estimate of a value, superscripts $\cdot^{+}$ and $\cdot^{-}$ refer respectively to updated (posterior, corrected) estimate and predicted (prior, propagated) estimate.

The filter also propagates error covariance

$$P_{n+1}^{-} = AP_n^{+}A^{\intercal} + Q, \tag{5.74}$$

where the term P is the state error covariance, i.e., the error covariance of the estimate error as computed by the filter. Note, that the error covariance increases because of accumulating effect of adding Q at every step.

The update state also includes state estimation and error covariance computations, but first, the measurement residual (innovation) is computed as the difference between the true measurement and the estimated one:

$$r_{n+1} = z_{n+1} - C\hat{x}_{n+1}^{-}, \tag{5.75}$$

and also the Kalman gain is calculated:

$$K_{n+1} = P_{n+1}^{-}C^{\intercal}\left(R + CP_{n+1}^{-}C^{\intercal}\right)^{-1}. \tag{5.76}$$

Then the updated state estimate is computed through these two values:

$$\hat{x}_{n+1}^{+} = \hat{x}_{n+1}^{-} + K_{n+1}r_{n+1}, \tag{5.77}$$

and the error covariance is updated:

$$P_{n+1}^{+} = \left(I - K_{n+1}C\right)P_{n+1}^{-}, \tag{5.78}$$

where I is an identity matrix. Note that the updated error covariance is smaller than the predicted one, meaning that the filter provides a more precise estimation after the correction step.

Kalman filter is a recursive algorithm, and as such it requires an initialization: state estimate $\hat{x}_0^+$ and error covariance matrix P_0^+. After the initialization is chosen, the filtering is performed step-by-step, and the values on each step only depend on the values from the previous one, allowing for memory-efficient implementations.

In practice it is generally hard to estimate covariance matrices, so for P it is usually suggested to pick up a matrix with large values (that will gradually reduce with updates). As for the noise covariance matrices Q and R, a technique called autocovariance least-squares (ALS) [227] is often used for this purpose. Alternatively, a modification of the Kalman filter that automatically adjusts these matrices in real time [184] can be used. This modification called *adaptive* Kalman filter is discussed next.

5.3.1.2. Adaptive Kalman filter

As declared above, in the practical application of filtering techniques is can be difficult to determine the variance Q of the system noise w_n and variance R of the measurement noise v_n. Overall performance can be improved by filter tuning, i.e., by adjusting parameters of the filter by the measurements provided in a particular dataset. However, this procedure will have to be performed for every new dataset, which makes the approach not desirable.

Adaptive processing is usually performed by one of the two common approaches: multiple model or innovation-based.

In multiple model approach, a set of Kalman filters with different parameters is used simultaneously on the same model, and then hypothesis testing is performed on the result to determine the weights with which the filtering results are then summed. This requires excessive computational resources and cannot be considered a practical approach to tackling the problem.

In innovation-based adaptive filtering the covariances Q_n and R_n are estimated over time (which is specified by the time-series subindex) based on the innovation sequence (the sequence of measurement residual) r_n. Usually, the covariances are re-estimated not on every step, but rather over a chosen time interval of size N. The sample covariance matrix is then defined as

$$\hat{S}_n = \frac{1}{N} \sum_{j=j_0}^{n} r_j r_j^\intercal, \tag{5.79}$$

where $j_0 = n - N + 1$ is the first time-frame used for estimation $\hat{S}_n$ at frame n.

Then the filter statistical information matrices are updated as follows [195]:

$$\hat{Q}_n = K_n \hat{S}_n K_n^\intercal, \tag{5.80}$$

$$\hat{R}_n = \hat{S}_n - C_n P_n^- C_n^\intercal. \tag{5.81}$$

Innovation-based adaptive filtering provides better-quality results for problems where the variances are difficult to estimate without excessive additional computations. However, if

the problem is not linear or the noises are not Gaussian, this approach will not lead to an improvement as compared with the basic Kalman filter.

5.3.1.3. Extended Kalman filter

The practical problems usually cannot be described by linear models. Extended Kalman Filter (EKF) first linearizes the model at the current estimate and then uses the same procedures as the basic linear Kalman filter.

The state-space equations for a generic discrete model were already stated in (5.69-5.70). Here they are presented again in a modified form, with the noise term separated from the functions:

$$x_{n+1} = f(x_n, u_n) + w_n, \tag{5.82}$$

$$z_{n+1} = g(x_{n+1}) + v_{n+1}, \tag{5.83}$$

where the state noise w_n and measurement noise v_{n+1} are assumed to be Gaussian.

To linearize the model around the current estimate, the Jacobians are computed, i.e., the first-order partial derivatives of the vector-valued functions are taken with respect to the state vector:

$$F_n = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_n^+,\, u_n} \tag{5.84}$$

$$G_{n+1} = \left.\frac{\partial g}{\partial x}\right|_{\hat{x}^-}. \tag{5.85}$$

The filter algorithm is then following the scheme presented for the basic linear Kalman filter.

In the prediction step, the state is estimated as

$$\hat{x}_{n+1}^- = f(\hat{x}_n^+,\ u_n), \tag{5.86}$$

and the error covariance is estimated with the computed Jacobian:

$$P_{n+1}^- = F_n P_n^+ F_n^\intercal + Q. \tag{5.87}$$

In the correction step, the measurement residual is computed:

$$r_{n+1} = z_{n+1} - h\left(\hat{x}_{n+1}^-\right), \tag{5.88}$$

and the Kalman gain is computed using the Jacobian of the measurement function:

$$K_{n+1} = P_{n+1}^- G_{n+1}^\intercal \left(R + G_{n+1} P_{n+1}^- G_{n+1}^\intercal\right)^{-1}. \tag{5.89}$$

Then the updated state estimate is

$$\hat{x}_{n+1}^+ = \hat{x}_{n+1}^- + K_{n+1} r_{n+1}, \tag{5.90}$$

and the updated error covariance is

$$P_{n+1}^{+} = (I - K_{n+1} G_{n+1})\, P_{n+1}^{-}. \tag{5.91}$$

While in theory extended Kalman filter allows to solve nonlinear filtering problems with just small modification as compared to the linear Kalman filter algorithm, in practice computing Jacobians for every time frame is expensive. As a standard modification, the Jacobians can be updated after k steps instead of every step. However, this introduces an additional parameter into the algorithm, and the tuning of that parameter should be performed carefully.

It should also be noted that the extended Kalman filter performs well for continuous functions with small non-linearities. If, however, the function does not belong to the described type, the first-order Taylor approximation cannot give good results. Other similar approaches exist with the approximation of higher order [153], but their immediate drawback is the complexity of computing the partial derivatives.

5.3.1.4. Unscented Kalman filter

The Unscented Kalman Filter (UKF) is another filter from the Kalman filter that estimates the posterior probability in nonlinear systems. This method is based on a deterministic sampling technique called unscented transform proposed in [272]: a set of points is specifically selected from the probability distribution and propagated through the non-linear model. Then, using the characteristics of the propagated points, the posterior distribution is estimated (see Figure 5.13).

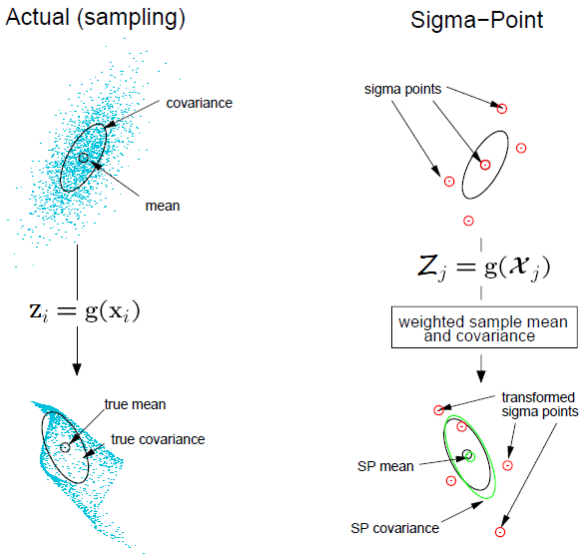


Figure 5.13.: A visualization of sigma-point propagation. Adapted from [187]

The UKF treats the non-linear state-space equations (5.69-5.70) in the following way. First, an *augmented* state vector x_n^a is composed (for convenience of notation) from the state vector x_n, the system noise w_n, and the measurement noise v_n as follows:

$$x_n^a = [x_n \quad w_n \quad v_n]^\intercal . \tag{5.92}$$

The initial value of the estimate $\hat{x}_0^a$ is set to

$$\hat{x}_0^a = E[x_0^a] = [\hat{x}_0 \quad 0 \quad 0]^\intercal, \tag{5.93}$$

where $\hat{x}_0$ is the expected value of the state vector at initial time frame. The initial value of the associated augmented state covariance, P^a, is chosen as

$$P_0^{a,+} = E\left[(x_0^a - \hat{x}_0^a)(x_0^a - \hat{x}_0^a)^\intercal\right] = \begin{bmatrix} P_0^{x,+} & 0 & 0 \\ 0 & Q_0 & 0 \\ 0 & 0 & R_0 \end{bmatrix}, \tag{5.94}$$

where P_0 is the (estimated) state covariance in the initial time frame, and Q_0 and R_0 are the variances of the system noise and measurement noise respectively.

Then, at every new frame n, a set of sigma points, $\mathcal{X}$, is generated from the augmented parameters as follows:

$$\begin{aligned} \mathcal{X}_{0,n} &= \hat{x}_n^{a,+} \\ \mathcal{X}_{i,n} &= \hat{x}_n^{a,+} + \left(\sqrt{(M+\lambda)P_n^{a,+}}\right)_i, \qquad i = 1, \dots M \\ \mathcal{X}_{i,n} &= \hat{x}_n^{a,+} - \left(\sqrt{(M+\lambda)P_n^{a,+}}\right)_{i-M}, \quad i = M+1, \dots 2M, \end{aligned} \tag{5.95}$$

where index i refers to the column number of the matrix square root, M and λ are the scaling parameters with $\lambda = \alpha^2(M+\kappa) - M$. Parameter α defines the spread of sigma-points around the mean and parameter κ allows for better scaling in some practical cases.

Each of the sigma-points brings a contribution to the estimation of expectation and covariance with a specific weight. The weights for the computation of the mean are defined as

$$\begin{aligned} W_0^{(m)} &= \frac{\lambda}{M+\lambda}, \\ W_i^{(m)} &= \frac{\lambda}{2(M+\lambda)}, \quad i = 1, \dots, 2M, \end{aligned} \tag{5.96}$$

and for computation of the covariance

$$\begin{aligned} W_0^{(c)} &= \frac{\lambda}{M+\lambda} + 1 - \alpha^2 + \beta, \\ W_i^{(c)} &= \frac{\lambda}{2(M+\lambda)}, \quad i = 1, \dots, 2M, \end{aligned} \tag{5.97}$$

where parameter β can be used to incorporate knowledge about the posterior distribution around the mean.

As evident from the definition of the sigma-points (5.95), the components of each of the points correspond to the components of the augmented state vector as follows:

$$\mathcal{X}_{i,n} = \left[\mathcal{X}^x_{i,n};\ \mathcal{X}^w_{i,n};\ \mathcal{X}^v_{i,n}\right]^\intercal. \tag{5.98}$$

Then, the Kalman filtering technique is employed with the defined sigma-points as follows. The prediction state begins with the propagation of the sigma-points:

$$\mathcal{X}^x_{i,n+1} = f\left(\mathcal{X}^x_{i,n}, u_{n-1}, \mathcal{X}^w_{i,n-1}\right). \tag{5.99}$$

The state vector is predicted as

$$\hat{x}^-_{n+1} = \sum_{i=0}^{2M} W_i^{(m)} \mathcal{X}^x_{i,n+1}, \tag{5.100}$$

and the covariance is predicted as:

$$P^{x,-}_{n+1} = \sum_{i=0}^{2M} W_i^{(c)} \left[\mathcal{X}^x_{i,n+1} - \hat{x}^-_{n+1}\right]\left[\mathcal{X}^x_{i,n+1} - \hat{x}^-_{n+1}\right]^\intercal. \tag{5.101}$$

In the correction step, the sigma-points are propagated through the model:

$$\mathcal{Z}^x_{i,n+1} = h\left(\mathcal{X}^x_{i,n+1}, \mathcal{X}^v_{i,n}\right). \tag{5.102}$$

After that, their mean

$$\hat{z}_{n+1} = \sum_{i=0}^{2M} W_i^{(m)} \mathcal{Z}_{i,n+1} \tag{5.103}$$

and covariance

$$S_{n+1} = \sum_{i=0}^{2M} W_i^{(c)} \left[\mathcal{Z}_{i,n+1} - \hat{z}_{n+1}\right]\left[\mathcal{Z}_{i,n+1} - \hat{z}_{n+1}\right]^\intercal \tag{5.104}$$

are computed.

Next, the $\mathcal{X}\mathcal{Z}$ cross-covariance is computed:

$$P_{xz,n+1} = \sum_{i=0}^{2M} W_i^{(c)} \left[\mathcal{X}^x_{i,n+1} - \hat{x}^-_{n+1}\right]\left[\mathcal{Z}_{i,n+1} - \hat{z}_{n+1}\right]^\intercal. \tag{5.105}$$

The Kalman gain is computed as follows:

$$K_{n+1} = P_{xz,n+1} S^{-1}_{n+1}. \tag{5.106}$$

Finally the current system state is updated:

$$\hat{x}^+_{n+1} = \hat{x}^-_{n+1} + K_{n+1}\left(z_{n+1} - \hat{z}_{n+1}\right), \tag{5.107}$$

and the current system covariance is computed as

$$P_{n+1}^{x,+} = P_{n+1}^{x,-} - P_{xz,n+1}K_{n+1}^{\intercal}. \tag{5.108}$$

The unscented Kalman filter doesn't require computation of Jacobian matrices or higher derivatives. It has been shown that for highly nonlinear systems UKF performs better than EKF. In the case of Gaussian noise, UKF gives third order of approximation, and in the case of non-Gaussian noise at least second order [187].

Although UKF has a lot of advantages, it should be noted that the implementation of the method should be performed carefully. When sigma-points are chosen in equation (5.95), a square root of the matrix has to be obtained. Generally, that operation is defined only for positive-definite matrices, and the matrix $P_n^{a,+}$ does not necessarily belong to this class because of accumulation of numerical errors [153]. Various techniques and implementations exist to stabilize the method that can, however, somewhat decrease the computational performance of the method [247, 153].

Adaptive fading unscented Kalman filter The adaptive fading unscented Kalman filter is a modification of the standard UKF. When the prediction of the filter is not accurate, the adaptive fading algorithm assigns it a smaller weight in the final prediction. The idea was proposed for basic linear Kalman filter in [161, 289].

First, the algorithm follows the same steps as the standard UKF. Then, when the sigma-points are propagated (5.102) and their expectation (5.103) and covariance (5.104) are predicted, the algorithm doesn't immediately continue with the computation of the local state. Instead, it performs a χ-test for Mahalanobis distance of the measurement estimation error $\widetilde{z}_n = z_n - \mathcal{Z}_n$.

The Mahalanobis distance of an observation x from a set of observations with mean μ and covariance matrix S is defined as

$$M(x) = \sqrt{(x-\mu)^{\intercal} S^{-1} (x-\mu)}. \tag{5.109}$$

The squared Mahalanobis distance of the measurement estimation error should follow the chi-square distribution with m degrees of freedom, where m is the dimension of measurement z_n:

$$M^2(\widetilde{z}_n) = \widetilde{z}_n^{\intercal} \cdot S_n^{-1} \cdot \widetilde{z}_n \sim \chi_{m,q}^2, \tag{5.110}$$

and if it holds that

$$\mathcal{P}[M^2(\widetilde{z}_n) \leq \chi_{m,q}^2] = 1 - q, \tag{5.111}$$

where q is the significance level, then the system does *not* have any modelling error and the algorithm continues with the standard unscented steps for this time frame.

If the equation (5.111) does not hold, then an adaptive fading factor γ_n is introduced into the predicted state covariance matrix $\widetilde{P}_n^{x,-} = \gamma_n P_n^{x,-}$ and error covariance matrix $\widetilde{S}_n = \gamma_n S_n$.

Then, the computations are repeated for the same time-frame starting from the prediction step with an updated fading factor until the Mahalanobis distance satisfies equation (5.111).

The adaptive fading factors are calculated recursively [103]:

$$\gamma_n^0 = 0 \tag{5.112}$$

$$\gamma_n^1 = 1 \tag{5.113}$$

$$\gamma_n^{t+1} = \gamma_n^t - \frac{\left[M_n^2(\widetilde{z}_n|t) - \chi_{m,q}^2\right] \cdot [\gamma_n^t - \gamma_n^{t-1}]}{M_n^2(\widetilde{z}_n|t) - M_n^2(\widetilde{z}_n|t-1)}, \tag{5.114}$$

where $M_n^2(\cdot|t)$ refers to squared Mahalanobis distance evaluated on adaptive iteration t of this time frame.

5.3.2. Particle Filter

The particle filter, or, more correctly, particle filters are a family of Bayesian filtering techniques based on sequential Monte Carlo sampling. Particle filters show good results on non-linear non-Gaussian problems, providing sub-optimal but accurate and reliable results for complex problems with non-linear system dynamics and measurement noise.

To describe the algorithm, recall the non-linear generic formulation of the filtering problem (5.69,5.70). For simplicity of notation, this section does not include control vector u (although naturally the control can also be included in the filter); therefore the system is described by the following two equations:

$$x_n = f_n(x_{n-1}, w_{n-1}), \tag{5.115}$$

$$z_n = g_n(x_n, v_n). \tag{5.116}$$

The filtering problem is to recursively calculate an estimate of the state vector x_n at time n based on observations $z_{1:n}$ (available until time-step n). Therefore, the goal is to construct the probability density function $p(x_n|z_{1:n})$. The prior pdf is assumed to be available without measurements: $p(x_0|z_0) \equiv p(x_0)$ and the new estimations are obtained recursively via predictions and updates.

Let $\left\{x_{0:n}^i, w_n^i\right\}_{i=1}^{N_s}$ denote a random measure that characterizes the posterior probability density function $p(x_{0:n}|z_{1:n})$. Here $x_{0:n}^i$ are support points with associated normalized weights $w_n^i : \sum_i w_n^i = 1$. The posterior density at step n can then be approximated with a Dirac delta-measure as

$$p(x_{0:n}|z_{1:n}) \approx \sum_{i=1}^{N_s} w_n^i \delta(x_{0:n} - x_{0:n}^i). \tag{5.117}$$

The weights are chosen with a specific principle called importance sampling [62]. The principle can be shortly summarized as follows: given a probability density $p(x)$ that is equal up to a constant to another pdf $p(x) \propto \pi(x)$. Suppose it is difficult to sample points from $p(x)$ itself, but $\pi(x)$ can be evaluated. Let $x^i \sim q(x)$, $i = \overline{1, N_s}$ be samples from some density. Then a weighted approximation of $p(x)$ is given by

$$p(x) \approx \sum_{i=1}^{N_s} w^i \delta(x - x^i), \tag{5.118}$$

where the normalized weight of ith sample is

$$w^i \propto \frac{\pi(x^i)}{q(x^i)}. \tag{5.119}$$

Density $q(x)$ is called the *importance density*, and the samples x^i are called *particles*.

Returning to Equation (5.117), if the samples $x^i_{0:n}$ were drawn from an importance density $q(x_{0:n}|z_{1:n})$, then the weights are defined as

$$w^i_n \propto \frac{p(x^i_{0:n}|z_{1:n})}{q(x^i_{0:n}|z_{1:n})}. \tag{5.120}$$

The idea of sequential estimation is to approximate $p(x_{0:n}|z_{1:n})$ at step n, already having an estimate from the previous step $p(x_{0:n-1}|z_{1:n-1})$. With an importance density of a factorizable form

$$q(x_{0:n}|z_{1:n}) = q(x_n|x_{0:n-1}, z_{1:n}) \cdot q(x_{0:n-1}|z_{1:n-1}), \tag{5.121}$$

samples for the new step $x^i_{0:n} \sim q(x_{0:n}|z_{1:n})$ can be obtained by augmenting the existing samples $x^i_{0:n-1} \sim q(x_{0:n-1}|z_{1:n-1})$ with the new state $x^i_n \sim q(x_n|x_{0:n-1}, z_{1:n})$. By expressing $p(x_{0:n}|z_{1:n})$ in terms of $p(x_{0:n-1}|z_{1_n-1})$, $p(z_n|x_n)$, and $p(x_n|x_n - 1)$, the weight update can be shown to be [13]

$$w^i_n \propto \frac{p(z_n|x^i_n) \cdot p(x^i_n|x^i_{n-1}) \cdot p(x^i_{0:n-1}|z_{1:n-1})}{q(x^i_n|x^i_{0:n-1}, z_{1:n}) \cdot q(x^i_{0:n-1}|z_{1:n-1})} = w^i_{n-1} \frac{p(z_n|x^i_n) \cdot p(x^i_n|x^i_{n-1})}{q(x^i_n|x^i_{0:n-1}, z_{1:n})}. \tag{5.122}$$

If further $q(x_n|x_{0:n-1}, z_{1_n}) = q(x_n|x_{n-1}, z_n)$, then the importance density is only dependent on the state vector on the previous step x_{n-1} and current measurement z_n (i.e., the path created by the state vector does not affect the sampling and can be discarded). The modified weight update, in this case, is

$$w^i_n \propto w^i_{n-1} \frac{p(z_n|x^i_n) \cdot p(x^i_n|x^i_{n-1})}{q(x^i_n|x^i_{n_1}, z_n)}, \tag{5.123}$$

and the posterior can be approximated as

$$p(x_n|z_{1:n}) \approx \sum_{i=1}^{N_s} w^i_n \delta(x_n - x^i_n). \tag{5.124}$$

In principle, particle filtering consists of recursive propagation of the support points and their weights. However, this simplistic sequential importance sampling has a major degeneracy problem: the variance of the importance weights increases over time [62] and after a few iterations all the weight accumulates in one particle, i.e., all the particle except one have negligibly small weight and thus negligibly small contribution to the posterior approximation. Degeneracy of the algorithm can be quantified by the effective sample size [31]:

$$N_{eff} = \frac{N_s}{1 + \mathrm{var}(w^{*i}_n)}, \tag{5.125}$$

where var$(\cdot)$ is variance and the "true" weights are defined as

$$w_n^{*i} = \frac{p(x_n^i|z_{1:n})}{q(x_n^i|x_{n-1}^i, z_n)}. \tag{5.126}$$

Small values of N_{eff} indicate severe degeneracy. As the effective sampling size N_{eff} itself cannot be evaluated exactly, an estimate can be used

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s} \left(w_n^i\right)^2}. \tag{5.127}$$

There are two main strategies that help avoid degeneracy: specific choice of importance density and resampling.

A good choice of importance density can minimize the variance of the true weights so that the effective sample size is maximized. The optimal importance density function can be shown to be constructed as follows [62]:

$$q_{optimal}(x_n|x_{n-1}^i, z_n) = \frac{p(z_n|x_n, x_{n-1}^i) \cdot p(x_n|x_{n-1}^i)}{p(z_n|x_{n-1}^i)}. \tag{5.128}$$

The two major drawbacks of this importance density are as follows. First, the pdf $p(x_n|x_{n-1}^i, z_n)$ needs to be sampled from. Second, the weight update with the defined optimal importance density is

$$w_n^i \propto w_n^i p(z_n|x_{n-1}^i) = w_{n-1}^i \int p(z_n|x_n') p(x_n'|x_{n-1}^i) dx_n' \tag{5.129}$$

and requires evaluation of the interval. Both of the required operations are non-trivial and cannot be resolved in the general case. The sampling and integration are available if x_n is a member of a finite set or if the measurements are linear (but the dynamics of the system can still be nonlinear) [13]. In other cases, suboptimal approximations of the optimal importance density functions are constructed (with linearizations, unscented transforms, etc.). Alternatively, the filter can be constructed by choosing the importance density to be the prior, i.e. $q(x_n|x_{n-1}^i, z_n) = p(x_n|x_{n-1}^i)$. Other approaches exist leading to new versions of particle filters.

The second technique that helps reduce the effects of degeneracy is resampling: when the effective sample size is lower than some chosen threshold, the particles with the lowest weights are eliminated from the solution. Various resampling schemes exist (e.g., systematic resampling [146], stratified resampling [173], residual resampling [173], multinomial resampling [116], etc.)

Resampling reduces the effects of degeneracy but it can also introduce different problems: the particles with high weights are statistically selected more times, leading to a loss of diversity in sampling. Some techniques exist that address the problem, leading to yet more particular filters of the particle family.

"He was constantly reminded of how startlingly different a place the world was when viewed from a point only three feet to the left."

— Douglas Adams

6. Registration

Registration is finding the spatial relationships between real and virtual worlds as well as determining of transformation between the coordinate systems represented by sensors, markers, or other physical objects. The quality of registration has a considerable influence on the overall system performance because it forms a base for proper fusion. The transformations between the IMUs or IMUs to other trackers need to be determined in order to transform all measurements to a joint coordinate system and then fuse the data.

In the research community, the word *calibration* is often confusingly used instead of registration such as hand-eye-, absolute-orientation- and tip-calibration [141]. Although the methods calculate the transformation between different sensors (which is the definition of registration), they are insistently called *calibration*.

Appropriate calibration, filtering, and registration of sensors allow minimizing or completely removing the errors in both tracking and visualization in Mixed Reality applications. However, when the MR application setup is changed or more sensors are added to the setup, the registration procedure needs to be repeated. That leads to a requirement for automatized registration techniques that can allow automatic adjustment in the software once the hardware part is changed.

The following standard classification of registration techniques was proposed in [34]:

- Tracker-based registration;
- Knowledge-based registration;
- Computer vision-based registration.

The major flaw of this classification is that it doesn't form a proper partition: computer vision-based registration technology is a part of tracker-based registration technology.

An alternative classification can be proposed as follows:

- Tracker-based registration:
  The registration is performed by sensors:
  - Mathematics- and physics-based registration: employing sensor characteristics.
  - Target-based registration: employing a target.

- Knowledge-based or Model-based registration
  Employing a specifically designed and constructed environmental model, i.e., the trackers are fixed on the equipment with a known structure to ensure the position and direction.

The main focus of this work is on Multi-IMU systems, thus the question of IMU to IMU registration is the focus of the first section of this chapter. Another point of interest is the registration of IMU to optical tracking systems. First, IMUs are often used in practice together with optical trackers, especially in MR applications, and second, optical tracking systems can provide a reference on data enhancement stages or for evaluating the quality of IMU-based tracking. Therefore, the second section of this chapter is devoted to the registration of optical trackers to IMUs.

6.1. IMU to IMU Registration

Because the main focus of this work is Multi-IMU systems, this section is devoted specifically to the registration of one IMU to another. Each IMU measures motion in its own coordinate system. To fuse the measurements of several IMUs, these measurements need to be transformed into a joint coordinate system. Different approaches exist that provide this transformation.

There are three methods for estimating the registration matrix (the methods are classified according to the system proposed earlier):

- Tracker-based registration:
  - registration by tip-calibration:
    an interactive method, where an engineer is pointing with a tracked device at three or more specified positions on each IMU [131, 130].
  - registration by using the absolute orientation:
    an interactive method, where the physical facts are used to estimate the orientation between multiple IMUs or between IMU and other sensors.
- Knowledge-based registration
  - registration by the use of CAD-model (Computer-Aided Design) information:
    an automated method that expects the IMUs to have been installed according to a planned arrangement (with known poses).

So, for the positional registration, the data obtained from IMUs are usually enhanced by external data (e.g., a construction plan of the setup). For an accurate registration, prior calibration of IMU components should be performed, otherwise, the registration will be inaccurate because of the presence of deterministic errors that cannot be distinguished from registration errors after the registration stage.

Further in this section, IMU to IMU registration methods are presented. Subsections 6.1.1 and 6.1.2 explain the inter-registration methods for several accelerometers and several gyroscopes respectively.

6.1.1. Accelerometer to accelerometer registration

In a rigid Multi-IMU setup, each accelerometer ACC_j, $j \in [0, M]$, is related to a reference ACC_0 by a fixed rotation matrix R_j. To determine R_j, the fact that accelerometer measurements are sensitive to gravity is exploited: vector $\vec{g_{i,j}}$ can be uniquely determined for each ACC_j in a series of static measurements $i \in [0, N]$, taken in different poses of the Multi-IMU setup. Since the gravity vector is unique, the following property holds for each ACC_j across all of these measurements $i \in [0, N]$

$$\vec{g_{i,j}} = R_j \cdot \vec{g_{i,0}} \qquad \forall i \in [0,N] \; ; \forall j \in [1,M], \tag{6.1}$$

where R_j is a 3×3 rotation matrix and vector $\vec{g_{i,j}} = \begin{pmatrix} g_x & g_y & g_z \end{pmatrix}$ is the gravity vector expressed in the body coordinate system of ACC_j by the measurement i with the absolute value of gravitational constant $\|\vec{g_{i,j}}\|_2 = g = 9.81 \; m/s^2$.

Rotation R_j of ACC_j can be calculated by minimizing the following equation:

$$L(R_j) = \sum_{i=0}^{N} \left\| \vec{g_{i,j}} - R_j \cdot \vec{g_{i,0}} \right\|^2. \tag{6.2}$$

The same equation can be written in quaternions:

$$L(q_j) = \sum_{i=0}^{N} \left\| \begin{pmatrix} 0 & \vec{g_{i,j}} \end{pmatrix} - q_j \otimes \begin{pmatrix} 0 & \vec{g_{i,0}} \end{pmatrix} \otimes q_j^{\star} \right\|^2, \tag{6.3}$$

where $q_j = (q_w, q_x, q_y, q_z)$ is the quaternion (see Section B.3) representation of rotation R_j and $q_j^{\star}$ is the conjugate of q_j.

A quaternion has 4 unknown parameters, therefore at least 4 measurements are needed to define 4 equations for finding the unknowns. In practice, however, the system should be over-determined to get reliable results, i.e., at least 5 different static measurements are needed to solve the system.

Figure 6.1 shows a set of five different static positions that can be used to collect a sufficient description of the gravity vector in two different coordinate systems (ACC_0, ACC_1).

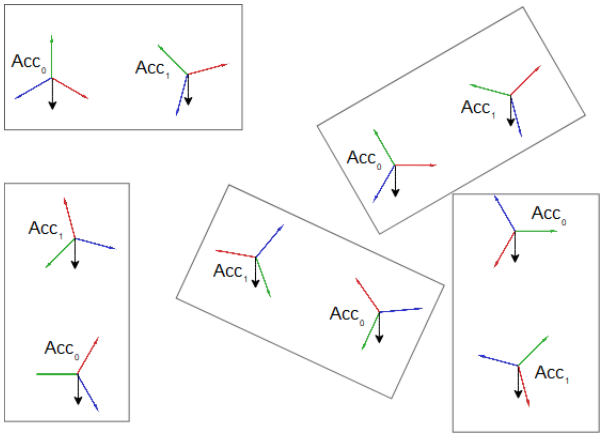


Figure 6.1.: Static positions for accelerometer registration. Gravity is shown with black arrows

6.1.2. Gyroscope to gyroscope registration

In some IMUs the accelerometers share the same coordinate system as the gyroscopes, in which case the registration method described in Section 6.1.1 can be used.

Alternatively, registration can also be performed based on gyroscope data. The only difference is that the data must be collected while the setup is being *rotated* rather than static, in order for a non-zero angular rate vector to be measured.

Let $\vec{\omega_{i,j}}$ be the representation of the angular rate in the $Gyro_j$'s coordinate system during the rotation measurement i.

The angular rates $\vec{\omega_{i,j}}$ and $\vec{\omega_{i,0}}$ describe the same rotation vector but in different coordinate system. Then through a rotation matrix, these rates are connected as

$$\vec{\omega_{i,j}} = R_j \cdot \vec{\omega_{i,0}} \qquad \forall i \in [0,N] \; ; \forall j \in [1,M] \; , \tag{6.4}$$

where R_j is a rotation matrix, N the number of dynamic measurement and M the number of gyroscopes.

Rotation R_j of $Gyro_j$ can be calculated by minimizing the following equation:

$$L(R_j) = \sum_{i=0}^{N} \left\| \vec{\omega_{i,j}} - R_j \cdot \vec{\omega_{i,0}} \right\|^2 . \tag{6.5}$$

As in the accelerometer-based registration, rotation can be expressed in quaternions:

$$L(q_j) = \sum_{i=0}^{N} \left\| \begin{pmatrix} 0 & \vec{\omega_{i,j}} \end{pmatrix} - q_j \otimes \begin{pmatrix} 0 & \vec{\omega_{i,0}} \end{pmatrix} \otimes q_j^\star \right\|^2 , \tag{6.6}$$

where $q_j = (q_w, q_x, q_y, q_z)$ is the quaternion representation of the rotation R_j and $q_j^\star$ is the conjugate of q_j.

For reliable registration, at least five different angular rate measurements are needed to solve the equation.

Figure 6.2 shows five different rotation steps that can be used to collect a sufficient number of measurements of an angular rate vector in two different coordinate systems ($Gyro_0, Gyro_1$). The user should provide a rotation that can be sensed by the gyroscope.

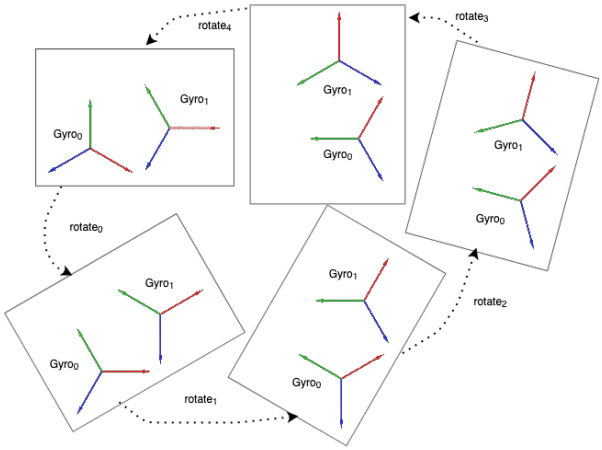


Figure 6.2.: Five different positions and rotations for gyroscope registration. The dotted lines represent different rotations

6.2. Optical Tracker to IMU Registration

Optical tracking is tracking by the use of a camera that makes use of visual information to track the user/the object. In Section 2.3.2 this type of tracking was explained and the two types of optical tracking were presented: outside-in (OSI) and inside-out (ISO). In the following subsections, the registration of IMUs to each of these systems is explained respectively.

6.2.1. ISO to IMU Registration

As explained earlier, ISO is the system where the tracking devices are moving and detecting the static features outside. This section explains how an ISO system (e.g., a head-mounted display) can be registered with an IMU.

The ISO system consists of one or several cameras. After an IMU is registered to a single camera from the setup with the cameras preregistered with each other, IMU registration is easily propagated to the remaining cameras with simple linear algebra.

The types of registration of an IMU to a single-camera ISO can be divided into two types: marker-based and marker-less. The following sections are devoted to these types of registration respectively.

6.2.1.1. Marker-based Registration

In this section the approaches that rely on external markers such as checkerboard patterns are introduced.

Early efforts in determining the spatial relationship between an IMU and a camera were made in [8].

As already mentioned in Chapter 4 in relation to this work, the authors used a pendulum for calibration of their IMU. For the registration, they calculated only the rotation between the two frames of reference by moving the system and observing the vertical direction with both optical and inertial sensors. For assistance in this procedure, the authors used a checkerboard. It should be noted that the authors refer to registration as a transformation between visual and inertial sensors.

Paper [156] presented another method for registration of the rotation between vision-based and inertial coordinate system. In particular, the method is based on the evaluation of rotation differences in vision reference coordinate system and rotation differences in inertial reference coordinate systems. The idea is to rotate the system consisting of a camera and a rigidly attached IMU, and then estimate the rotation from the data obtained by the camera. At the same time, the rotation is also estimated by integrating the angular rate measured by the gyroscope. Before the registration is performed, the IMU is proposed to be calibrated separately. The paper provides no information on the characteristics of the camera and no synchronization is performed because the computations are performed offline. The paper also suggests the use of their method for the registration of an outside-in optical system to IMU. The setups are shown in Figure 6.3.

Figure 6.3.: Hybrid setups: inside-out tracker (left) and outside-in tracker target board (right). Source: [156]

Research [174] continues on the same topic proposing a two-step approach: first, the relative *orientation* of the IMU and the camera is determined, and then a turntable is used to determine their relative *position*. The authors make use of the gravitation vector, measured by the accelerometers, and a vertical calibration pattern to estimate the rotational alignment of the inertial-visual system. The translation between the devices is then estimated using a turntable.

The assumption that makes the joint system observable is that the accelerometers lie in the center of the turntable so that they become the center of rotation (see Figure 6.4). In this way, simplified equations known from hand-eye calibration can be used to solve the system for the translation. While this approach has the advantage of being static and does not need

dynamic motions, the necessary system setup is rather cumbersome and error-prone.

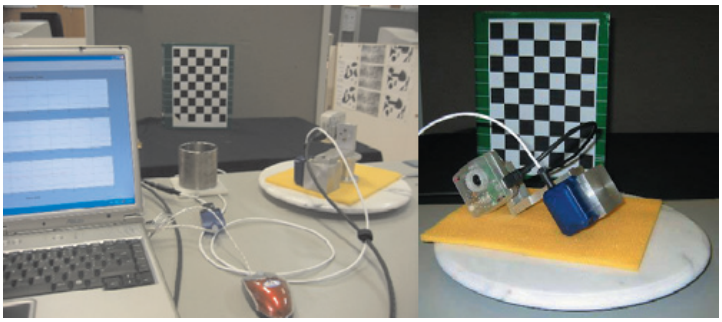


Figure 6.4.: The registration setup with a turntable. Source: [174]

A large group of approaches employs filters of the Kalman group for registration. The most popular solution is to use an extended or unscented Kalman filter (EKF, UKF) to estimate the sensor alignment. Approaches of this category implement the following states in their filter: position, orientation, linear and angular velocity, the bias of the gyroscopes, bias of the accelerometer, and finally translation and rotation between IMU and camera. If the calibration step is done offline, the computational overhead of the UKF becomes irrelevant, while it provides a higher-order approximation than the EKF and should, thus, be preferred.

The authors of [193] register the camera and IMU by fusing the corner locations of a checkerboard in the camera images with the measurements of the gyroscopes and the accelerometers. For that, they use an error-state (indirect) Kalman filter. An initialization stage to find good start values precedes the filtering. Further, they also explore the observability of the Kalman filter, with the result that only rotations in two degrees of freedom are necessary to estimate the IMU camera transformation. As a benchmark for their experiments, the authors use the results of the batch least square method as the ground truth values are not available in the real-life setup.

It should be noted, that although the authors stated that they used a camera with a refresh rate of 3.75 *Hz* and an IMU with frequency 100 *Hz* (i.e., the IMU delivers roughly 30 measurements while the camera has only an image captured), they do not explicitly explain how they synchronize the obtained measurements from two different systems. It should also be noted that the authors persistently call the described procedure calibration, although they say that camera calibration is already performed prior to that.

The major difference of this approach from other approaches is that the authors suggest estimating the rotation *and* translation transformations between the camera and the IMU. This paper is a pioneer in this direction.

The next approach uses a similar technique of propagating the states and the covariance matrices linearly as within an EKF. The authors of [120] estimate the spatial alignment using a common system identification technique. The innovation in an EKF is minimized by standard gradient descent methods. Their experiment results are promising, but one drawback is that the EKF linearizes the motion model which yields significant errors, especially in the case of

large rotations. On the other hand, the results suggest that the registration of an optical system to an IMU with the proposed method can be performed from a small set of measurements with a high accuracy which could form a good basis for automatized registration.

Similarly to [193], the authors of [295] estimate the transformation parameters between the IMU and camera and additionally estimate the deterministic errors of the IMU using a sigma-point Kalman filter. They also use a checkerboard for estimating the feature points pose relative to the camera. For the initialization of the transnational transform, they used rudimentary measurement equipment, and for the rotational transform, they used the gravity measured by the accelerometer and the fact that the checkerboard is placed on a horizontal plane perpendicular to the gravity vector. Again, the authors of this paper confuse calibration and registration.

The authors used a simulation of the setup with a camera with 10 Hz and IMU with 100 Hz to prove the functionality of the method with a sigma-point Kalman filter. However, because of the simulation, they had a controlled environment and they had no need to synchronize the sensors.

Research [120] was continued in [121]. Here the registration is performed without any additional hardware except a checkerboard-pattern paper (see Figure 6.5). Authors provide registration in various setups (including, for example, different lens types) for a system consisting of a rigidly connected IMU and a camera.

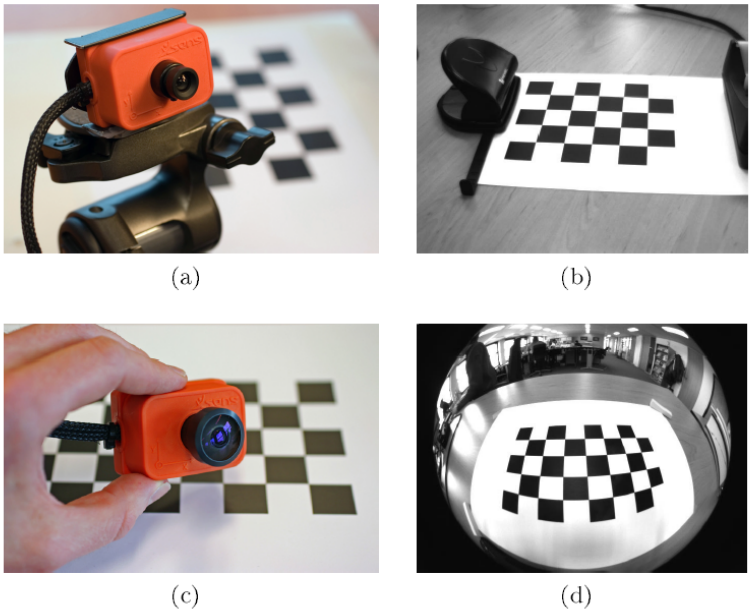


Figure 6.5.: Configurations of sensor input: (a) and (b) - 4 mm perspective lens; (c) and (d) - 190°fish-eye. Source: [121]

The approach presented in the paper turns the parameter estimation problem into a system identification problem. Therefore, the approach can be seen as a hybrid scheme: first, the

motion estimation task is performed, and then the registration parameters are iteratively estimated.

The authors of [81] showed how the IMU-camera registration problem can be modelled and resolved in a nonlinear optimization framework by modelling the sensors' trajectory through batch optimization. Their approach estimated a continuous trajectory encoded as a spline rather than representing the motion as a discrete sequence of states, contrary to the Kalman filter-based methods described in [193] and [120]. Hence, their registration results consider all available information at the same time and do not estimate the constant IMU-camera alignment on a sample-by-sample basis.

All of the studies described above have a common feature: all the approaches assume that the IMU and the camera have been calibrated in a separate, preceding step and that any offset in the timing of their measurements has been compensated for.

Paper [85] pursued a similar continuous-time approach as in [81], but additionally included the estimation of a temporal offset between the camera and the IMU into the estimator – a parameter that was previously estimated in a separate procedure in, e.g., [142, 181]. The proposed algorithm is similar to [81] because also employs B-splines to parameterize the motion of the device. The internal noise parameters of the used IMU are estimated separately using Allan Variance analysis.

Paper [152] extends this work by also introducing additional sensing errors in the IMU model, namely measurement scale, axis misalignment, cross-axis sensitivity, the effect of linear accelerations on gyroscope measurements, and the orientation between the gyroscope and the accelerometer.

A similar model that includes calibration parameters of the IMU is used in [201]. In this paper the authors propose a non-parametric batch formulation that helps avoid basis function selection for the pose and bias trajectories, thus simplifying the approach and reducing the preprocessing stage. The non-parametric and parametric formulation are compared in real-world experiments with the conclusion that the accuracy and precision of both methods are similar, with the proposed non-parametric one being easier in use.

The approach proposed in [231] is based on [85] and extends that method to incorporate multiple IMUs into a single estimator. The research studies registration of a sensor suite comprising one or multiple IMUs and one or multiple exteroceptive sensors. In this paper registration of multiple sensors is performed conjointly with calibration of IMUs. It should be noted that the same formulation can be employed to determine the displacement of individual accelerometer axes, arriving at a more complete model even in sensor suites comprising only a single IMU.

Another method for registration of rigidly attached camera and IMU is proposed in [231].

Figure 6.6 shows an ISO setup consisting of a camera with the coordinate frame $\mathcal{F}_{\mathcal{C}}$, an IMU with the coordinate frame $\mathcal{F}_{\mathcal{A}}$, and the checkerboard with its coordinate frame $\mathcal{F}_{\mathcal{W}}$.

The approach is based on a continuous batch-estimator, allowing for precise estimation of parameters. The proposed estimator includes parameters for multiple IMU calibration and

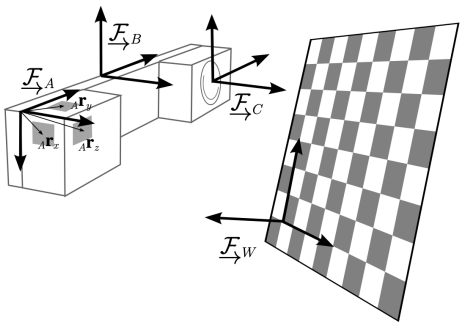


Figure 6.6.: The setup for registration of camera and IMU. Adapted from [231]

registration with respect to each other and one or several optical sensors. The approach allows determining locations of individual accelerometer axes which can be of high importance when working with IMUs that have multiple integration circuits for included accelerometers.

6.2.1.2. Marker-less Registration

In contrast to marker-based, marker-less (self-registration) methods rely on natural features to set up the sensor models without the need for external markers such as checkerboards.

Paper [135] presents an approach for online registration of an IMU sensor to a camera that does not require a marker for optical tracking. The method also allows estimating calibration parameters of the IMU. The authors include a lot of calibration parameters and even propose a special procedure for bias instability detection and estimation.

The setup consists of a monocular camera and an IMU with a computational unit (see Figure 6.7) and other sensors, including stereo cameras, GPS, and lidar are used for verification of the results obtained by the original setup.

Figure 6.7.: Experimental setup: monocular camera and IMU. Source: [135]

The results provided in the paper show good quality of registration and subsequent

tracking with the use of a custom filter. However, the authors note that in case of encountered optical sensor failures or feature recognition failures, the algorithm proved to be unreliable. The authors propose a method for complementary modalities that help compensate for the failures.

In [166] the following parameters are included into a (non-parametric) EKF-based estimator: time offset between camera and IMU, scale errors and axis misalignment of all inertial axis, linear acceleration effect on the gyroscope measurements (g-sensitivity), camera intrinsics including lens distortion and the rolling-shutter line-delay. A simulation study and real-world experiments indicate that all these quantities can indeed be estimated online solely based on natural features.

The work of Patron-Perez et al. [217] additionally calibrates the camera intrinsics and uses a continuous-time formulation with a B-splines parametrization.

6.2.2. OSI to IMU registration

In outside-in applications, external tracking devices are installed stationary to detect the target placed inside of the stationary setup.

An example of a Multi-IMU system employed within an OSI framework is presented in Figure 6.8: a plate with a Multi-IMU system (consisting of 8 IMUs) and a target attached to it are surrounded by static optical trackers.

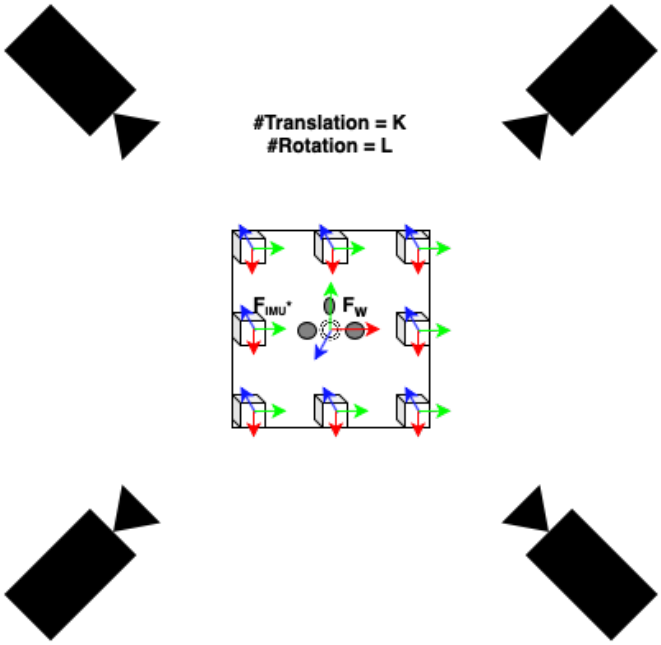


Figure 6.8.: The setup for registration of OSI and IMU

The registration is performed by comparing the measured accelerometer and gyroscope data with processed camera data.

For accelerometer registration, the position of the target, obtained from the cameras, is derived twice to obtain acceleration. Then, acceleration is measured by the accelerometers attached to the plate. Note, that it is very important to delete the gravity contribution from the measurements of accelerometers because the constant gravity field is not registered by the cameras. In principle, the procedure then follows that described in Subsection 6.1.1. It is recommended to move the plate in at least 5 directions. It is also recommended to move it with linearly changing velocity (i.e., with constant acceleration) to ensure easier comparison and computations.

The same procedure, described in Subsection 6.1.2, should be performed for gyroscopes. Namely, by performing at least 5 rotational tests, a comparison of the angular velocity measured by the gyroscope with the angular velocity computed as the first derivative of the angle obtained from the cameras should be performed.

6.3. Registration with Tip-calibration

As explained earlier, registration with tip-calibration is an optical tracker-based interactive method, where an engineer is pointing with a tracked device at three or more specified positions on each IMU [131, 130]. In this section, the method is applied to obtain the orientations of IMU sensors in a Multi-IMU.

Figure 6.9 shows the algorithm and the schematic of the tip-calibration.

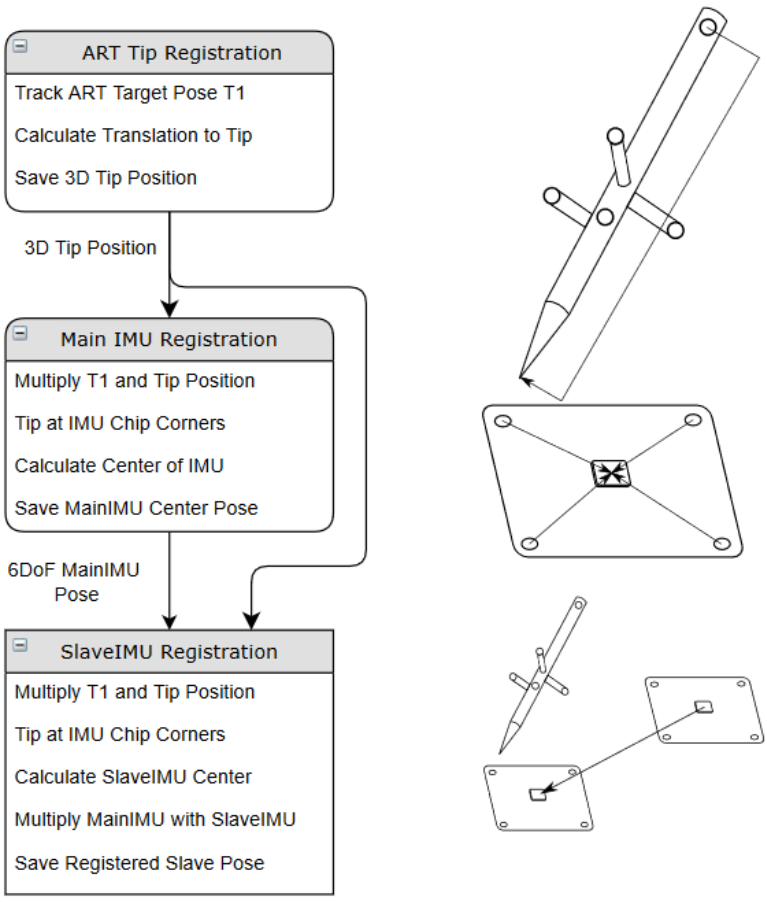


Figure 6.9.: Algorithm and schematic of tip-calibration method

The physical setups for the linear and planar cases are shown in Figures 6.10 and 6.11 respectively. In the linear setup the 3 IMUs are arranged in a line. In the planar setup the 3 IMUs are arranged in the same plane (but not in a line).

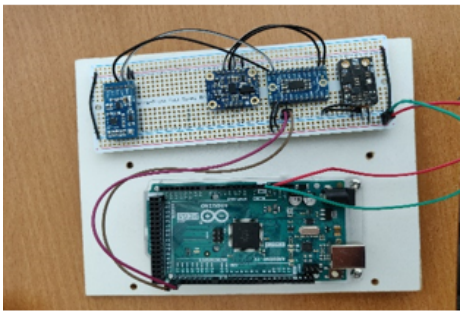


Figure 6.10.: Linear setup

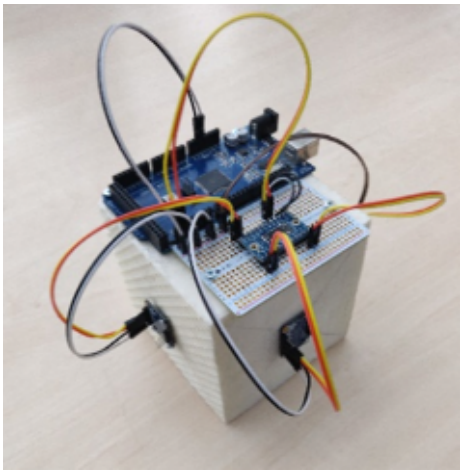


Figure 6.11.: Planar setup

Figure 6.12 shows the special relationship graph for registration of the IMUs and the outside-in optical tracker from Advanced Realtime Tracking company (ART).

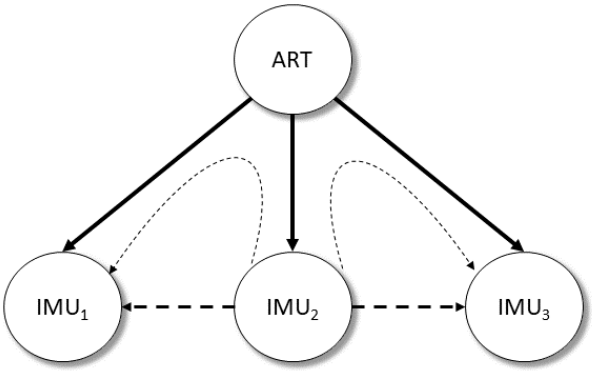


Figure 6.12.: Special relationship graph for IMUs to OSI registration

The results of Multi-IMU registration obtained with the tip-calibration algorithm are presented in Figures 6.13 and 6.14. The figures show the coordinate systems of the IMUs as computed with the algorithm.

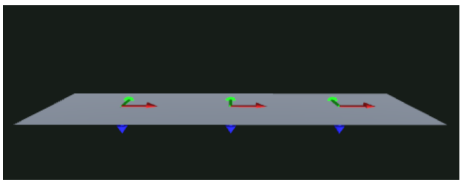


Figure 6.13.: Aligned orientations: linear case

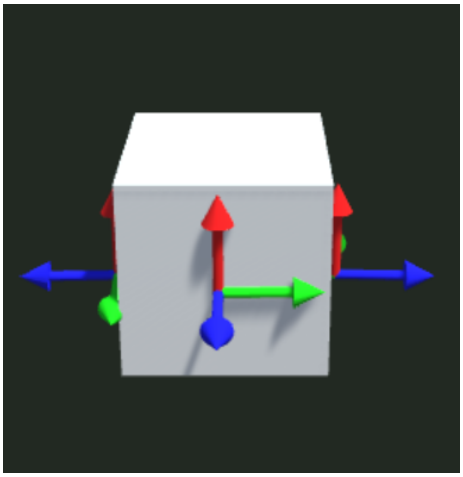


Figure 6.14.: Aligned orientations: planar case

6.4. Registration of Multi-IMU

This section is focused on a practical problem that shows the importance of registration in Multi-IMU systems.

The problem was presented as part of a project with a company that specializes in the production of head-mounted displays (HMD).

The CAD model of the HMD is presented in Figure 6.15. The setup consists of 3 IMUs, each measuring the data in its own coordinate system.

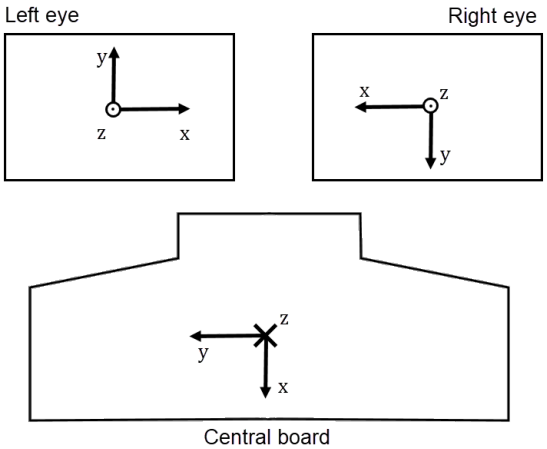


Figure 6.15.: CAD model of the setup

The physical setup is presented in Figure 6.16. As seen in the photos, the setup is also equipped with an optical target to allow precise OSI tracking. The data obtained from the

OSI tracker can be used for verification of individual and fused IMU data. For example, the orientation can be numerically differentiated to obtain the angular rate, which can then be compared with the measurements from the IMU gyroscopes, as described in Subsection 6.2.2.

Figure 6.16.: Photos of the setup

Originally, the project was only focused on data fusion. Because the choice of fusion algorithm presents a whole range of various options, it was believed that by using an appropriate algorithm registration can be omitted. In practice, however, it turned to be an infeasible solution: the unregistered fused data (blue line) are presented in Figure 6.17 together with the data obtained from an optical tracker (magenta line, "DTrack") and raw individual data from the central (green line), left (cyan line), and right (red line) IMUs. The curve corresponding to the fused data does definitely not coincide with the benchmark curve from the optical tracker.

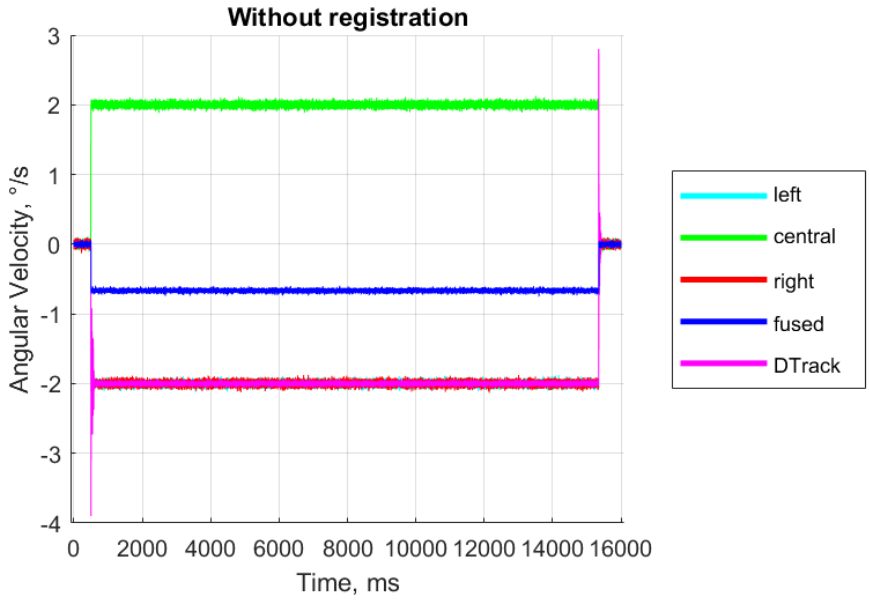


Figure 6.17.: Measured angular velocity: without registration

Figure 6.18 shows the same curves with the fused curve plotted after IMU registration. It presents a nice correspondence of the fused data with the data obtained with the OSI optical tracking system.

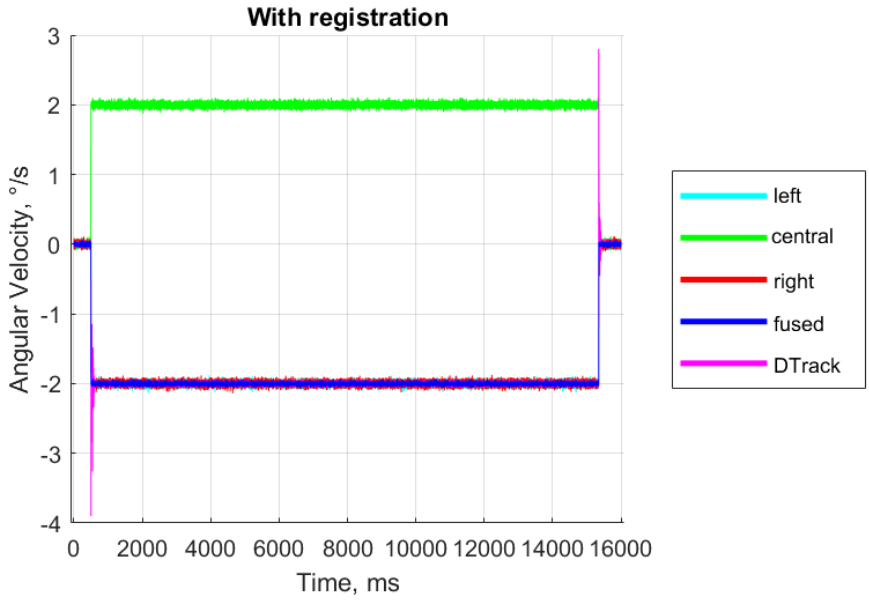


Figure 6.18.: Measured angular velocity: with registration

This experiment clearly shows that without registration the data cannot be fused correctly, and therefore registration is necessary in the case of Multi-IMU, especially in the case of physically unobservable setups.

"All we have to decide is what to do with the time that is given us."

— J. R. R. Tolkien

7. Synchronization

Another critical problem that needs to be resolved when using multiple sensors is the consistent treatment of measurement time. When several unsynchronized measurements are used together, dynamic errors are introduced in intermediate steps [24].

When parts of a common object are observed with multiple sensors, especially if the sensors measure with different data acquisition systems, it can be difficult to merge the sensor data. The sensors are very likely to have different sampling rates and clock times. Various synchronization methods provide a solution to this problem.

There are two major classes of synchronization methods: hardware synchronization and software synchronization. Hardware synchronization implies that the acquisition of the data by sensors happens at the same time. In software synchronization, the time frames of the data measured by sensors are adjusted to be similar. The next two sections focus on hardware and software synchronization respectively.

7.1. Hardware Synchronization

Hardware synchronization relies on prior accurate synchronization of the free-running clock used by a sensor with the clock used in the processing unit that allows to then fuse the obtained data without problems. When hardware synchronization is used, the acquisition of sensor data is triggered with a trigger signal by a central hardware clock connected to all sensors of the system. The trigger signal is often derived from an oscillator in one of the participating sensors.

To ensure that all the sensors are measuring at the same time, the sensors should contain a special time-trigger connection (port), which can be used to generate the trigger signal when the sensor captures the data.

Hardware synchronization should provide a digital timing signal from a hardware output line along with a software packet that timestamps the digital signal in clock reference. The usual flow of hardware synchronization is as follows: One of the sensors is put in the master mode, generating the critical timing signals, and the other sensors are put in the slave mode. Once the synchronization cables are assembled and connected to the sensors in a specific topology (daisy-chain or star), a master device needs to be set up that will trigger the slave devices.

This approach is common in current sensor-fusion scenarios. Also note that this approach almost completely prohibits dynamic construction or reconfiguration of sensor setups as well as the use of common off-the-shelf hardware, which mostly lacks hardware synchronization interfaces.

Multiplexing the IMU A multiplexer, or MUX, is a device that selects one of several analog or digital input signals and then forwards the data or signal along that line into a single input. This means that multiple sensors can share the same data-line simultaneously and the multiplexer chooses which device to listen to based on the inputs of the selector bits on the MUX. Multiplexers are mainly used to increase the amount of data that can be sent over a network within a certain amount of time.

In its simplest form, a multiplexer has two signal inputs, one control input, and one output. An example of this configuration is a home stereo unit that allows the user to switch between the audio of the CD player, the DVD player, or the cable television line. Multiplexers are also used in devices such as central processing units, as well as graphics controllers.

When multiple IMUs are used with one micro-controller, there is a problem that there are only two I2C (Inter-Integrated Circuit) address options for the IMU (namely, 0x68-LOW and 0x69–HIGH). A Multiplexer I2C can be used to connect to the address select pin of each of the sensor chips. The MUX assigns a new address to each IMU so that the data from heterogeneous or homogeneous IMU can be acquired without conflict and the processor of the micro-controller can then access this sensor alone without affecting the others.

For some scenarios, e.g., systems with GPS, the hardware synchronization is difficult to use, exploit or implement because sensors of this type have their own built-in clocks. Software synchronization is an alternative option for synchronization that can be used instead. The next section explains this type of synchronization.

7.2. Software Synchronization

When hardware synchronization is not available, the next best option is to use software-supported synchronization.

Software synchronization depends on correctly assigning timestamps to each sensor measurement. The timestamps can preferably be provided by the sensor itself or can be generated by the tracking framework as soon as the measurement enters the system. In either case, the timestamps of different sensors have to be adjusted to refer to a common time base to provide the basis for further fusion.

An example that shows the importance of synchronization is the following: Consider error evaluation of tracking setups where the data measured by the tracking system under test is compared to ground-truth data from a second, more precise system. Unless the data is only recorded when all sensors are completely motionless, a temporal offset between the system under test and the ground-truth causes an additional tracking error which necessarily disturbs the experiment.

Basically, there are two cardinally different approaches to software synchronization. First, an approach called Single Constraint At A Time (SCAAT) [285, 33] can be used. As follows from the name of the approach, instead of bringing the sensor data to the same time frame, each measurement is used sequentially when it is obtained. The details of the SCAAT approach are provided in the respective section in Chapter 8 because the usage of SCAAT provides the basis for several fusion techniques by basically eliminating synchronization. The methods with similar logic rely on the use of predictive filters [16] with an appropriate motion model for the measurement data. This process needs synchronized accurate clocks on all machines to get useful estimates from the filters. The success of this method highly depends on the choice of a good motion model and therefore is hard to apply in a general context.

The alternative class of synchronization methods is what is usually just referred to as synchronization. It consists of two main stages: First, the time delay of the sensors should be detected and measured or estimated. Then, based on the obtained delay value, an interpolation and/or extrapolation strategy should be devised and implemented that will later allow for data fusion. In the following subsections, common approaches for time delay estimation and interpolation/extrapolation stages are reviewed.

7.2.1. Time Delay Estimation

In order to correctly combine data from two sensors, it is necessary to know the exact temporal relationship between the data acquired from different sources.

For example, consider two one-dimensional signals that have been interpolated and can be assumed to be continuous on the same time domain.

The estimation of the relative lag of these two signals can be performed by a method called Time Delay Estimation (TDE). This method is widely applied in signal processing and is used for applications such as RADAR or SONAR [42]. Principally, the aim of this method is to estimate the temporal offset of a specific pattern contained in a generally noisy signal. In the original application sphere, the pattern is usually transmitted by the system and later received as a reflection.

The application of the time delay estimation method to tracking measurements differs from the standard application. In tracking, no template signal is being actively transmitted as one instance of the pattern searched for in the second channel; rather, the signal of one sensor is treated as the pattern that is being searched for in the other sensor signal. The offset between these two pattern instances is then assumed to be the relative lag between the sensors.

The general procedure of the time delay estimation keeps one signal fixed and shifts the second one in time (relative to the first one). For each possible time shift, a similarity measure is computed. Finally, the time shift that maximizes this measure is accepted as the time-delay estimate.

Cross-Correlation One of the earliest and still most used similarity measures for the mentioned setups is the normalized cross-correlation.

The normalized correlation [283] coefficient (also called Pearson correlation) is defined as the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{E[XY]}{\sigma_X \sigma_Y}. \tag{7.1}$$

Generally, when sample Pearson correlation can be computed, by substituting the theoretical covariance and deviations with their sample equivalents.

The computations of the measure can also be optimized by the method represented in [128] and others.

In application areas that have to deal with reverberation or multi-path effects, other methods which compensate for channel effects are employed, such as, e.g., the generalized cross-correlation [150]. Since in tracking data application no such "propagation channel" of the signal exists, such effects can be ignored.

The time-delay estimation as a solution for synchronization of multi-sensor tracking in Mixed Reality application was proposed in [123].

In [181] two methods specifically for IMU-camera time delay calibration were compared: cross-correlation and phase congruency. Both algorithms use the IMU and camera relative orientation information.

In [136] the authors showed how the random and unknown delays of measurements can be fused using the covariance union algorithm.

Beyond the techniques based on cross-correlation, there has been relatively little research on systematic time delay estimation for isolated black-box sensors. While substantial work has been done in the area of clock synchronization for sensor networks (e.g., [167],[192]), these algorithms depend on two-way communication between the nodes. The sensors can have one-way (sensor-to-host) devices, and thus, techniques that require two-way communication cannot be applied.

Other developed synchronization algorithms such as TICSync [110] also require two-way timing data and determine the timing between sensors using low-level data, such as direct measurements of network response times.

Another technique presented in [299] determines the systematic delay between two sensor data streams by incrementally restricting the interval in which a random event occurs (as measured by both sensors). This approach relies on the ability to detect and correctly identify such events, however, and requires a significant number of trials to converge.

Some approaches exist that allow estimating synchronization parameters while performing calibration, registration, filtering, or fusion.

In [85] the authors proposed a full maximum likelihood estimator to determine sensor extrinsics (i.e., perform registration) and a time delay between the camera and the IMU.

In [142] an algorithm called time delay iterative closest point (TD-ICP) has been presented, for calibrating the relative time delay between inertial and visual sensor data streams. The TD-ICP algorithm tries to minimize the distance between the orientation trajectory in 3D space by

estimating the angular and temporal alignment between the sensors (by aligning two curves in a 3D orientation space). The curves are generated from integrated IMU gyroscope data and from camera observations of landmarks in the environment. Each point on the respective curve has a corresponding timestamp, identifying the time at which the measurement arrived at the receiver. By registering the orientation curves, it is possible to use the timestamp values to estimate the relative delay between the sensor data streams. The proposed approach is related to the methods described in [238] and [270].

A passive approach for synchronization that considers random delays and clock drift is developed in [208]. The method operates by identifying low-latency sensor messages. As such, the technique does not identify the relative transmission and processing delays. However, the approaches in [208] and [142] are complementary and could be used together to estimate both systematic and random delays.

In [165] the authors proposed an online approach for estimating the time offset between the camera and IMU during EKF-based vision-aided inertial navigation. The time offset was explicitly included in the EKF state vector and estimated jointly with all other variables of interest. This enabled characterizing time-varying offsets and estimating online the variation in time offset.

7.2.2. Interpolation and Extrapolation

While working with multiple signals, it is a common assumption that each signal is represented as measurements that are sampled as equidistant points in time. In practice, however, even the sample rate of a single sensor may not be constant. Therefore, the relationship between the sample points of two signals produced independently by two sensors is generally not known. Each sensor has a variation in its update rate of measurements; this variation can aggravate the calculation of measurements for a similar time.

The idea that forms the basis of the solution to this problem is to create a common basis for sample points for both signals. Using the originally assigned timestamps for each measurement of any of the sensors, both signals are interpolated (usually using linear interpolation) between the individual sampling points, resulting in continuous signals.

To express the signals in a common time frame, correspondence of samples needs to be detected in both signals. The previous subsection was devoted to the time delay estimation method, which is used to calculate the time difference between the sensors' update rate. Note that in practice it suffices to perform the actual interpolation on only one signal.

The sampling points for similarity computations can conveniently be chosen to coincide with the original sampling points of one of the two sensors. If the sampling rates of the sensors differ, it is beneficial to interpolate the signal with a higher sampling rate to minimize interpolation errors.

Also, note that it is generally preferable to interpolate the one-dimensional projected signal as opposed to the higher dimensional tracking data. First, the interpolation at this stage is more straightforward and more clearly defined. This may be harder for general tracking

data (for example, there are different methods of quaternion interpolation or more generally interpolation of rotations that can lead to different data structures). Second, the computational complexity is also less due to the reduced number of dimensions.

7.3. Synchronization of Multi-IMU

This section is devoted to an experiment that focuses on the problem of synchronization in systems with Multi-IMU. The experiment described in this section was performed in the same setting as the experiment for registration described in Section 6.4.

As described earlier in this chapter, synchronization is necessary when dealing with a system that incorporates several sensors. When synchronization is not performed or is performed incorrectly, the system obtains measurements from several IMUs in different time frames, and then the data are processed as if the measurements were taken simultaneously. In this case, the end result can be inaccurate (and sometimes quite drastically so), even if the measurements are taken within short intervals of time.

Figure 7.1 shows the measurements of the angular velocity without synchronization from the system defined in the previous section. A close-up of the starting interval is shown in Figure 7.2. As can be seen from the plot, the fused angular velocity has a step-like shape that corresponds to the lack of synchronization.

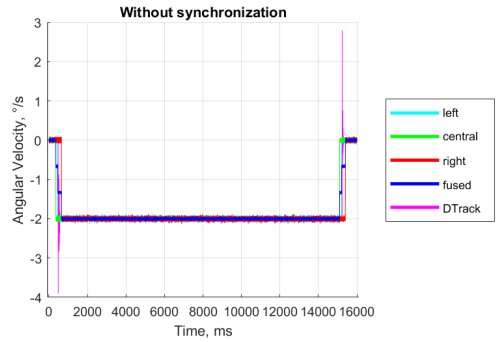


Figure 7.1.: Measured angular velocity: without synchronization

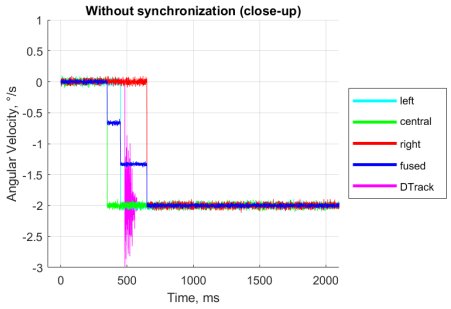


Figure 7.2.: Closeup: $t \in [0; 2000]ms$

In Figures 7.3 and 7.4, the data are synchronized first, and here the fused curve corresponds better to the benchmark curve as synchronization provides a basis for the correct fusion of the sensor data.

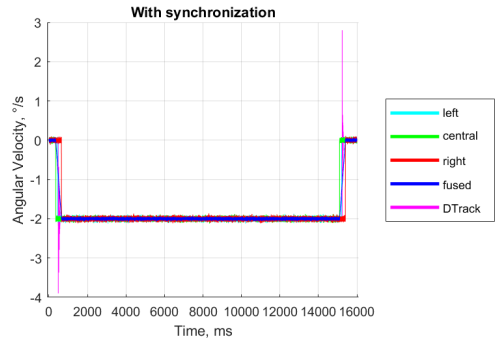


Figure 7.3.: Measured angular velocity: with synchronization

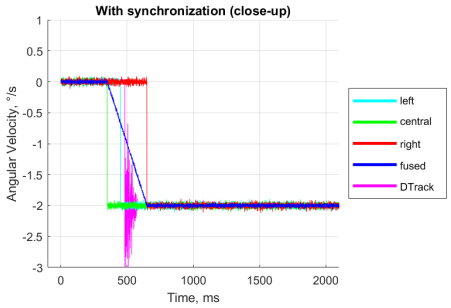


Figure 7.4.: Closeup: $t \in [0 : 2000] ms$

As can be seen from the results of this evaluation, synchronization is a necessary step before the sensor fusion.

"A single violin player is his own conductor; an orchestra requires a separate one."

— Karl Marx

8. Fusion

Sensor fusion is a process of combining sensory data (or related information) from several different sensors in order to enhance the estimation of the system state [86]. The resulting estimate is enhanced, i.e., is "better", than the estimates obtained from single individual sources. The enhancement can be in the sense of accuracy, reliability, availability, etc. Figure 8.1 shows a schematic of the sensor fusion framework.

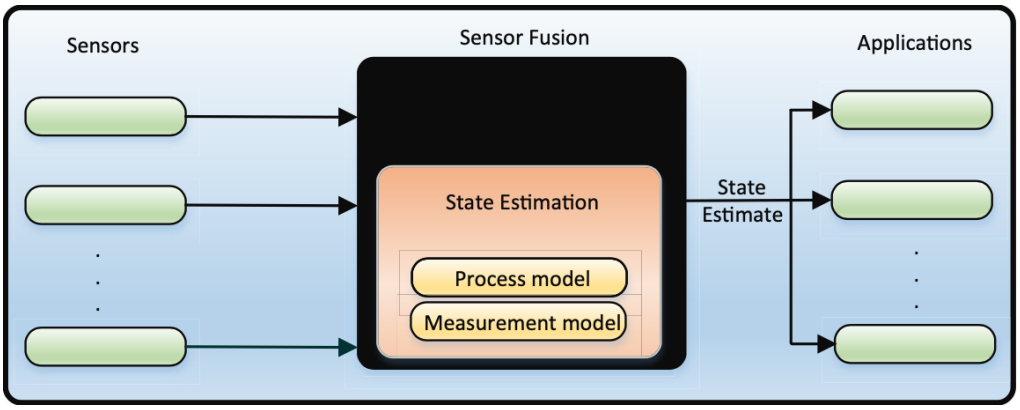


Figure 8.1.: Sensor fusion framework. Adapted from [175]

However, it should be noted that simply fusing the data is not enough. Data preprocessing and enhancement (including calibration, filtering, sensor registration, and synchronization) should not be omitted. Fowler criticized in [82]:

> *Massaging a lot of crummy data doesn't produce good data; it just requires a lot of extra equipment and may even reduce the quality of the output by introducing time delays and/or unwarranted confidence. [. . .] It takes more than correlation and fusion to turn sows' ears into silk purses.*

In general, systems with multiple sensors that used their fused data have several advantages over mono-sensor systems. The latter often exhibit the following problems [66]:

- Sensor deprivation:

  A breakdown of a sensor element causes a loss of perception of the desired object.

- Limited spatial coverage:

  An individual sensor usually covers a restricted region and extrapolation of its measurement rarely provides reasonable estimation.

- Limited temporal coverage:

  Some sensors need a particular set-up time to perform and to transmit a measurement, thus limiting the maximum frequency of measurements.

- Imprecision:

  Measurements from individual sensors are limited to the precision of the employed sensing element.

- Uncertainty:

  Uncertainty, in contrast to imprecision, depends on the object being observed rather than the observing device. Uncertainty arises when features are missing (e.g., occlusions), when the sensor cannot measure all relevant attributes of the precept or when the observation is ambiguous. A single-sensor system is unable to reduce uncertainty in its perception because of its limited view of the object.

The following advantages derive from the fusion of sensor data from a set of heterogeneous or homogeneous sensors [66]:

- Robustness and reliability:

  Multiple sensor suites have an inherent redundancy, which enables the system to provide information even in case of partial failure.

- Extended spatial and temporal coverage:

  One sensor can look where others cannot; therefore, it can measure while others cannot.

- Increased confidence:

  A measurement of one sensor is confirmed by the measurements of other sensors covering the same domain.

- Reduced ambiguity and uncertainty:

  Joint information reduces the set of ambiguous interpretations of the measured value.

- Robustness against interference:

  By increasing the dimensionality of the measurement space (e.g., measuring the desired quantity with optical sensors and ultrasonic sensors), the system becomes less vulnerable to interference.

- Improved resolution:

  When multiple independent measurements of the same property are fused, the resolution of the resulting value is better than a single sensor's measurement.

Fusion processes are often classified in a three-level model distinguishing low-, intermediate-, and high-level fusion:

1. Low-level fusion or raw data fusion combines several sources of raw data to produce new data that is expected to be more informative than the inputs.
2. Intermediate-level fusion or feature level fusion combines various features such as edges, corners, lines, textures, or positions into a feature map that may then be used for segmentation and detection.
3. High-level fusion also called decision fusion combines decisions from several experts. Methods of decision fusion include voting, fuzzy logic, and statistical methods.

Alternatively, fusion can be classified by the relationships between the fused sensors. Figure 8.2 explains the three classes of fusion in this classification: competitive, complementary, and cooperative fusion.

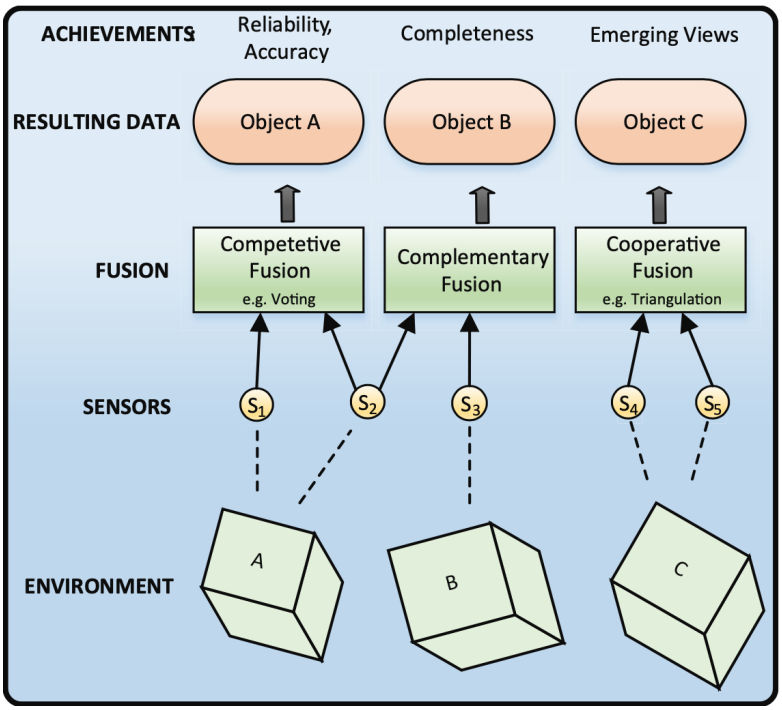


Figure 8.2.: Competitive, complementary and cooperative fusion. Adapted from [66]

It should be noted, however, that nowadays machine learning techniques exist that allow for data fusion with artificial neural networks. Sensor fusion with machine learning techniques is discussed in Section 8.2. Therefore the other methods are gathered in Section 8.1 under the title of “traditional” fusion.

8.1. Traditional Fusion

A common classification is adopted in this work for the traditional fusion methods: the sensor fusion methods are classified according to their architecture into centralized and decentralized methods.

Centralized fusion methods accumulate all the measured data from the sensors in a fusion center, where the data are stacked and then processed as a single measurement of larger dimensionality. Centralized fusion methods can provide globally optimal state estimations by directly combining local measurement models. Centralized methods have minimal information loss. On the other hand, they require large computation and data memories and present poor robustness and reliability if one of the sensors fails.

Decentralized fusion methods present architecture with a master filter and one or more local filters. First, the local filters process the data from the assigned sensors in parallel to provide local estimates and then the master filter fuses the obtained local estimates. Decentralized methods can also provide globally optimal (suboptimal) state estimates. At the same time, they cause small computation and communication loads in the fusion center and provide easy fault detection and isolation in case of sensor failure.

8.1.1. Centralized Architecture

As mentioned above, centralized fusion methods transfer individual sensor data to a fusion center, where the data are combined into a high dimensional vector that can be processed as a regular observation vector by standard algorithms. A schematic representation of a centralized filter employed for data fusion and state estimation is shown in Figure 8.3.

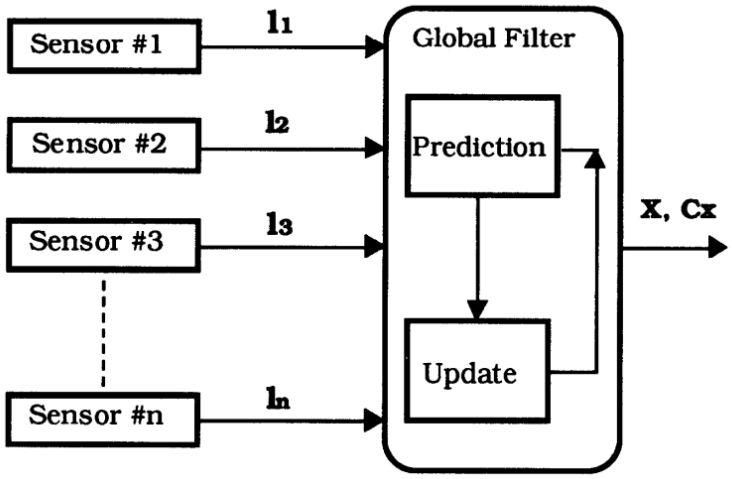


Figure 8.3.: Centralized filter architecture. Adapted from [88]

In principle, as described earlier, centralized architecture has no clear advantages over decentralized one. However, centralized Kalman filter-based fusion algorithms are still often

used in practice because of the ease of implementation and intuitive principles. Standard Kalman filter equations are applied to a combined global state vector with a corresponding global description of the process noise and update measurements provided by all the available sensors.

There are two other important fusion approaches that can be classified as centralized. The first one is SCAAT – Single Constraint At A Time – a method that allows for the fusion of data without direct synchronization. The other one is the Madgwick filter which is an IMU-specific method for fusing data obtained from accelerometer and gyroscope to provide orientation estimate. These two approaches are discussed further in this section.

8.1.1.1. SCAAT

Single Constraint At A Time (SCAAT) is an approach for asynchronous complementary sensor fusion proposed by G. Welch and E. Foxlin in [285]. The underlying idea is to store measurement models and covariance for each of the sensors and whenever new measurement from a certain sensor is available to use this sensor's model in a prediction-correction cycle step.

A common use of the SCAAT approach is the pose estimation with Kalman filter from various unsynchronized measurements each of which under-constrains the solution. The update rate is increased by computing the new estimate whenever a new measurement is available. Thus, the accuracy is increased by incorporating every measured data via a Bayesian filter.

The SCAAT Kalman filter for tracking can be constructed as follows. The model includes 3 positions and 3 orientations. The internal state vector is composed as follows:

$$\vec{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \Delta\phi\ \Delta\theta\ \Delta\psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}]^{\mathsf{T}}, \tag{8.1}$$

where x, y, z are positions in the respective axes, $\dot{x}, \dot{y}, \dot{z}$ are velocities, ϕ, θ, ψ are orientations.

Discrete state transitions are described by the equation

$$\vec{x} = A(\delta t)\vec{x}(t - \delta t) + \vec{w}(t), \tag{8.2}$$

where $\vec{w}(t)$ is the process noise vector and δt is the sample time.

Process noise covariance matrix Q reflects mutual correlations:

$$E[\vec{w}(t) \cdot \vec{w}^{\mathsf{T}}(t + \delta t)] = \begin{cases} Q, & \text{if } \delta t = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{8.3}$$

The observation equation can be written as

$$\hat{z}(t) = \vec{h}\left(\vec{x}(t),\ \vec{\alpha}(t), \vec{b}(t), \vec{(}c)(t)\right) + \vec{v}(t), \tag{8.4}$$

where $\hat{z}$ is the estimate of the measurement vector, $\vec{h}(\cdot)$ is the measurement function, $\vec{\alpha}$ is the external orientation vector, $\vec{b}, \vec{c}$ are the device parameters describing the source and the sensor

respectively. Vector $\vec{v}(t)$ is the uncorrelated over time measurement noise with covariance matrix $R(t)$.

A measurement Jacobian of the measurement function $\vec{h}$ is defined as follows

$$H(\vec{x}(t),\vec{\alpha}(t),\vec{b}(t),\vec{c}(t))[i,j] = \frac{\partial h(\vec{x}(t),\vec{\alpha}(t),\vec{b}(t),\vec{c}(t))[i]}{\partial \vec{x}[j]}. \tag{8.5}$$

So, in the introduced notation the SCAAT Kalman filter algorithm can be outlined as follows:

1. Time update (prediction):

$$\begin{aligned} \hat{x}^{-} &= A(\delta t)\hat{x}(t-\delta t) \\ P^{-} &= A(\delta t)P(t-\delta t)A(\delta t)^{\intercal} + Q(\delta t) \end{aligned} \tag{8.6}$$

2. Measurement prediction:

$$\hat{z} = \vec{h}(\hat{x}^{-},\hat{\alpha}(t),\vec{b}(t),\vec{c}(t)) \tag{8.7}$$

3. Jacobian computation:

$$H = H(\hat{x}^{-},\hat{\alpha}(t),\vec{b}(t),\vec{c}(t)) \tag{8.8}$$

4. Kalman gain computation:

$$K = P^{-}H^{\intercal}(HP^{-}H^{\intercal} + R(t))^{-1} \tag{8.9}$$

5. Measurement residual:

$$\Delta\vec{z} = \vec{z} - \hat{z} \tag{8.10}$$

6. Correction:

$$\begin{aligned} \hat{x}(t) &= \hat{x}^{-} + K\Delta\vec{z} \\ P(t) &= (I - KH)P^{-} \end{aligned} \tag{8.11}$$

7. External orientation vector update:

$$\begin{aligned} \Delta\hat{\alpha} &= quaternion(\hat{x}[\Delta\phi],\hat{x}[\Delta\theta],\hat{x}[\Delta\psi]) \\ \hat{\alpha} &= \hat{\alpha} \otimes \Delta\hat{\alpha} \end{aligned} \tag{8.12}$$

8. Orientation correction:

$$\hat{x}[\Delta\phi] = \hat{x}[\Delta\theta] = \hat{x}[\Delta\psi] = 0. \tag{8.13}$$

8.1.1.2. Madgwick filter

The filter currently known as the Madgwick filter was originally proposed by R. Mahony in [180]. S. Madgwick proposed an efficient numerical scheme [178, 179] for solving the equations derived in the original work, and the resulting filter was named after him. However, sometimes the filter is referred to as the Madgwick-Mahony filter, and sometimes, especially if an alternative numerical scheme is proposed for resolving equations, as the Mahony filter [74].

Before describing the filter itself, a special notation should be introduced. Let I, W, B denote the inertial, world, and body frames respectively. Let pre-subscript denote the source coordinate frame and pre-superscript denote the destination coordinate frame. If only a pre-superscript is present, it should be understood that the quantity was measured and is represented in the coordinate frame presented by the pre-superscript.

The Madgwick filter formulates the orientation estimation problem in quaternion space. Let q denote the orientation of the body in quaternion form (see Appendix B.3). The desired output of an orientation estimation algorithm is the orientation in the world body frame, i.e. ${}^{W}_{I}q$. The general idea is to estimate the orientation by fusing the integrated gyroscope measurements and accelerometer measurements. Gyroscope estimates are used as accurate estimates on a short scale and for faster movements and accelerometer estimates are used to compensate for long-term gyroscope drifts.

The filtering problem is defined as a minimization problem

$$\min_{{}^{W}_{I}\hat{q}\in\mathbb{R}^{4\times 1}} f\left({}^{I}_{W}\hat{\mathbf{q}},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}'\right), \tag{8.14}$$

where the target function is

$$f\left({}^{I}_{W}\hat{\mathbf{q}},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}'\right) = {}^{I}_{W}\hat{\mathbf{q}}^{*} \otimes {}^{W}\hat{\mathbf{g}} \otimes {}^{I}_{W}\hat{\mathbf{q}} - {}^{I}\hat{\mathbf{a}}'. \tag{8.15}$$

Here $\hat{\mathbf{q}}^{*}$ denotes the conjugate of $\hat{\mathbf{q}}$ and $\otimes$ is quaternion multiplication. Quaternion ${}^{W}\hat{\mathbf{g}}$ denotes the normalized gravity vector and is given by ${}^{W}\hat{\mathbf{g}} = [0\ 0\ 0\ 1]^{\mathsf{T}}$ and ${}^{I}\hat{\mathbf{a}}$ denotes the normalized accelerometer measurements. Notation ${}^{I}\hat{\mathbf{x}}$ denotes the normalized version of ${}^{I}\mathbf{x}$.

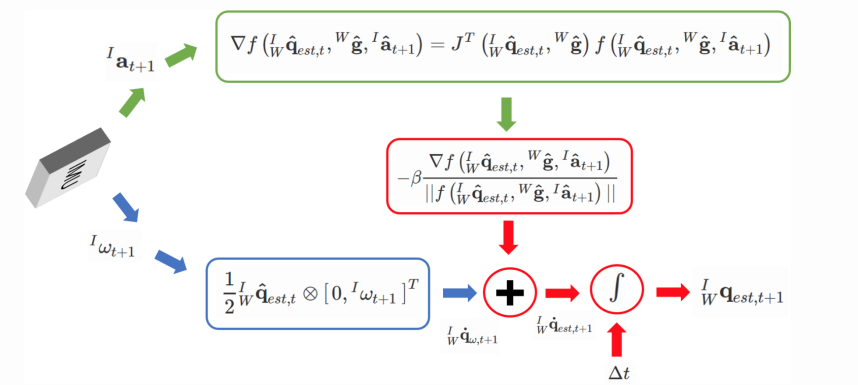


Figure 8.4.: Schematic of Madgwick filter. Source:[242]

The schematic for orientation estimation using the Madgwick filter is presented in Figure 8.4. The algorithm steps are as follows (to be repeated for every time stamp):

1 Obtain sensor measurements

Obtain gyroscope ${}^{I}\omega_t$ and acceleration ${}^{I}a_t$ measurements from the IMU. Let ${}^{I}\hat{a}_t$ denote the normalized acceleration measurements.

2a Orientation increment from accelerometer

Note: this step is represented with green in Figure 8.4.

Compute orientation increment from acceleration measurements (with gradient step).

$$\nabla f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}_{t+1}\right) = \mathbf{J}^{T}\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}}\right) f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}_{t+1}\right) \tag{8.16}$$

$$f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}_{t+1}\right) = \begin{bmatrix} 2\left(q_2q_4 - q_1q_3\right) - a_x \\ 2\left(q_1q_2 + q_3q_4\right) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix} \tag{8.17}$$

$$\mathbf{J}\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}}\right) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{8.18}$$

Update term (orientation component from accelerometer measurements) is given by

$${}^{I}_{W}\mathbf{q}_{\nabla,t+1} = -\beta\frac{\nabla f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}_{t+1}\right)}{\| f\left({}^{I}_{W}\hat{\mathbf{q}}_{est,t},\ {}^{W}\hat{\mathbf{g}},\ {}^{I}\hat{\mathbf{a}}_{t+1}\right)\|}. \tag{8.19}$$

2b Orientation increment from gyroscope

Note: this step is represented with blue in Figure 8.4.

Compute orientation increment from gyroscope measurements (with numerical integration):

$${}^{I}_{W}\dot{\mathbf{q}}_{\omega,t+1} = \frac{1}{2}\,{}^{I}_{W}\hat{\mathbf{q}}_{\omega,t+1} \bigotimes \left[0,\ {}^{I}\omega_{t+1}\right]^{T}. \tag{8.20}$$

3 Fuse measurements

Note: this step is represented with red in Figure 8.4.

Fuse the measurements from both the accelerometer and gyroscope to obtain the estimated attitude ${}^{I}_{W}\hat{\mathbf{q}}_{est,t}$:

$$\begin{aligned} {}^{I}_{W}\dot{\mathbf{q}}_{est,t+1} &= {}^{I}_{W}\dot{\mathbf{q}}_{\omega,t+1} + \frac{1}{2}\,{}^{I}_{W}\mathbf{q}_{\nabla,t+1}. \\ {}^{I}_{W}\mathbf{q}_{est,t+1} &= {}^{I}_{W}\hat{\mathbf{q}}_{est,t+1} + \frac{1}{2}\,{}^{I}_{W}\dot{\mathbf{q}}_{est,t+1}\Delta t. \end{aligned} \tag{8.21}$$

Here, Δt is the time elapsed between two samples at t and $t+1$.

In the Madgwick filter, the only tunable parameter is the trade-off parameter β which determines when the gyroscope has to take over the accelerometer. Also, the user needs to specify the initial estimates of the orientation, biases, and sampling time.

The initial orientation can be assumed to be zero if the device is at rest or alternatively it has to be obtained by external sources such as a motion capture system or a camera. The bias is computed by taking an average of samples with the IMU at rest and computing the mean value. Note that the bias changes over time and the filter will start to drift because of that.

8.1.2. Decentralized Architecture

As stated earlier, decentralized architecture implies a two-stage data processing technique, that employs a master filter and one or more local filters. First, the local filters process the data of the assigned sensors to produce the best possible local estimates. After that, the master filter fuses the local estimates, providing the best global estimate. Figure 8.5 shows a schematic of a decentralized filter.

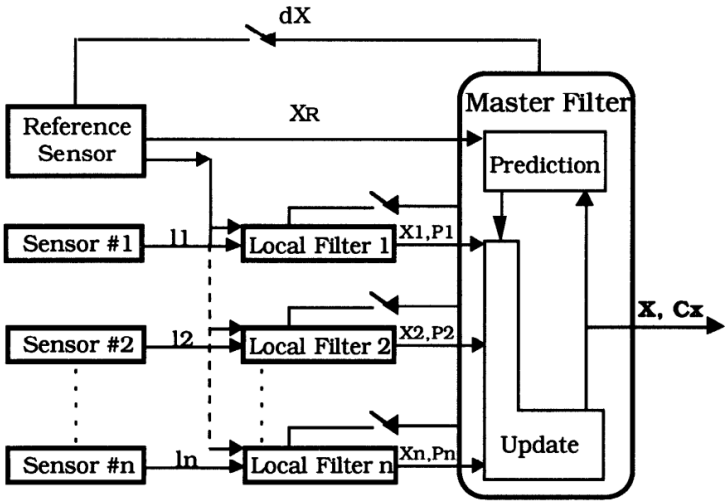


Figure 8.5.: Decentralized filter architecture. Adapted from [88]

8.1.2.1. Fusion By Average

The simplest master filter that can be used is averaging, where the data coming from local filters are fused by a simple average. The fused measurement is calculated as

$$x_m = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{8.22}$$

8.1.2.2. Kalman Filters for Fusion

Kalman filters are one of the most commonly used filters both for noise elimination and for sensor data fusion. The are various decentralized Kalman filter techniques, here some of them are shortly reviewed. Federated Kalman filter is then considered in more detail.

Decentralized Kalman filter fusion methods:

- Federated Kalman Filter (FKF)
  - \+ Eliminates correlation between local estimations by using covariance upper bound.
  - \- Only suitable for *linear* multi-sensor stochastic systems.
  - \- Poor robustness and reliability in case of single sensor failure.
  - \- Suboptimal local and global state estimations: the filter uses covariance upper bound instead of the covariance itself.
  - \- Requires local estimations from local filters to be independent.
- Unscented Kalman Filter (UKF)
  - \+ Applicable to nonlinear stochastic systems.
  - \- Requires an accurate system model.
- Unscented Kalman Filter-based Federated Kalman Filter (UKF-FKF)
  - \+ Applicable to multi-sensor nonlinear stochastic systems.
  - \- Suboptimal estimates due to the use of covariance upper bound.
  - \- Requires an accurate system model.
- Adaptive Unscented Kalman Filter (AUKF)
  - \+ Improved adaptability and robustness of UKF against process-modelling errors.
  - \- Large computation burden.
- Adaptive Fading Unscented Kalman Filter (AFUKF)
  - \+ Improved adaptability and robustness of UKF against process-modelling errors.
  - \+ Modest computation load.

Federated Kalman filter A special case of decentralized filtering is the federated filtering technique introduced by N. Carlson in [40]. Federated Kalman filter is a decentralized filtering algorithm with a two-level structure. The difference between the federated Kalman filter and other decentralized filters is that the federated Kalman filter contains an information-sharing process [131]. During this process, the total system information is divided among the local filters based on an information-sharing principle. The basic concepts of information sharing also include that it can perform local time propagation and measurement update processing (adding local sensor information) and it can recombine the updated local information into a

new total sum [122, 19, 301]. The list of possible information to be shared is understood to be the kinematic process noise, the initial conditions information, and common measurement information. Usually, it is the kinematic process noise that is to be shared.

The local filters and the master filter, comprising the federated Kalman filter, are discussed next.

Local filters For each local filter $i = 1, \dots n$, unscented Kalman filter (UKF) is used for the triaxial accelerometer bundles and for the triaxial gyroscope bundles.

The motion model of the accelerometers of IMU_i consists of

- the estimated state vector $\hat{X}_i$ defined by position p_i, velocity v_i and acceleration a_i: $\hat{X}_i = [p_i \; v_i \; a_i]^\intercal$,
- the observation vector z_i defined by measured acceleration a_i: $z_i = [a_i]$,
- the transition matrix F_i for the prediction step assuming constant acceleration:

$$F_i = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{8.23}$$

- the observation model H_i: $H_i = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$,
- the process noise Q_i: a 3x3 white noise matrix (velocity random walk),
- the measurement noise R_i: a 1x1 matrix (rate random walk),
- the state variance matrix P_i.

Similarly, the motion model of the gyroscopes IMU_i consists of

- the estimated state vector $\hat{X}_i$ defined by angle α_i and angular velocity ω_i: $\hat{X}_i = [\alpha_i \; \omega_i]^\intercal$,
- the observation vector z_i defined by measured angular velocity ω_i: $z_i = [\omega_i]$,
- the transition matrix F_i for prediction step, assuming constant angular velocity:

$$F_i = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \tag{8.24}$$

- the observation model H_i: $H_i = \begin{bmatrix} 0 & 1 \end{bmatrix}$,
- the process noise Q_i: a 2x2 white noise matrix (angle random walk),
- the measurement noise R_i: a 1x1 matrix (rate random walk),
- the state variance matrix P_i.

Each local filter receives as input the observation vector z_i from the local sensor i, consisting of acceleration data a_i and angular velocity ω_i.

As output, they each provide their updated state vector $\hat{X}_i$ as well as their state variance matrix P_i.

Master filter The master filter is also an unscented Kalman filter. It receives the outputs $\hat{x}_i$, P_i from all local filters as input. As outputs, it produces the updated state vector of the overall system state $\hat{X}_f$, as well as an overall state variance estimate P_f and a scalar weight β_i. These are given as additional input back to all local filters, influencing their states and their noise estimates. Furthermore, they constitute the overall system output, made available to applications using the IMUs as a bundled tracking device.

Internally, the master filter takes the system states and the state variance matrices of the local filters and performs the following steps:

1. **Time Update (Prediction):** compute a time update of $\hat{X}_f$, P_f:

$$\begin{aligned} X_m &= F \cdot X_f \\ P_m &= F \cdot P_f \cdot F^T + Q. \end{aligned} \tag{8.25}$$

2. **Fusion:** compute P_f, $\hat{X}_f$ and β_i:

$$\begin{aligned} P_f^{-1} &= P_m^{-1} + \sum_{i=1}^{n} {P_i}^{-1} \\ \hat{X}_f &= P_m^{-1} \cdot X_m + \sum_{i=1}^{n} {P_i}^{-1} \cdot \hat{X}_i \\ \beta_i &= \frac{tr(\Lambda_i)}{tr(\Lambda_m) + \sum_{i=1}^{n} tr(\Lambda_i)}, \end{aligned} \tag{8.26}$$

   where Λ_i is the eigenvalue of $P_i.{P_i}^T$.

3. **Updating local values:** update $\hat{X}_i$, P_i according to $\widehat{X}_f$, P_f:

$$\begin{aligned} \hat{X}_i &= \hat{X}_f, \\ P_i &= {\beta_i}^{-1} \cdot P_f. \end{aligned} \tag{8.27}$$

8.2. Machine Learning-based Fusion

Nowadays many applications in robotics, navigation, and XR rely on the use of different kinds of sensor data. Generally, sensors provide incomplete, inconsistent, or inaccurate data. Sensor fusion has proved successful where it has been employed to improve the accuracy of sensor measures. Work [73] investigates the accuracy and reliability of different AI techniques

such as SVM (Support Vector Machine), Cascade, MLP (Multilayer Perception), Cubist, RBF (Radial Basis Function), and M5 (Model Tree) for sensor fusion. It shows the SVM algorithm has the best performance compared to the ground truth. This also shows the potential of Machine Learning for sensor fusion.

Paper [102] describes an approach of using the artificial neural networks to do the sensor fusion to detect the eye movement using an infrared sensor array. From their experiment, they showed that the artificial neural networks perform well under the situation of noise and sensor fault.

In [6] the authors attempted to measure different physiological and behavioral indicators of arousal, cognitive effort, and stress, based on the AdaBoost algorithm with sensor fusion. Their experiments indicated that fusing data from vocal sensors and eye trackers can significantly improve accuracy.

In [5] the authors fused a Time-of-Flight and a stereo camera to get depth data and then feed the data into their proposed CNN-based deep learning model. Their experimental results showed that the proposed approach increases the accuracy of the depth estimation.

In [196] authors showed that sensor-specific normalization increases the prediction accuracy of the deep learning methods of sensor fusion in human activity recognition (HAR) data.

8.3. Comparison of Traditional IMU Fusion Algorithms

In this section, some of the commonly used fusion algorithms are compared when applied to a practical problem.

In this particular experiment, two Kalman-based decentralized fusion algorithms are compared in the same setting. A decentralized fusion architecture calls for local filters for each of the used IMUs and a central master filter. The standard linear Kalman filter (KF) and the unscented Kalman filter (UKF) are used as local filters. For the master filer, a simple averaging filter (AVE) and a modified federated Kalman filter (MFKF) are used.

The considered setup consists of 3 IMUs of the same type (i.e., the expected quality of the obtained measurements can be reasonably assumed to be similar).

When comparing the performance of the fusion algorithms, two characteristics are of major interest: computational time and accuracy. The measured characteristics are presented in Table 8.1.

Algorithms	Time, ms	Error, °/s $\times 10^{-3}$
KF_AVE	4.025	3.50
UKF_AVE	5.664	3.50
KF_MFKF	5.030	3.08
UKF_MFKF	6.440	2.95

Table 8.1.: Fusion algorithms: Time complexity vs Accuracy

The data presented in Table 8.1 are visualized in Figure 8.6. The plot clearly shows that while UKF_MFKF allows for the most accurate results, it also has the longest computational time. Two algorithms, KF_AVE and UKF_AVE, have the same accuracy, but KF_AVE requires a much shorter computational time.

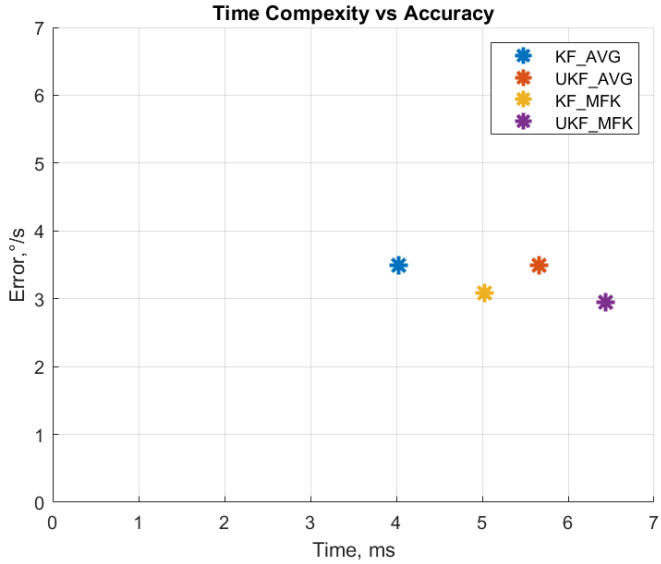


Figure 8.6.: Fusion algorithms: Time complexity vs Accuracy

Next, consider the setup where one of the IMUs provides results of worse quality than the other two. The data for this experiment were obtained from the measurements from the original one by corrupting the data provided by one of the IMUs. This choice of data acquisition ensures that the other two IMUs present the same data as before (including, for example, the noise terms that can vary from run to run).

The results of this experiment are summarized in Table 8.2.

Algorithms	Time, ms	Error, °/s $\times 10^{-3}$
KF_AVE	4.027	5.71
UKF_AVE	5.663	5.71
KF_MFKF	5.028	3.11
UKF_MFKF	6.441	3.01

Table 8.2.: Fusion algorithms: Time complexity vs Accuracy

Note that the computational time changed insignificantly because the algorithms take the same steps as before on the datasets of the same dimensions. The error column, on the other hand, shows substantial changes. In particular, the error obtained via the KF_AVE and UKF_AVE, i.e., the two filters that employ averaging for the master filter, has increased by

more than 50%. The other two filters use the modified Kalman filter as the master filter. These filters treat the data from the corrupted IMU deservingly as corrupted data (based on the covariance matrix computed from the data), bringing a smaller contribution from it to the fused result.

Figure 8.7 shows the data from Table 8.2.

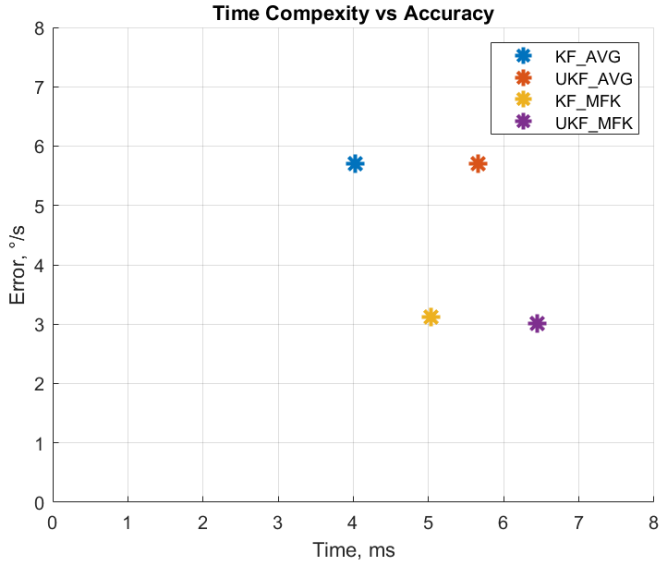


Figure 8.7.: Fusion algorithms: Time complexity vs Accuracy

In general, accuracy and computational time should be prioritized separately in every particular application: some applications call for high accuracy with no respect to time, others, conversely, require only rough estimation of parameters but the time is of the essence. By running massive offline computations on representative measurement sets, fusion algorithms can be chosen according to the particular needs, and then be used in an online regime on real data.

"We're not lost. We're locationally challenged."

— John M. Ford

9. Pose Estimation

This chapter explains how to estimate the pose by using only a Multi-IMU system.

Calculating pose data from multiple IMUs requires a number of carefully developed and analyzed processing steps. Especially for low-cost, commodity IMUs, it is essential to design a processing pipeline and to evaluate and document how each step increases the tracking quality while being easy to perform by service engineers when they install such Multi-IMU setups in gadgets.

9.1. Pose Estimation with Traditional Approach

This section is focused on the so-called traditional approach for pose estimation that requires precise and thorough data enhancement. Figure 9.1 shows an overview of the proposed pipeline for pose estimation.

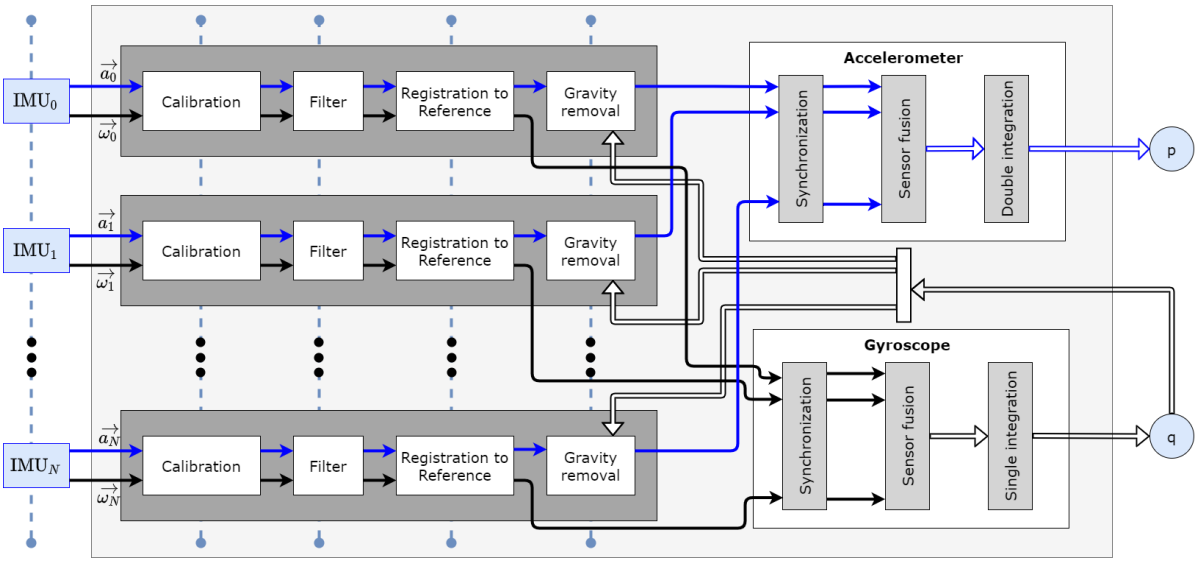


Figure 9.1.: Multi-IMU pipeline for pose estimation

Conceptually, the initial steps of the pipeline (data acquisition, calibration, filtering, registration, and gravity removal) are applied separately to each IMU. Yet, it is possible to handle these steps jointly for all IMUs, thus reducing the required time and potentially improving

the quality. The dashed vertical lines in Figure 9.1 represent that possibility. The strong outline of the rectangle surrounding synchronization, fusion, and time integration steps of the accelerometer (and respectively, gyroscope) indicates that these steps are combined into a joint spatial-temporal approach.

The following subsections present and discuss solutions for each step of the pipeline.

9.1.1. Gyroscope: From Angular Rate to Orientation

9.1.1.1. Preprocessing

As shown in Figure 9.1, there are several steps that should be performed to get an improved data measured by the gyroscope. These steps should be undertaken prior to the sensor fusion.

The first step of preprocessing is calibration, which allows removing the deterministic errors from the raw data. The calibration procedure can be done offline by collecting some sampling data. The output of the calibration is the intrinsic parameters (bias vector and scale and misalignment matrices) of the gyroscope, which can be used to eliminate the deterministic error from the raw data.

The second step is filtering that helps minimize the stochastic error in the calibrated data. The most commonly used filters are the standard Kalman filter (KF) and the Unscented Kalman filter (UKF). KF and UKF need covariance matrices as parameters for the computation.

The third step is the registration, where the data measured by $Gyro_i$ should be transformed to a common coordinate system, for example taking the coordinate system of $Gyro_0$ as the origin coordinate system. The registration can also be performed offline and the output transformation matrix from the registration can be used in the registration component.

The next section explains the processing steps of the pipeline.

9.1.1.2. Processing

Synchronization is the fourth step in the pipeline, where the gyroscope data coming from all sensors should be adjusted to the same time base.

Only after all of the previous steps are performed, the data can be fused. In this research, only decentralized sensor fusion such as averaging and federated Kalman filter were investigated.

The last step is called integration: orientation is calculated by integrating the angular velocity over time. Various numerical integration methods can be used, although in practice standard reliable numerical methods like Runge-Kutta 4^{th} order normalized are commonly used. The orientation is defined by the following differential equation:

$$\dot{q} = \frac{1}{2}\,\Omega(\omega(t))\,q, \tag{9.1}$$

where $\Omega(\omega(t))$ is the operator which turns the considered 3-dimensional angular velocity

into the real skew-symmetric matrix representation, that is:

$$\mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}.$$

The output of numerical integration is a quaternion, representing the orientation of the object, which needs to be normalized:

$$q_{out} = \frac{q_{out}}{\|q_{out}\|}. \tag{9.2}$$

By performing all of the steps from measurement to integration, the object's orientation is obtained.

9.1.2. Accelerometer: From Acceleration to Position

9.1.2.1. Preprocessing

As shown in Figure 9.1, there are steps that should be performed to get an improved data measured by the accelerometer.

The preprocessing for accelerometer data follows the same steps as the preprocessing for gyroscopes, namely, calibration and filtering are performed to eliminate deterministic and stochastic sensor errors respectively. The next step is the registration, where the data measured by Acc_i should be transformed to a common coordinate system, for example taking the coordinate system of Acc_0 as the origin coordinate system.

Accelerometer data processing includes an additional step as compared with the gyroscope. The fourth step is gravity removal.

The measured acceleration $\vec{a}$ consists of the pure acceleration $\vec{a}_*$ plus the gravity vector $\vec{g}$:

$$\vec{a} = \vec{a}_* + \vec{g}. \tag{9.3}$$

The pure acceleration $\vec{a}_*$ can be determined by subtracting the gravity vector $\vec{g}$ from the acceleration $\vec{a}$

$$\vec{a}_* = \vec{a} - \vec{g}. \tag{9.4}$$

For every IMU placement, gravity can be computed as

$$\vec{g} = rotate(\vec{g}_{prev}, q), \tag{9.5}$$

where $\vec{g}_{prev}$ is the previous gravity and q the newly calculated quaternion.

It is extremely important to know the initial orientation of the gravity vector in a specific IMU coordinate system to have a good initial gravity vector representation at the initialization of the Multi-IMU system.

9.1.2.2. Processing

Next, the fifth step is synchronization, where the accelerometer data coming from all sensors should be adjusted to a similar time base. After the data are synchronized, they can be fused. Synchronization and fusion are performed similarly to the corresponding gyroscope operations. Moreover, fusion is usually performed for both sensors simultaneously.

The last step is integration where the position is calculated by integrating twice the acceleration over time. Here numerical integration methods like Euler numerical integration, trapezoid integration, or Runge-Kutta 4^{th} order normalized method can be used. Unlike for orientation and angular rate, there is no differential equation between the acceleration and position. Therefore, it is difficult to get an accurate position. A simple constant error in acceleration can cause a quadratic error in the position. Another factor is the gravity discounting from the acceleration. Here there is a need to know the initial orientation of the gravity vector.

9.1.3. Zero-velocity update

Zero-velocity update (ZUPT) is a strategy that can be used for the enhancement of IMU sensor data while tracking a walking person. The data are enhanced with the processed data obtained with a foot-mounted navigation system. The strategy proposes to correct the velocity, obtained from the accelerometer readings, using the gyroscope data. Namely, the assumption is that if the foot is in a still-phase (i.e., placed on the ground), the velocity should be zero. Then, if the computed velocity is in fact non-zero, it should be corrected. A schematic of the concept is shown in Figure 9.2.

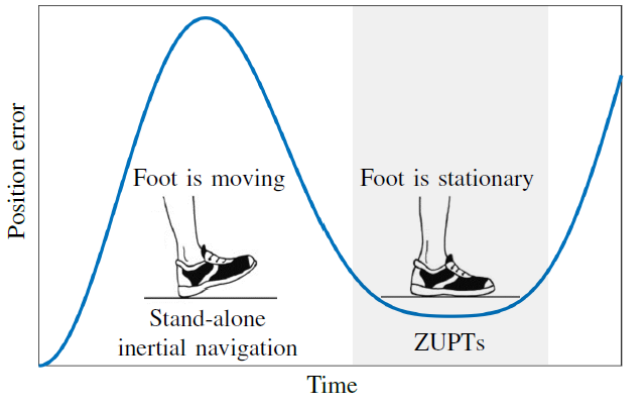


Figure 9.2.: The use of zero-velocity updates. Source: [279]

The performance of solutions obtained with zero-velocity updates depends on the detection of the still-phase. The indication is usually obtained from the gyroscope measurements: If the norm of the angular rate vector is smaller than a chosen threshold value, the velocities should

be reset:

$$\begin{cases} ||\omega_k|| < \delta_{ZUPT}, & \text{still-phase,} \\ \text{otherwise,} & \text{swing-phase,} \end{cases} \tag{9.6}$$

where ω_k is the angular rate vector at the sampling step k in the sensor body coordinate.

Figure 9.3 shows the comparison between a basic velocity estimate and an estimate enhanced with zero-velocity updates.

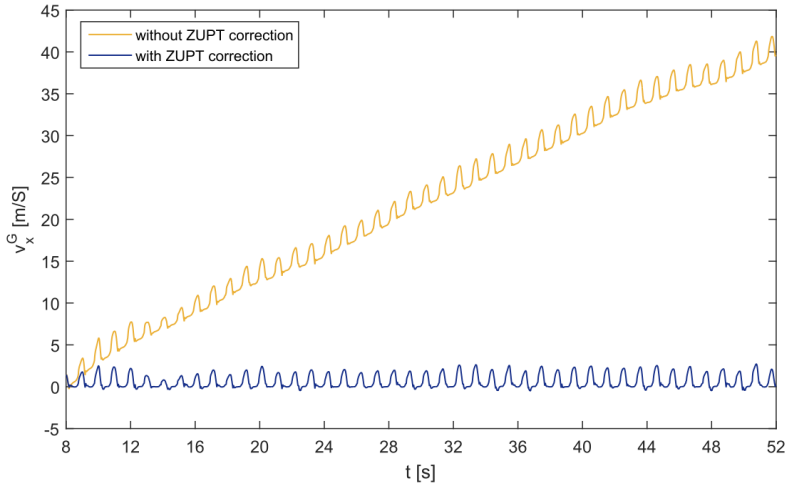


Figure 9.3.: Computed velocity: with and without ZUPT. Source: [302]

9.1.4. Multi-IMU Experiment

This subsection is devoted to the application of the developed pipeline for Multi-IMU pose estimation. The experiments are conducted with the intention to show the robustness of Multi-IMU systems as compared to single-sensor ones.

9.1.4.1. Setup

Physical setup The setup consists of 8 IMUs (BMI160), a multiplexer, a micro-controller, and a dedicated battery. A photo of the physical setup is presented in Figure 9.4. The setup is built on a breadboard with IMUs placed around it in a circular planar pattern and the coordinate systems of the IMUs are aligned in the same orientation (see Figure 9.5).

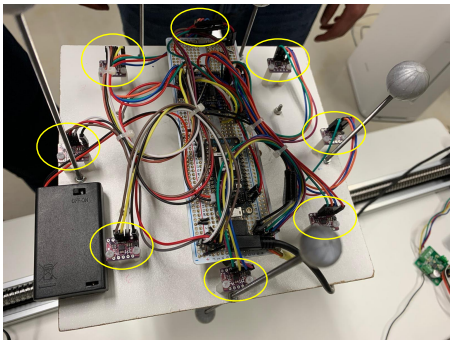


Figure 9.4.: Platform with the assembled Multi-IMU system and a tracking target. IMUs are shown with yellow circles

To collect reference data with an optical tracking system, a large optical target is integrated into the setup (see Figure 9.4). The target consists of 10 retro-reflective spheres in the range of 1-4 cm. The spheres are attached by rods with varying distances to the rim of the platform. Overall, the non-symmetric arrangement covers a volume of approximately 25 $cm \times 25\ cm \times 40\ cm$.

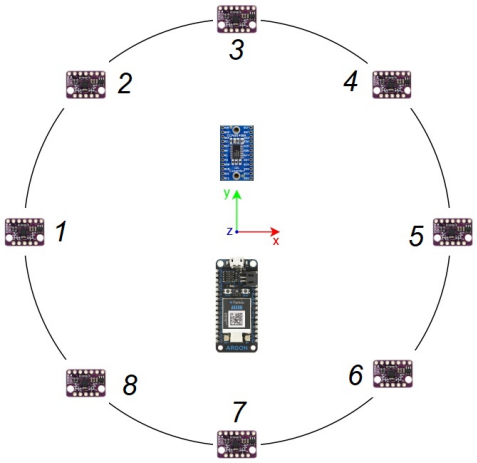


Figure 9.5.: schematic of the setup

The goal of these experiments is to show how the pose of the tracked object can be obtained with a Multi-IMU system. The enhancement of the obtained data is performed to provide reliable adequate results through following the steps of the suggested pipeline.

Adapted pipeline The adapted pipeline is presented in Figure 9.6.

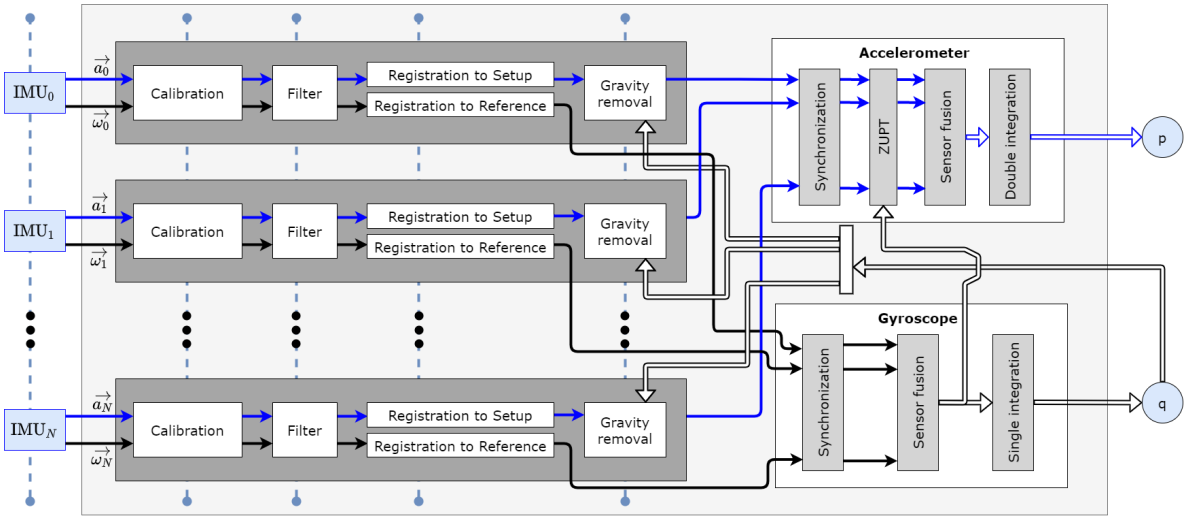


Figure 9.6.: Adapted pipeline for pose estimation with Multi-IMU

The IMUs are calibrated with the method proposed by Tedaldi et al. in [265] and described in Chapter 4. A modification proposed by the author and described in Section 4.4 is used.

Filtering and fusion are performed with a federated Kalman filter, where both local and master filters are UKF. The local filtering is explained in detail in Section 5.3.1.4 and the federated Kalman filtered is presented in Section 8.1.2.2. The fusion procedure is enhanced with the zero-velocity update technique described in Subsection 9.1.3.

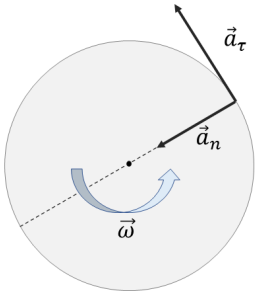


Figure 9.7.: Acceleration in case of rotation

Registration is performed in a specific way, namely, gyroscopes are registered to the center of the plate, while accelerometers are registered to their expected physical locations. The latter is necessary because the error in location can arise during the installation process or can be caused by the inner placement of the sensors within the IMU, and as explained in Chapter 6, even a small variation can lead to large errors. The idea for this specific registration comes from the mechanics. Consider a plate rotation around the center with some angular rate ω. Then the points of the plate (excluding the center point itself) get some acceleration, which

can be represented by the normal (centripetal) and tangential components (see Figure 9.7).

Therefore, by strategically placing the sensors in prescribed positions (see Figure 9.5), registering them to the physical locations (to eliminate possible errors), and then using them in pairs/fours/eights, both normal and tangential components of the acceleration can be eliminated for the case of pure rotation, or compensated for in case of other types of movement.

The multiplexer which is a part of the physical setup allows for data serialization. Serialized data can then be synchronized without time-delay estimation because each measurement gets its own timestamp. Synchronization is then performed with linear interpolation as described in Section 7.2.2.

Importance of zero-velocity update As explained in the corresponding subsection above, the zero-velocity update technique allows diminishing the influence of the accelerometer drifts on the pose estimation results. Zero-velocity updates are used for data enhancement in the filtering stage. The importance of using them can be illustrated with the following comparison.

Consider the following readings from IMU 1 (note: the data are taken from the static experiment presented further): measured acceleration in one of the axes (Figure 9.8) and measured angular rate in one of the axes (Figure 9.9). The curves in the plots show the measured values and the filtered ones. For the sake of simplicity, consider the measurements in other axes to be zero.

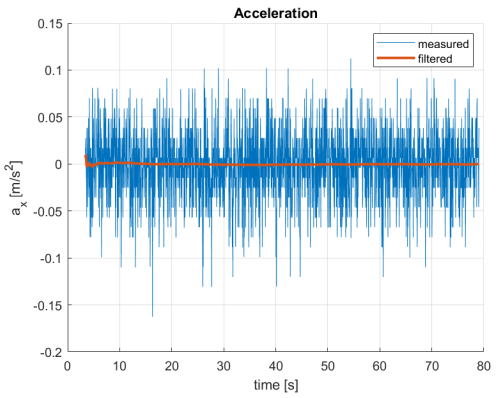


Figure 9.8.: Acceleration

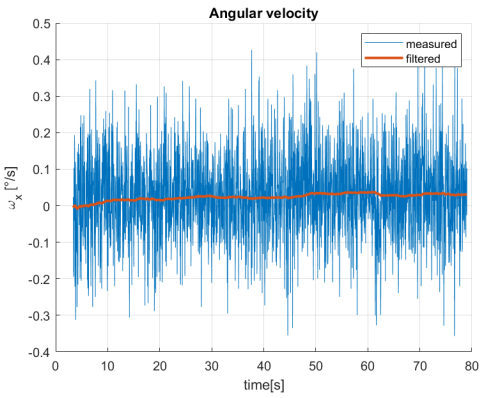


Figure 9.9.: Angular velocity

Now, consider the velocity (Figure 9.10) and position (Figure 9.11) values obtained by integration of acceleration data.

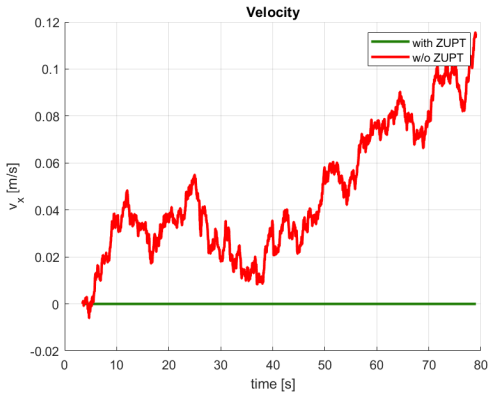


Figure 9.10.: Velocity

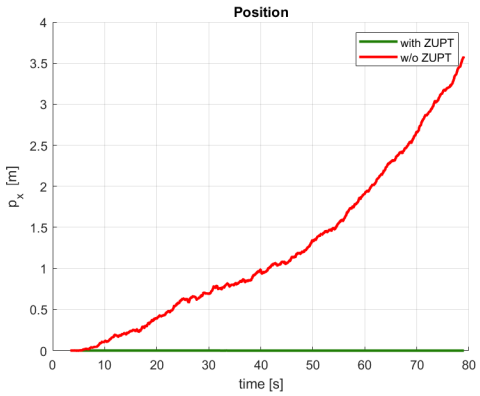


Figure 9.11.: Position

The provided plots show clearly that even within the small time-scale the velocity and position start to drift, although, in reality, the object is static. This shows the importance of the use of additional information about the setup, the model, or the data from other sensors for correcting accelerometer observations. The zero-velocity update technique is used in the forthcoming experiments.

9.1.4.2. Experiment: Static

In this experiment, the data were collected from the Multi-IMU sensors while the tracked object was static. The position obtained from the optical tracking system is presented in Figure 9.12.

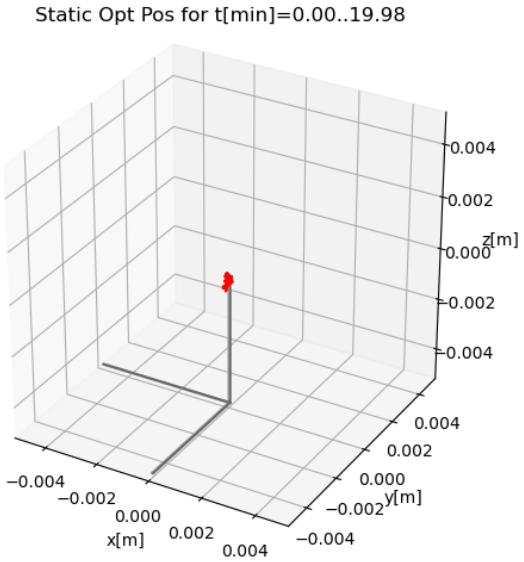


Figure 9.12.: Position in space, tracked by the optical tracker

The results of the pose estimation are presented in the following series of figures. First, no fusion was performed, i.e., the data obtained from a single IMU is presented. Measured enhanced values are presented together with filtered values (for angular velocity and acceleration) along with angles, velocities, and positions, obtained by integration of the enhanced data. Note, that for easier comparison the data tagged as measured are calibrated, registered, and (for acceleration) the gravity is removed, because in static setup gravity removal is trivial.

The orientation data for IMU 1 are shown in Figure 9.13.

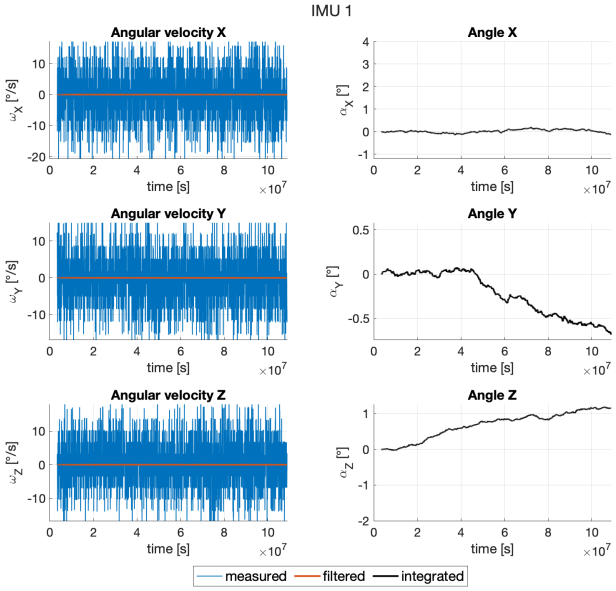


Figure 9.13.: Angular velocity and angle, IMU 1

The position data for IMU 1 are shown in Figure 9.14.

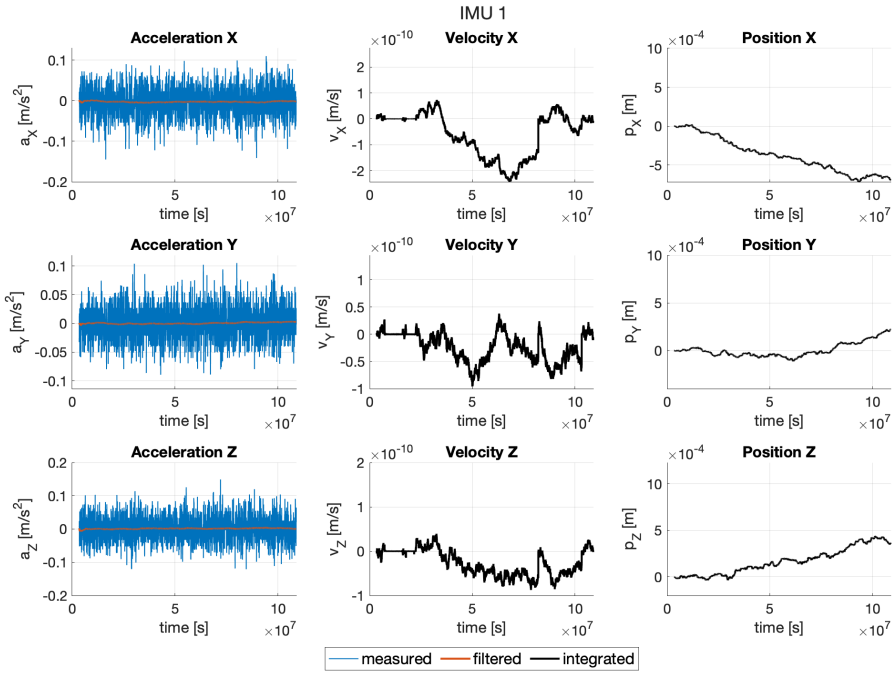


Figure 9.14.: Acceleration, velocity, and position, IMU 1

The orientation data for IMU 2 are shown in Figure 9.15.

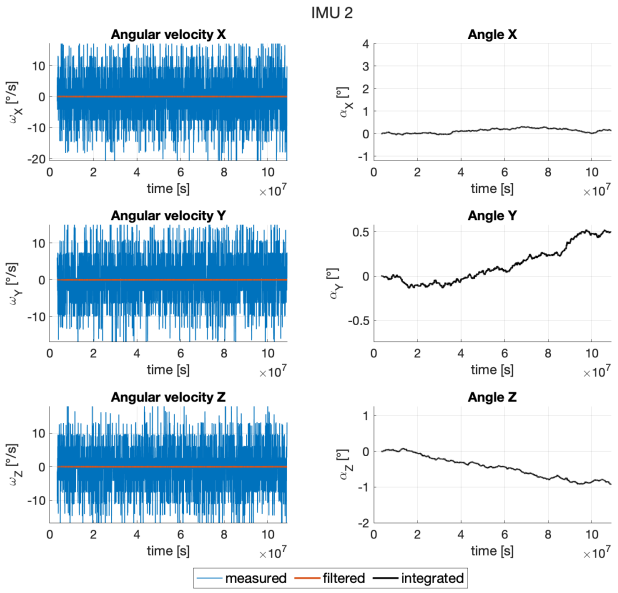


Figure 9.15.: Angular velocity and angle, IMU 2

The position data for IMU 2 are shown in Figure 9.16.

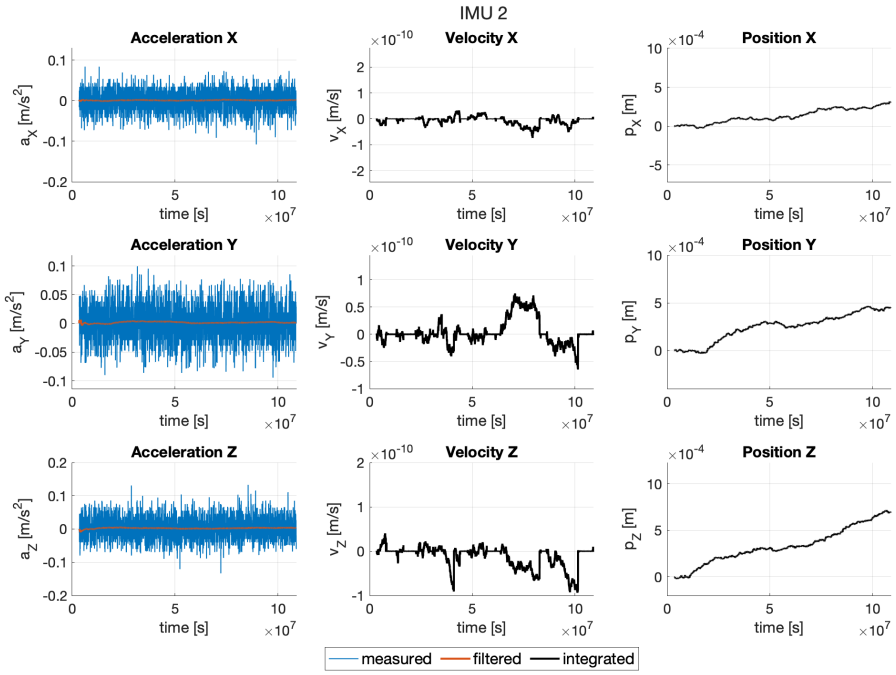


Figure 9.16.: Acceleration, velocity, and position, IMU 2

The orientation data for IMU 3 are shown in Figure 9.17.

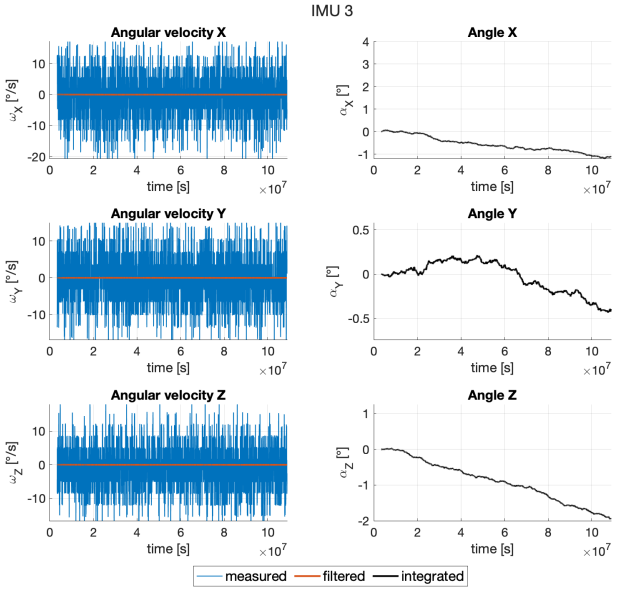


Figure 9.17.: Angular velocity and angle, IMU 3

The position data for IMU 3 are shown in Figure 9.18.

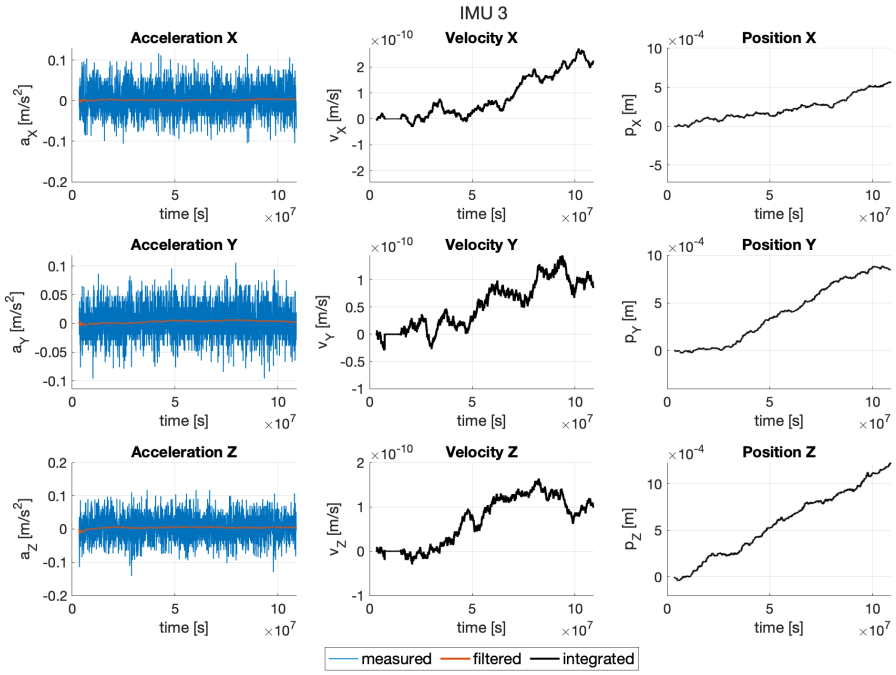


Figure 9.18.: Acceleration, velocity, and position, IMU 3

The orientation data for IMU 4 are shown in Figure 9.19.

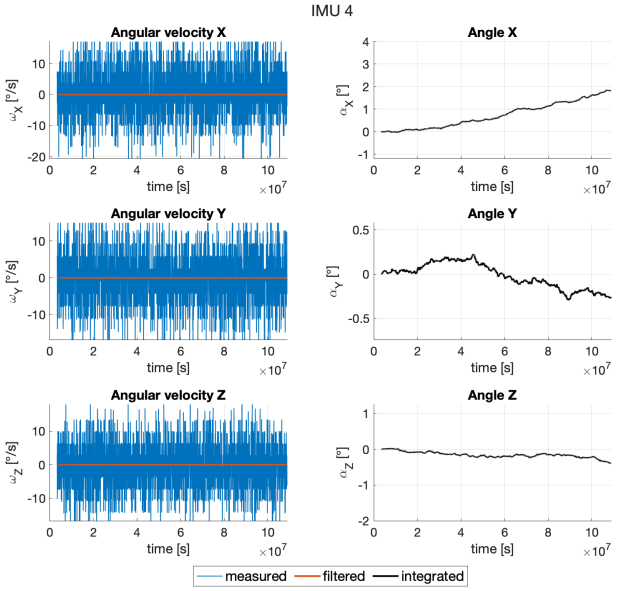


Figure 9.19.: Angular velocity and angle, IMU 4

The position data for IMU 4 are shown in Figure 9.20.

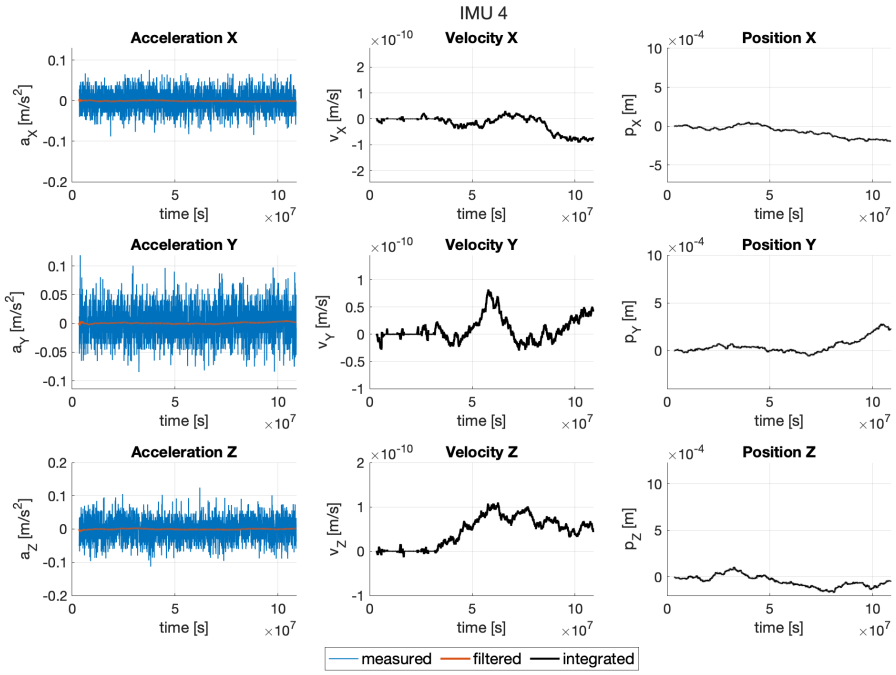


Figure 9.20.: Acceleration, velocity, and position, IMU 4

The orientation data for IMU 5 are shown in Figure 9.21.

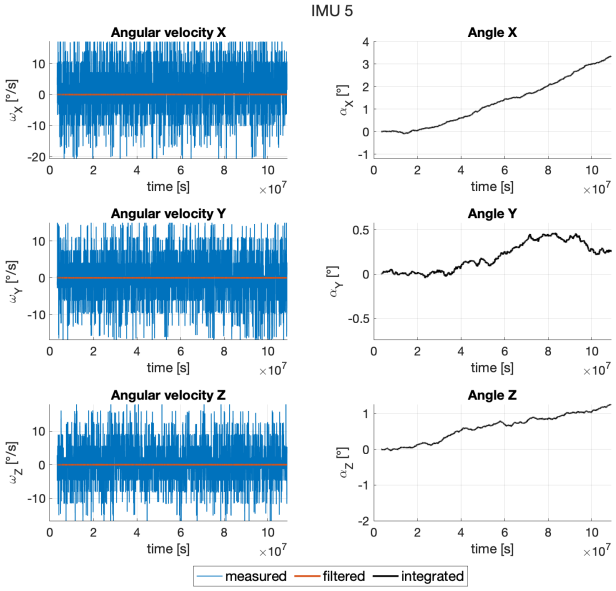


Figure 9.21.: Angular velocity and angle, IMU 5

The position data for IMU 5 are shown in Figure 9.22.

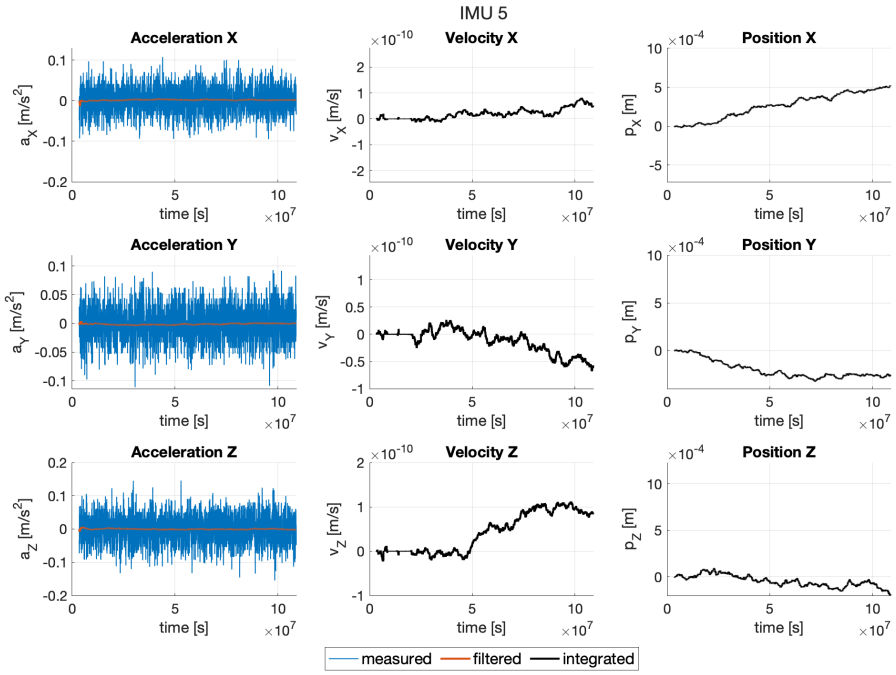


Figure 9.22.: Acceleration, velocity, and position, IMU 5

The orientation data for IMU 6 are shown in Figure 9.23.

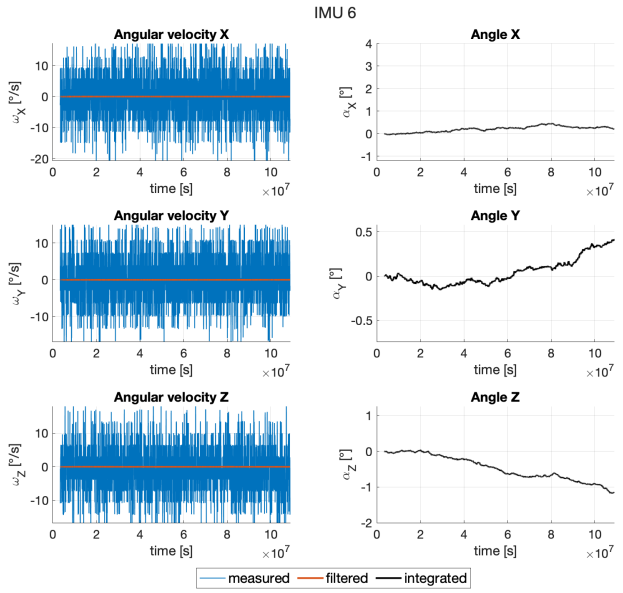


Figure 9.23.: Angular velocity and angle, IMU 6

The position data for IMU 6 are shown in Figure 9.24.

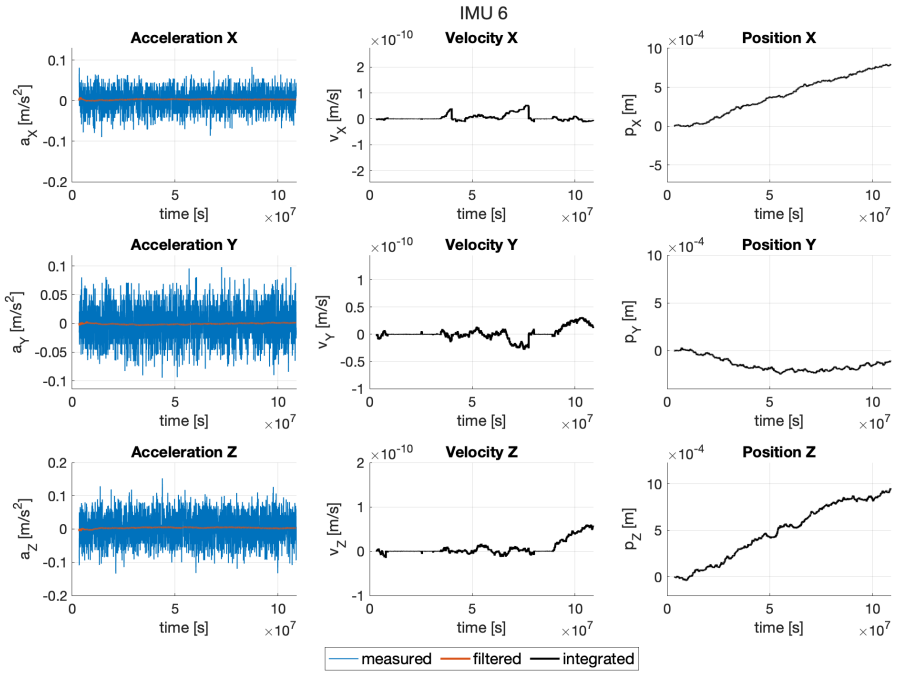


Figure 9.24.: Acceleration, velocity, and position, IMU 6

The orientation data for IMU 7 are shown in Figure 9.25.

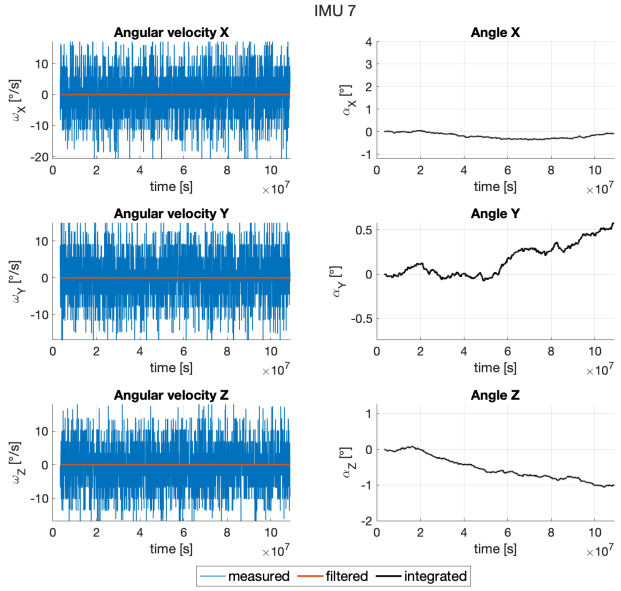


Figure 9.25.: Angular velocity and angle, IMU 7

The position data for IMU 7 are shown in Figure 9.26.

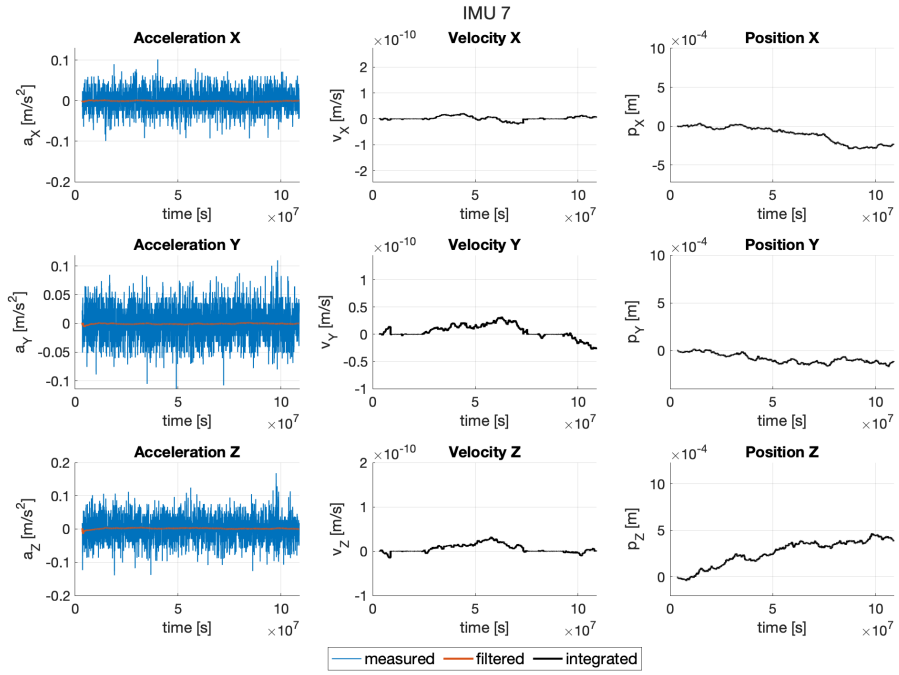


Figure 9.26.: Acceleration, velocity, and position, IMU 7

The orientation data for IMU 8 are shown in Figure 9.27.

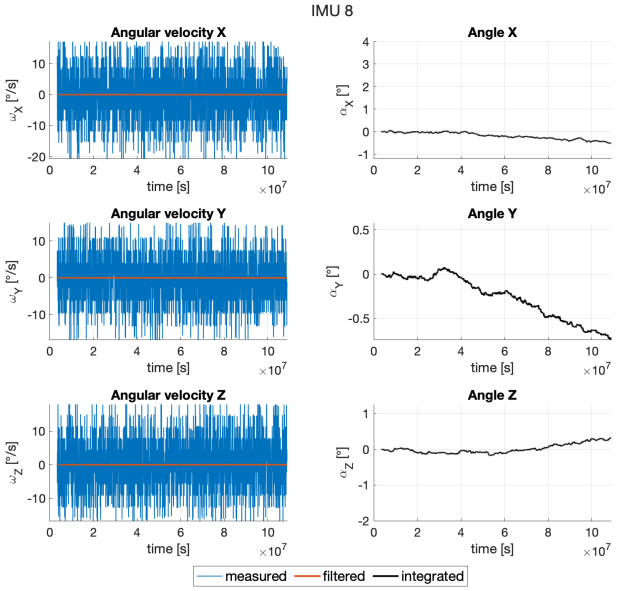


Figure 9.27.: Angular velocity and angle, IMU 8

The position data for IMU 8 are shown in Figure 9.28.

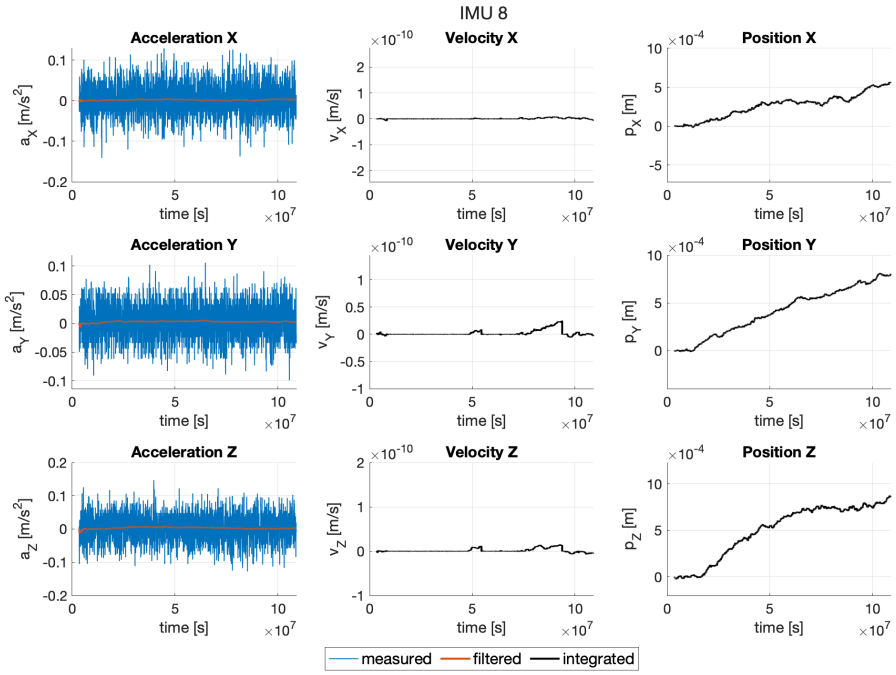


Figure 9.28.: Acceleration, velocity, and position, IMU 8

Next, the data is fused for the couples of paired IMUs: 1 and 5, 2 and 6, 3 and 7, and 4 and 8. These pairs of IMUs are chosen as they are located across from each other in the physical setting, meaning that with the implemented registration method the noise can be further reduced by fusion of the chosen IMUs.

The orientation data for IMUs 1 and 5 are shown in Figure 9.29.

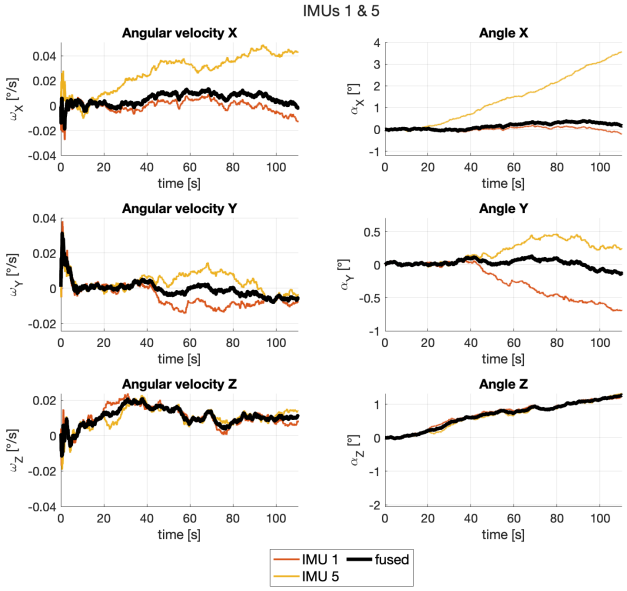


Figure 9.29.: Angular velocity and angle, IMUs 1 & 5

The position data for IMUs 1 and 5 are shown in Figure 9.30.

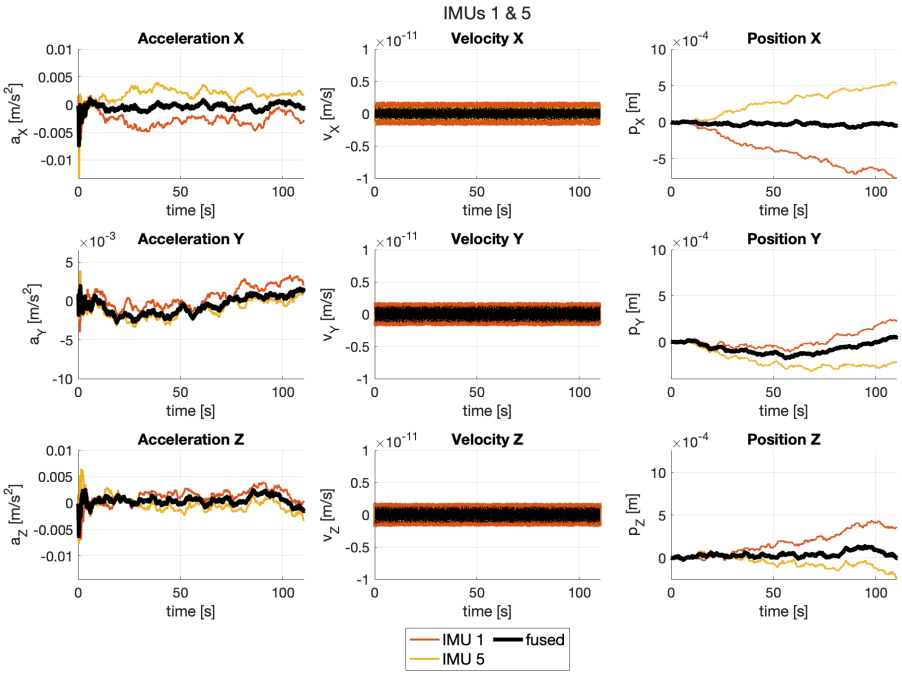


Figure 9.30.: Acceleration, velocity, and position, IMUs 1 & 5

The curve corresponding to the fused data shows good correspondence to the expected results. Also note that all components of velocity are basically zero (at the magnitude of 10^{-11} m/s), so one cannot see both of the original IMU data curves because they together with the fused one are almost zero and overlap drastically.

The orientation data for IMUs 2 and 6 are shown in Figure 9.31.

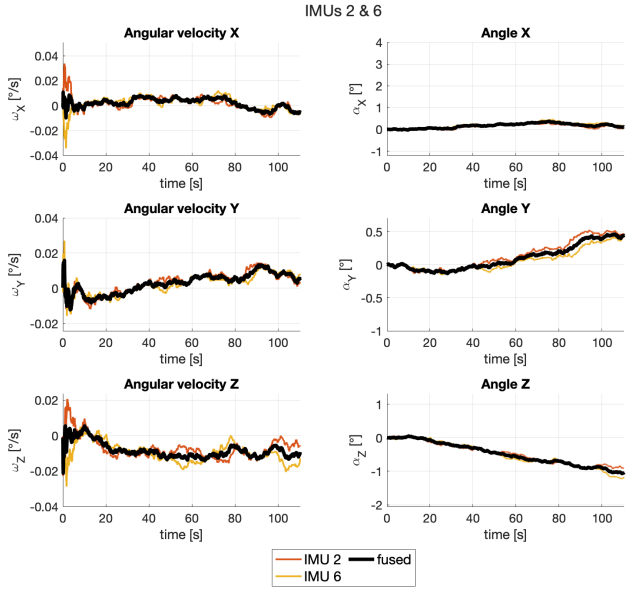


Figure 9.31.: Angular velocity and angle, IMUs 2 & 6

The position data for IMUs 2 and 6 are shown in Figure 9.32.

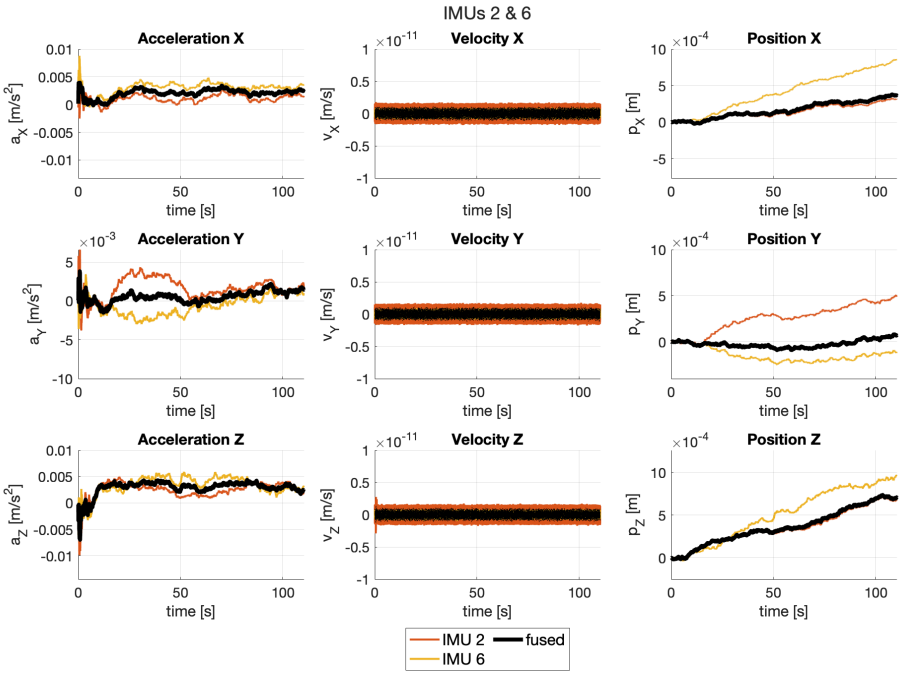


Figure 9.32.: Acceleration, velocity, and position, IMUs 2 & 6

The orientation data for IMUs 3 and 7 are shown in Figure 9.33.

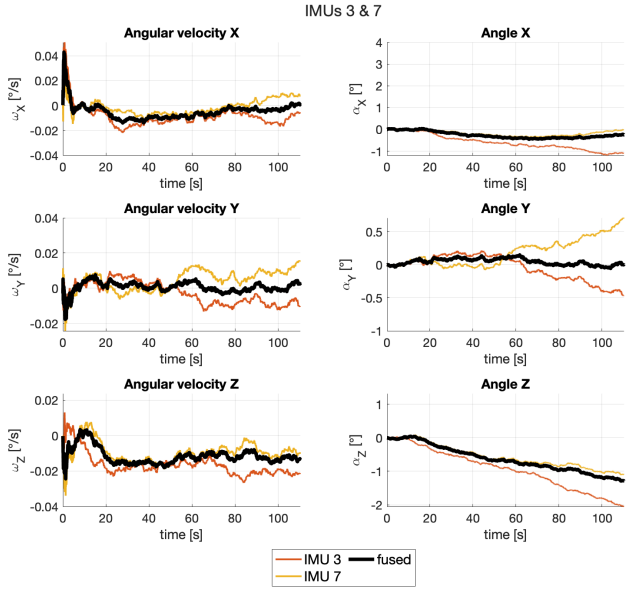


Figure 9.33.: Angular velocity and angle, IMUs 3 & 7

The position data for IMUs 3 and 7 are shown in Figure 9.34.

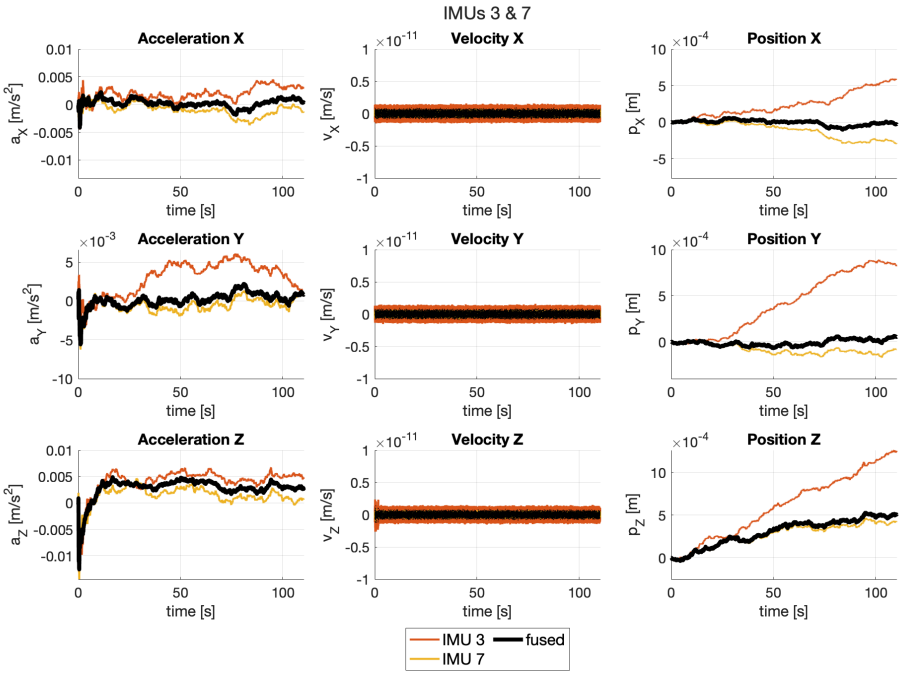


Figure 9.34.: Acceleration, velocity, and position, IMUs 3 & 7

The orientation data for IMUs 4 and 8 are shown in Figure 9.35.

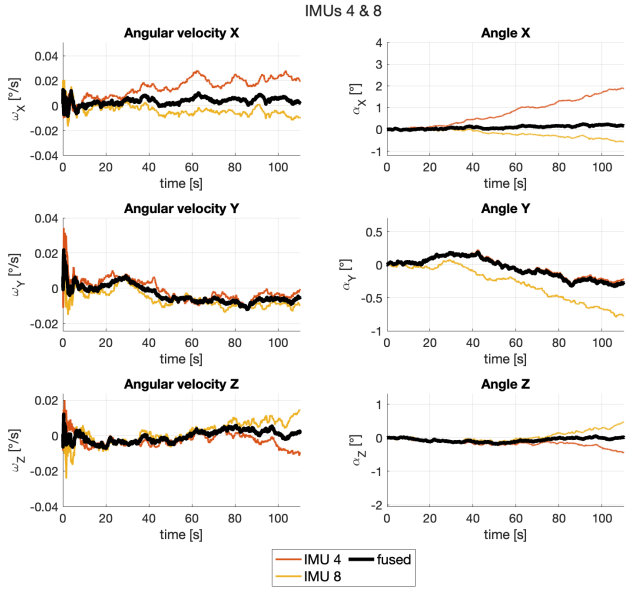


Figure 9.35.: Angular velocity and angle, IMUs 4 & 8

The position data for IMUs 4 and 8 are shown in Figure 9.36.

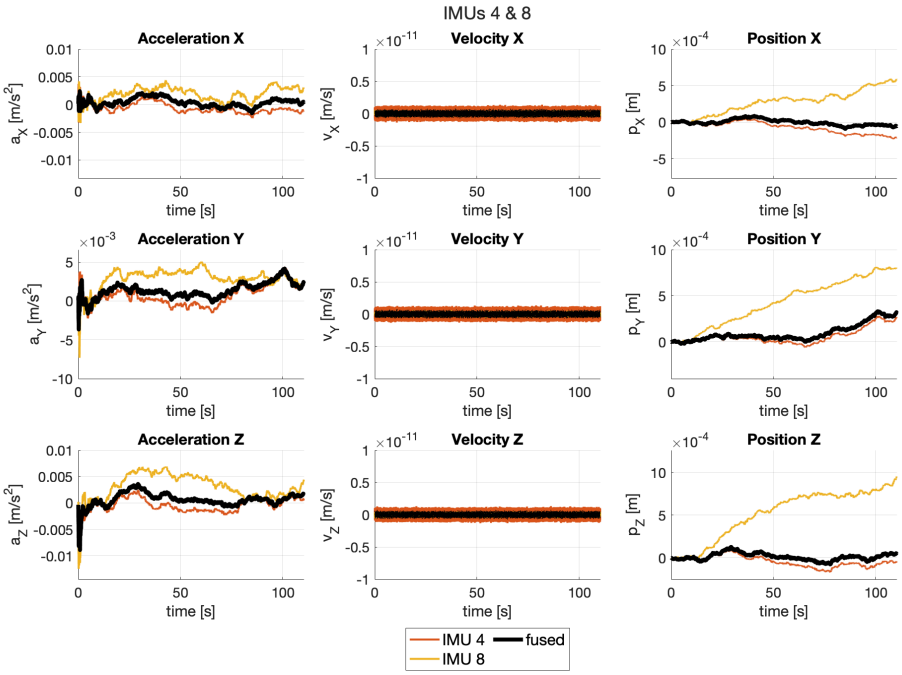


Figure 9.36.: Acceleration, velocity, and position, IMUs 4 & 8

Next, the IMUs are fused in fours, namely consider the group of IMUs 1, 3, 5, 7 and 2, 4, 6, 8.

The orientation data for IMUs 1, 3, 5, 7 are shown in Figure 9.37.

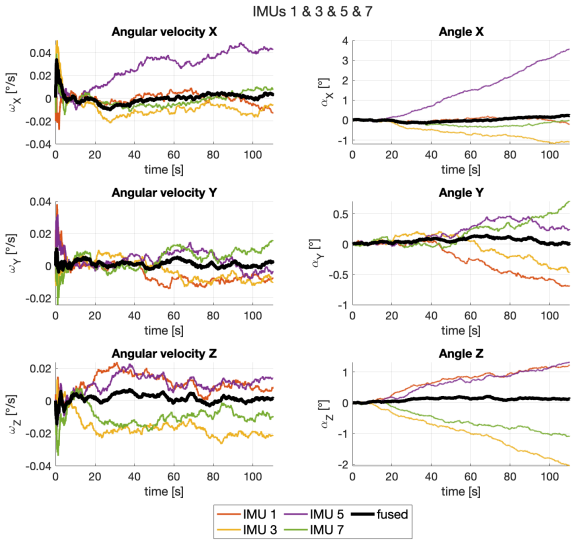


Figure 9.37.: Angular velocity and angle, IMUs 1&3&5&7

The position data for IMUs 1, 3, 5, 7 are shown in Figure 9.38.

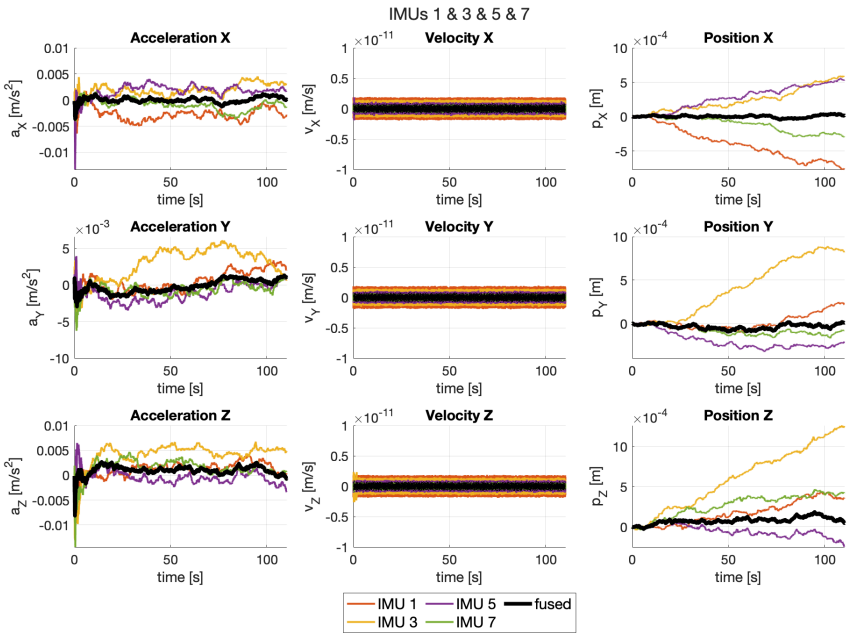


Figure 9.38.: Acceleration, velocity, and position, IMUs 1&3&5&7

The orientation data for IMUs 2, 4, 6, 8 are shown in Figure 9.39.

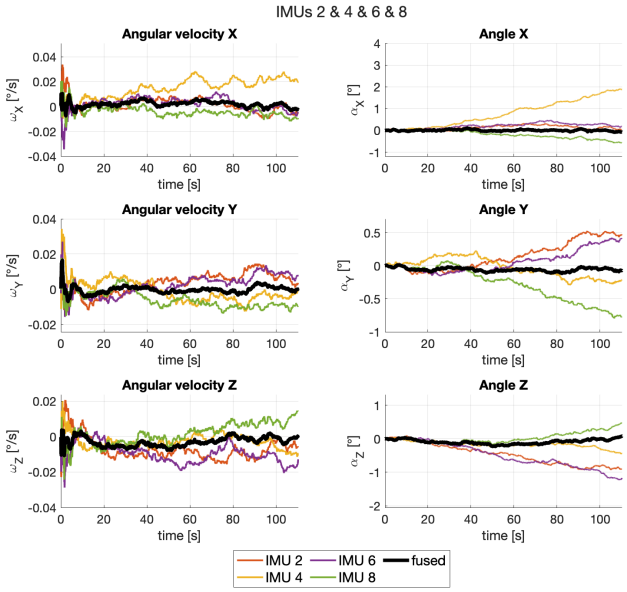


Figure 9.39.: Angular velocity and angle, IMUs 2&4&6&8

The position data for IMUs 2, 4, 6, 8 are shown in Figure 9.40.

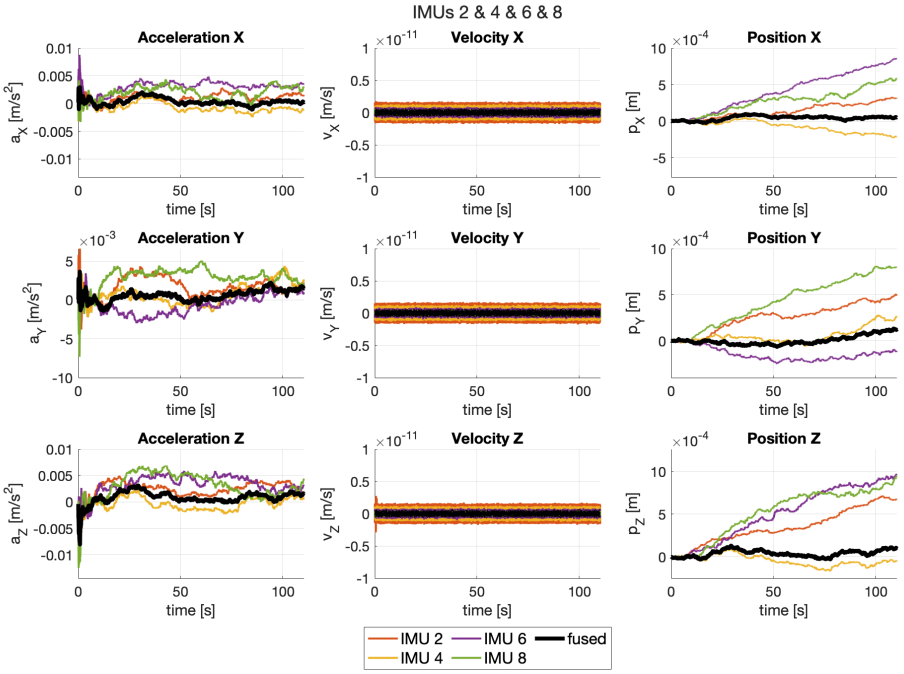


Figure 9.40.: Acceleration, velocity, and position, IMUs 2&4&6&8

Now, the fusion of all 8 IMUs is considered.
The orientation data for all 8 IMUs are shown in Figure 9.41.

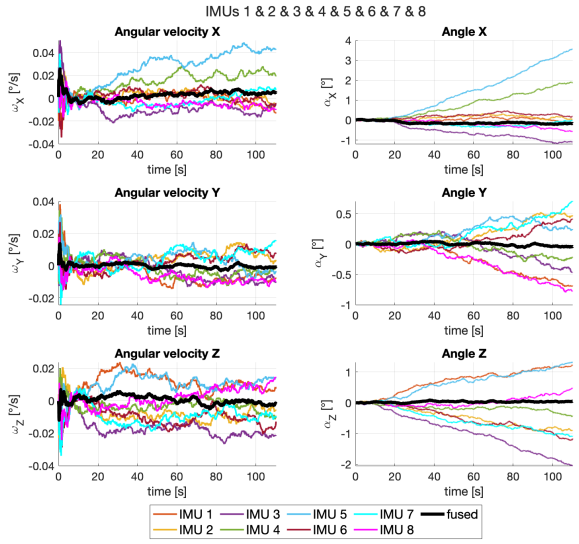


Figure 9.41.: Angular velocity and angle,
IMUs 1&2&3&4&5&6&7&8

The position data for all 8 IMUs are shown in Figure 9.42.

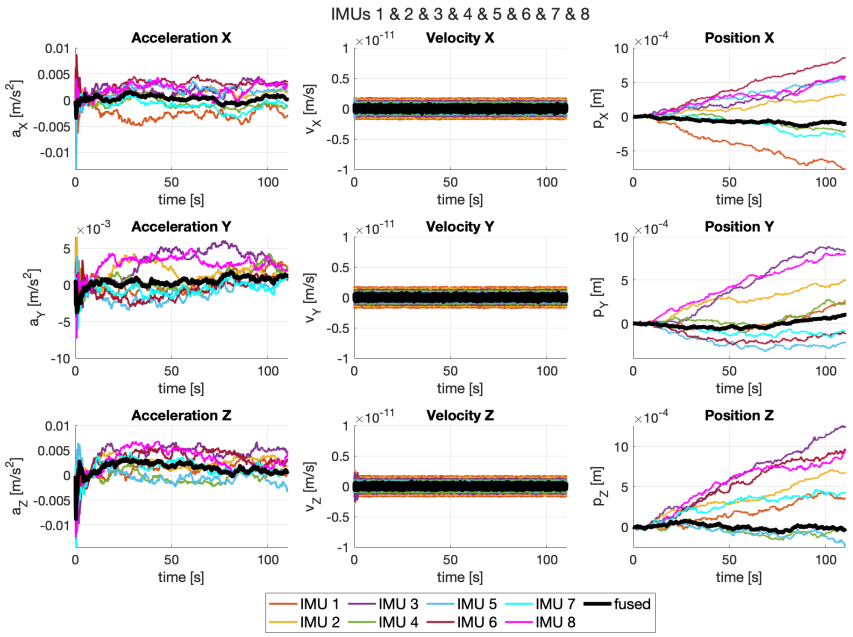


Figure 9.42.: Acceleration, velocity, and position,
IMUs 1&2&3&4&5&6&7&8

The presented experiment shows the advantages of fusing Multi-IMU data over the usage of single IMUs. The results are discussed further in Subsection 9.1.4.5.

9.1.4.3. Experiment: Rotation

In this experiment, the data were collected from the Multi-IMU sensors while the tracked object was rotating on a turn-table. The rotation was performed around the Z-axis corresponding to the point in the center of the plate. Note that the experimental data include a small static period at the beginning of the experiment.

The rotation of one of the targets tracked by the optical tracking system is presented in Figure 9.43. The engine rotated the platform by 180° counterclockwise and then by 180° clockwise.

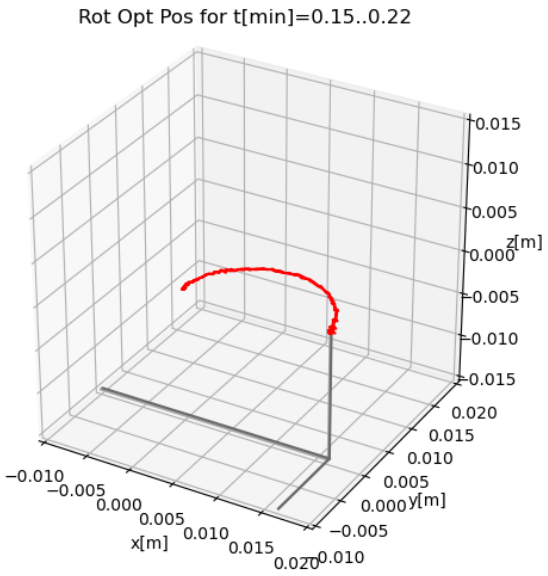


Figure 9.43.: Position change in space, tracked by the optical tracker

The idea for performing this experiment is as follows. The IMUs are placed in a circle around the rotation center, and each IMU has a paired one, located on the other end of the diameter. There is no translation applied to the plate; however, because the IMUs are placed not in the center of the plate but rather closer to the edge, there is a linear acceleration present, measured by the accelerometers. The gyroscopes just register the rotation in Z. The goal is to describe the movement of the plate. Accelerometer data will be used to estimate the position change in the center of the plate by fusing the non-registered data from 2, 4, or 8 IMUs. Gyroscope data will be used to estimate the rotation of the plate.

The results of the pose estimation are presented in the following series of figures. First, no fusion was performed, i.e., the data obtained from a single IMU is presented. Measured enhanced values are presented together with filtered values (for angular velocity and acceleration) along with angles, velocities, and positions, obtained by integration of the enhanced

data. Note, that for easier comparison the data tagged as measured are calibrated, registered, and (for acceleration) the gravity is removed, because in the discussed setup (rotation around the axis where gravity exists) gravity removal is trivial.

The orientation data for IMU 1 are shown in Figure 9.44.

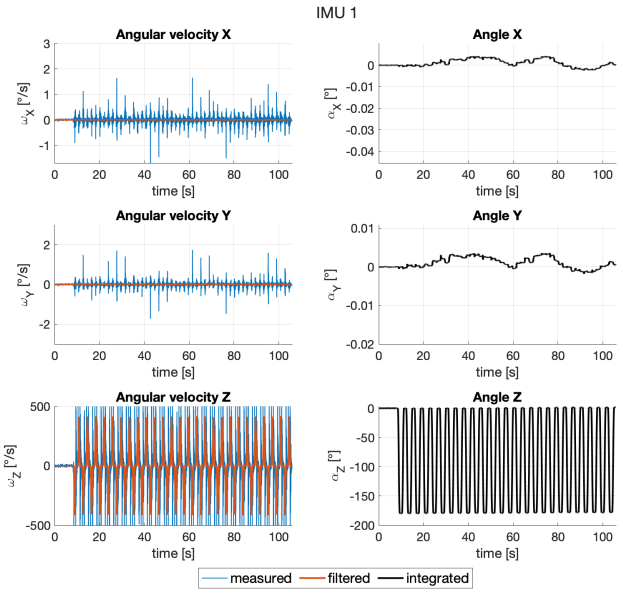


Figure 9.44.: Angular velocity and angle, IMU 1

The position data for IMU 1 are shown in Figure 9.45.

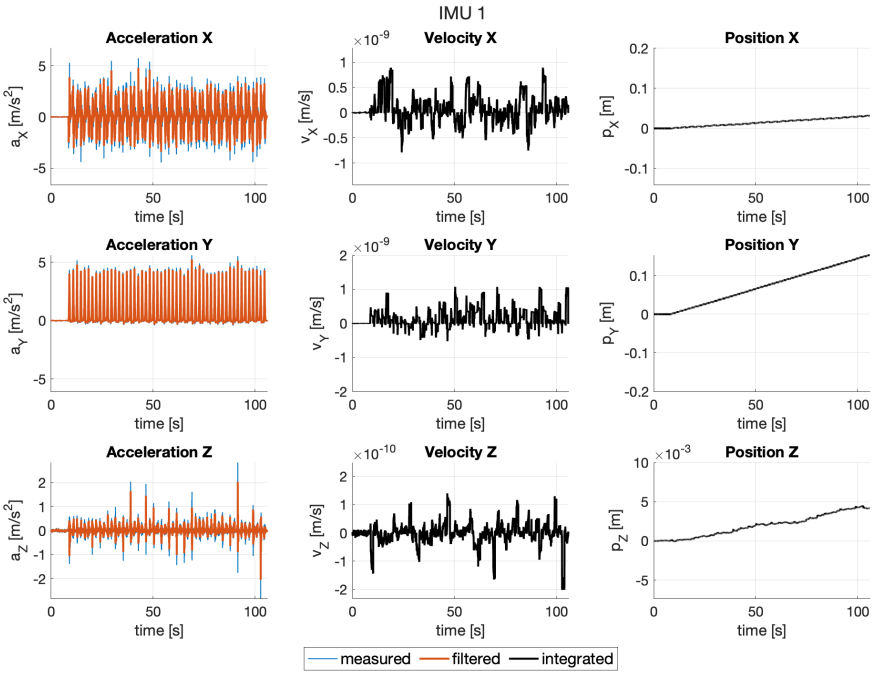


Figure 9.45.: Acceleration, velocity, and position, IMU 1

The orientation data for IMU 2 are shown in Figure 9.46.

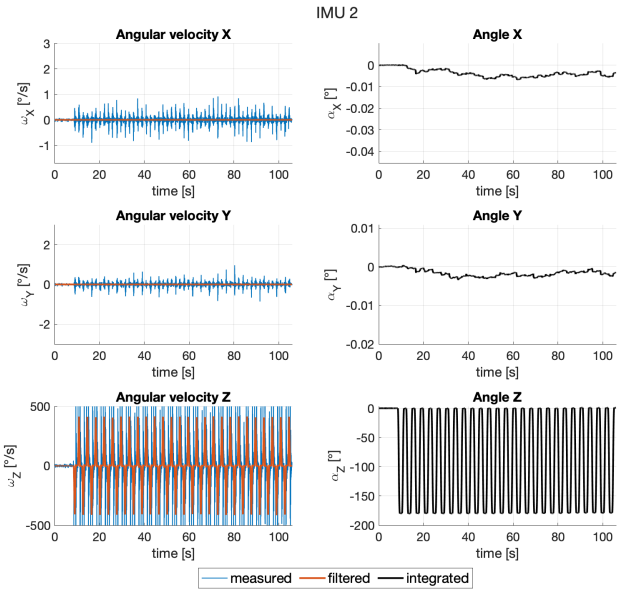


Figure 9.46.: Angular velocity and angle, IMU 2

The position data for IMU 2 are shown in Figure 9.47.

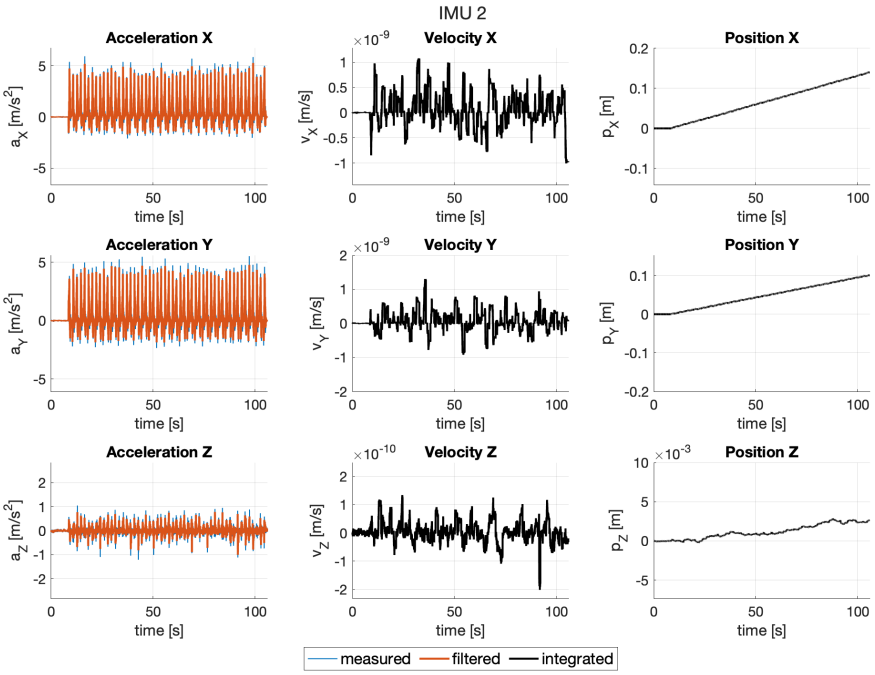


Figure 9.47.: Acceleration, velocity, and position, IMU 2

The orientation data for IMU 3 are shown in Figure 9.48.

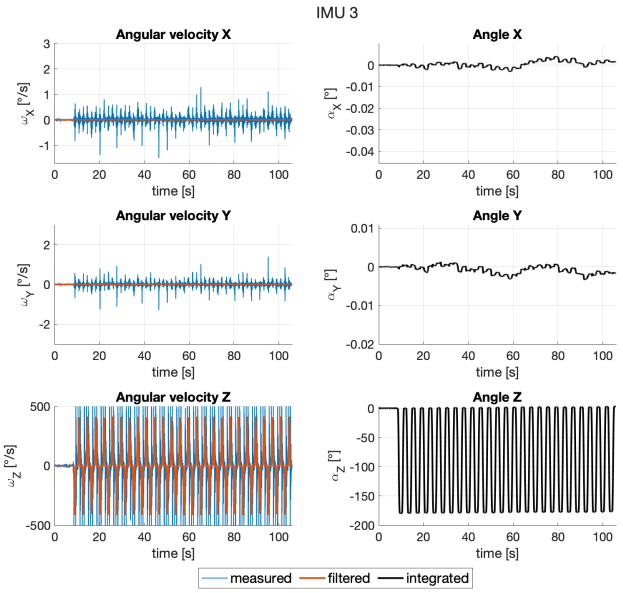


Figure 9.48.: Angular velocity and angle, IMU 3

The position data for IMU 3 are shown in Figure 9.49.

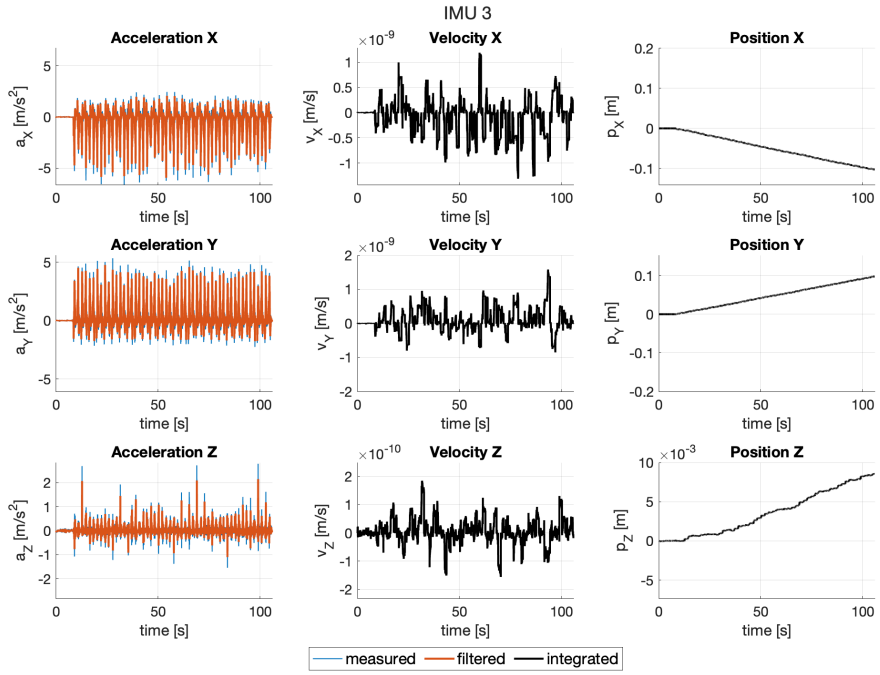


Figure 9.49.: Acceleration, velocity, and position, IMU 3

The orientation data for IMU 4 are shown in Figure 9.50.

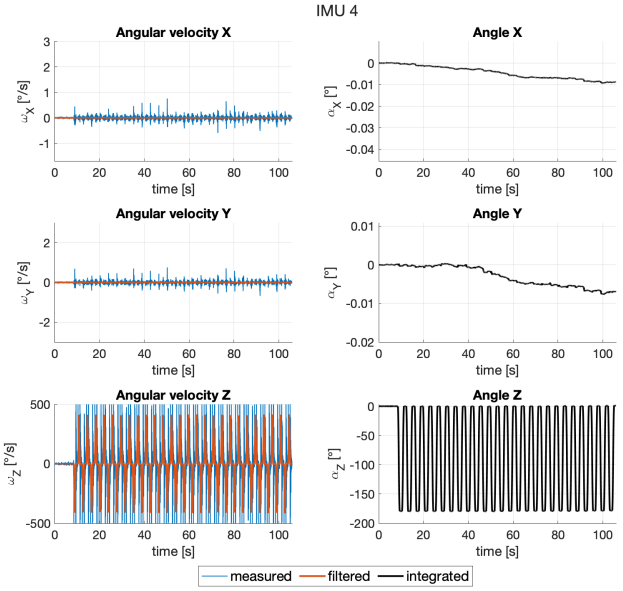


Figure 9.50.: Angular velocity and angle, IMU 4

The position data for IMU 4 are shown in Figure 9.51.

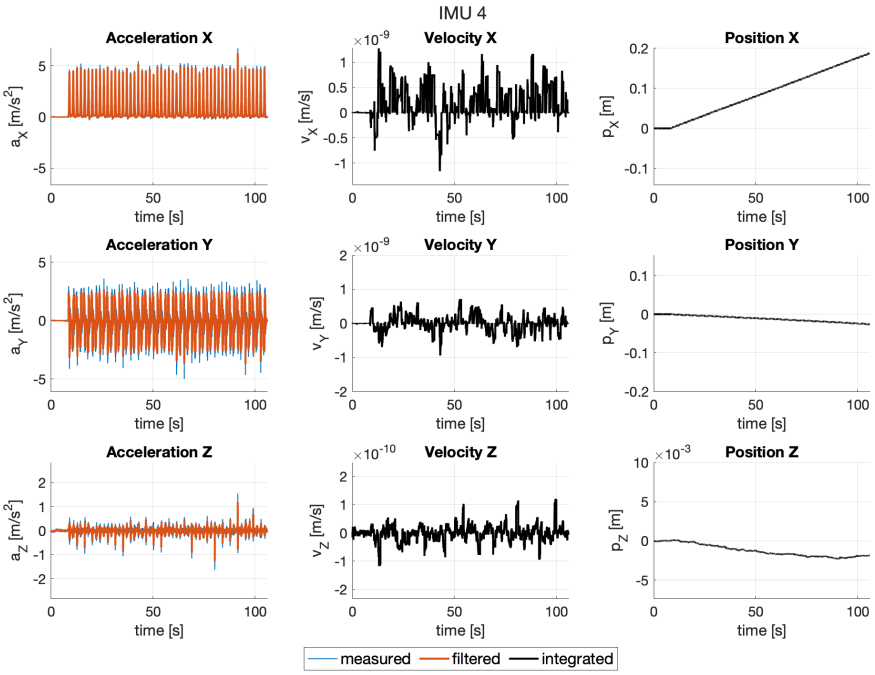


Figure 9.51.: Acceleration, velocity, and position, IMU 4

The orientation data for IMU 5 are shown in Figure 9.52.

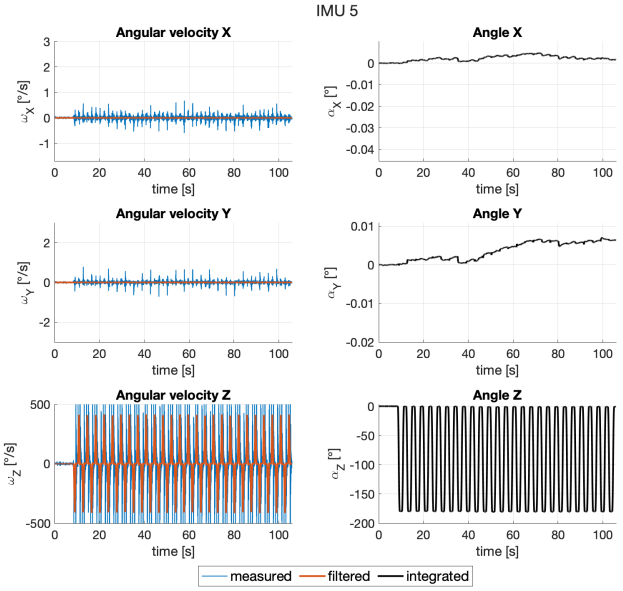


Figure 9.52.: Angular velocity and angle, IMU 5

The position data for IMU 5 are shown in Figure 9.53.

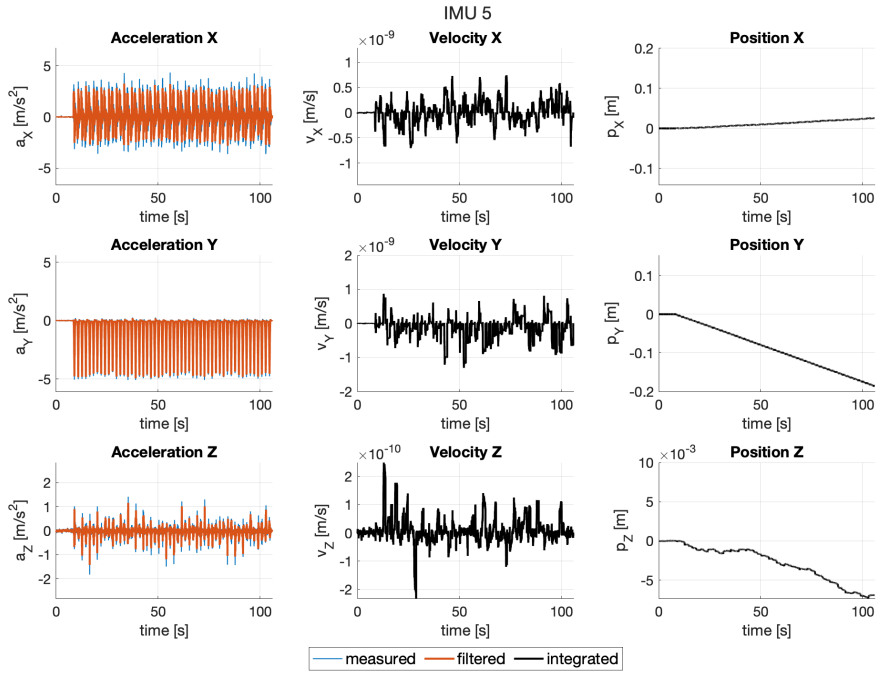


Figure 9.53.: Acceleration, velocity, and position, IMU 5

The orientation data for IMU 6 are shown in Figure 9.54.

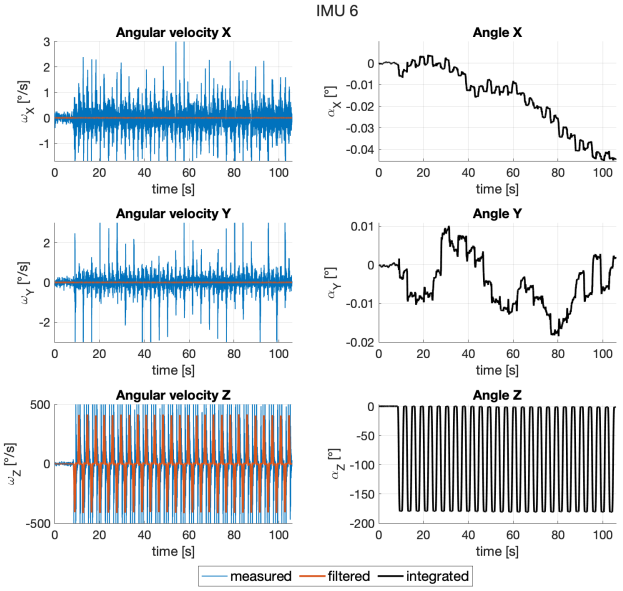


Figure 9.54.: Angular velocity and angle, IMU 6

The position data for IMU 6 are shown in Figure 9.55.

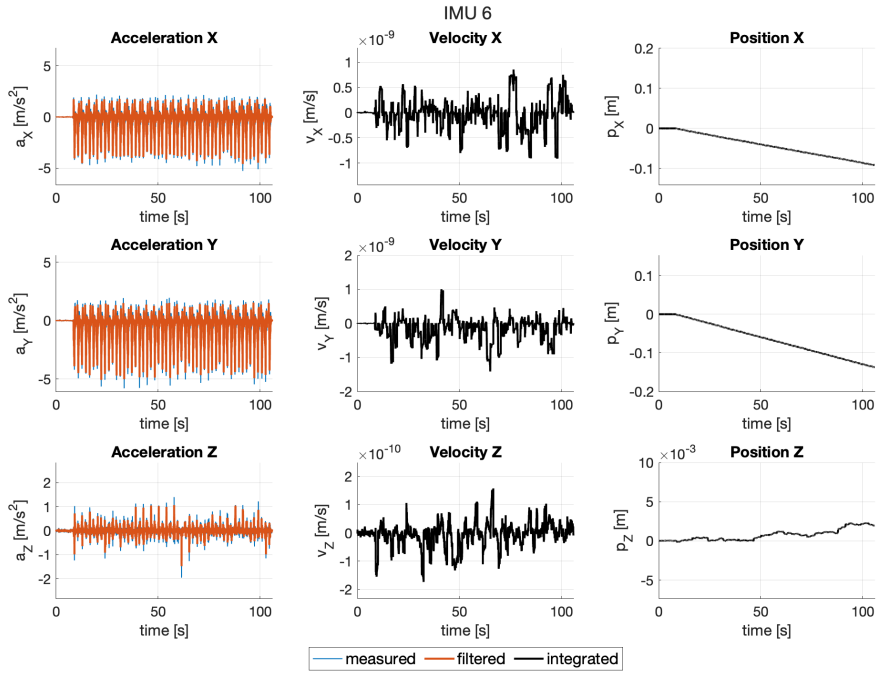


Figure 9.55.: Acceleration, velocity, and position, IMU 6

The orientation data for IMU 7 are shown in Figure 9.56.

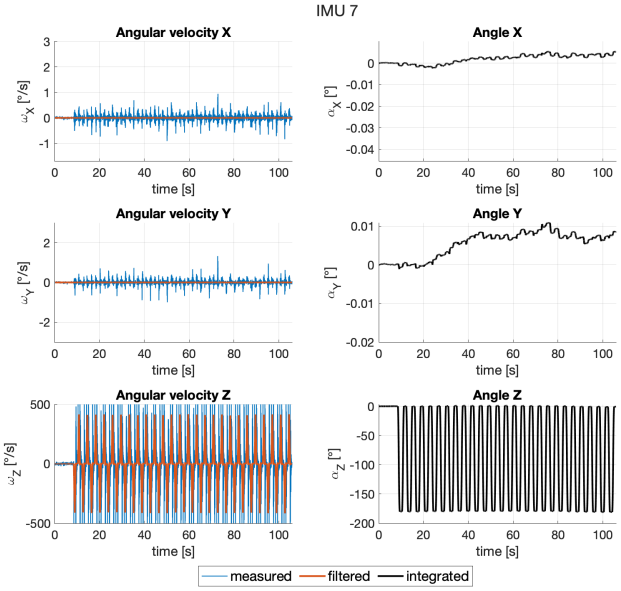


Figure 9.56.: Angular velocity and angle, IMU 7

The position data for IMU 7 are shown in Figure 9.57.

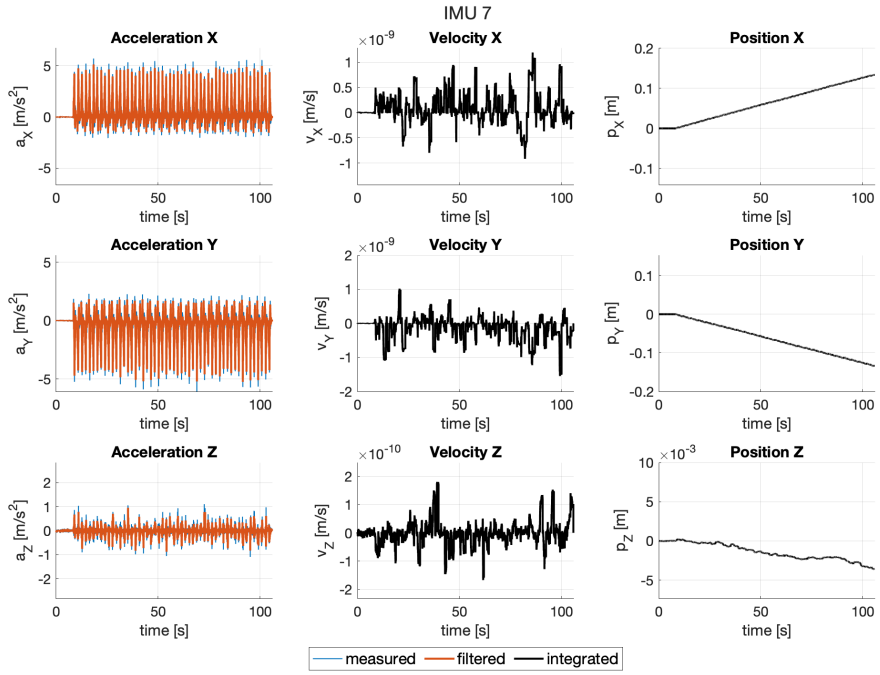


Figure 9.57.: Acceleration, velocity, and position, IMU 7

The orientation data for IMU 8 are shown in Figure 9.58.

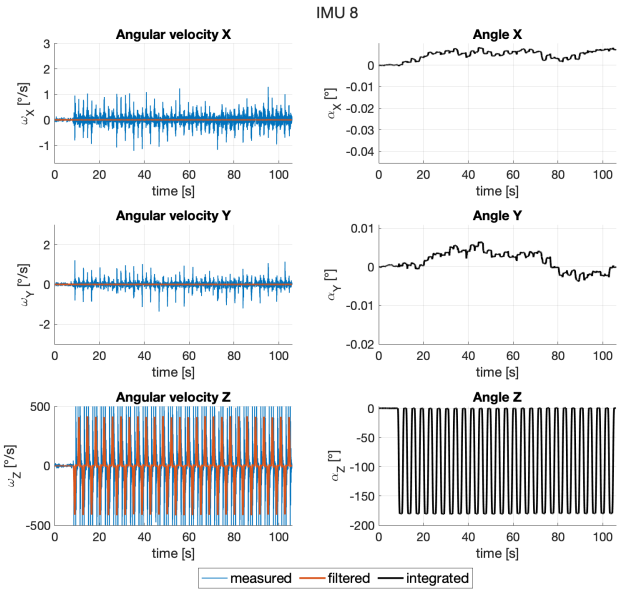


Figure 9.58.: Angular velocity and angle, IMU 8

The position data for IMU 8 are shown in Figure 9.59.

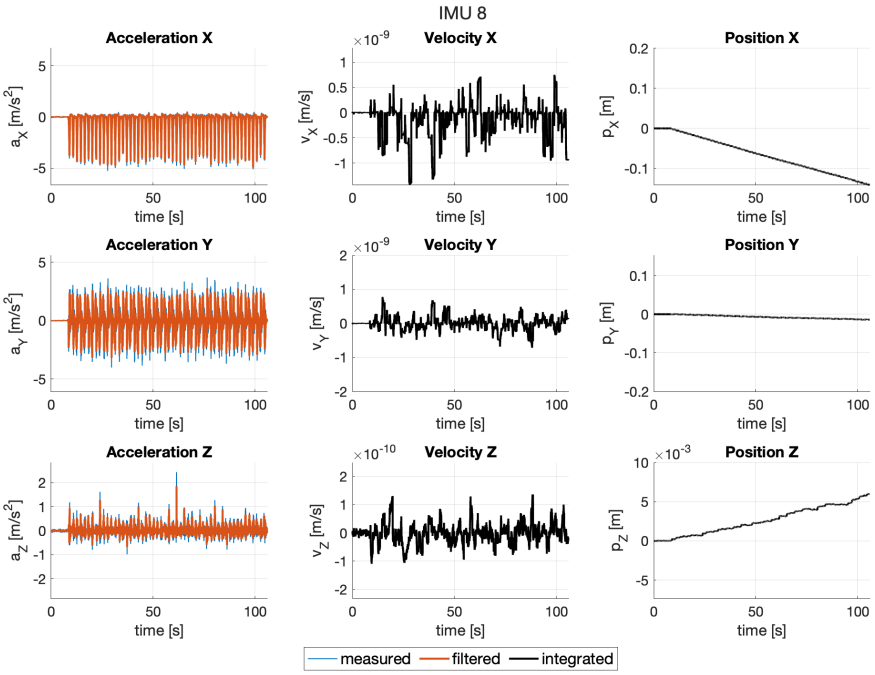


Figure 9.59.: Acceleration, velocity, and position, IMU 8

Next, the data is fused for the couples of paired IMUs: 1 and 5, 2 and 6, 3 and 7, and 4 and 8. These pairs of IMUs are chosen as they are located across from each other in the physical setting, meaning that with the implemented registration method the noise can be further reduced by fusion of the chosen IMUs.

The orientation data for IMUs 1 and 5 are shown in Figure 9.60.

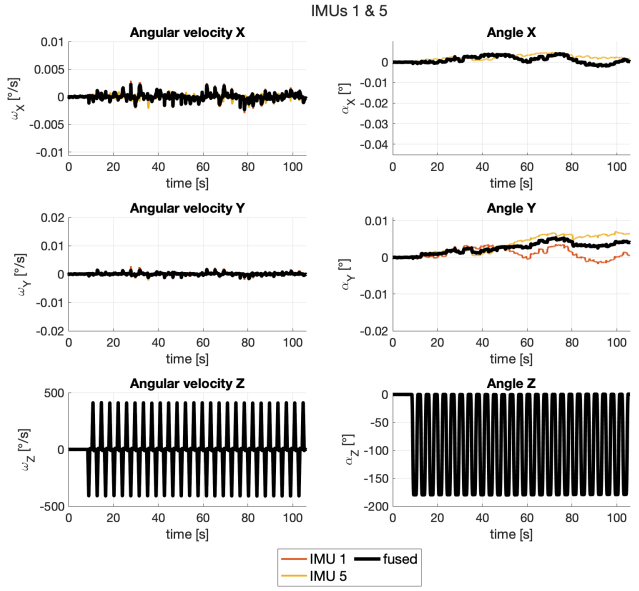


Figure 9.60.: Angular velocity and angle, IMUs 1 & 5

The position data for IMUs 1 and 5 are shown in Figure 9.61.

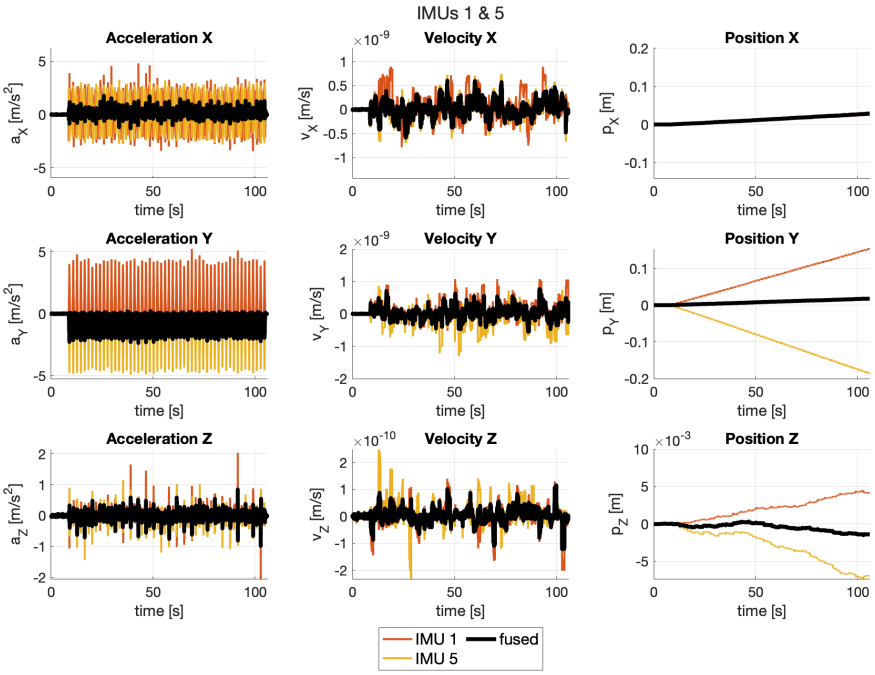


Figure 9.61.: Acceleration, velocity, and position, IMUs 1 & 5

The orientation data for IMUs 2 and 6 are shown in Figure 9.62.

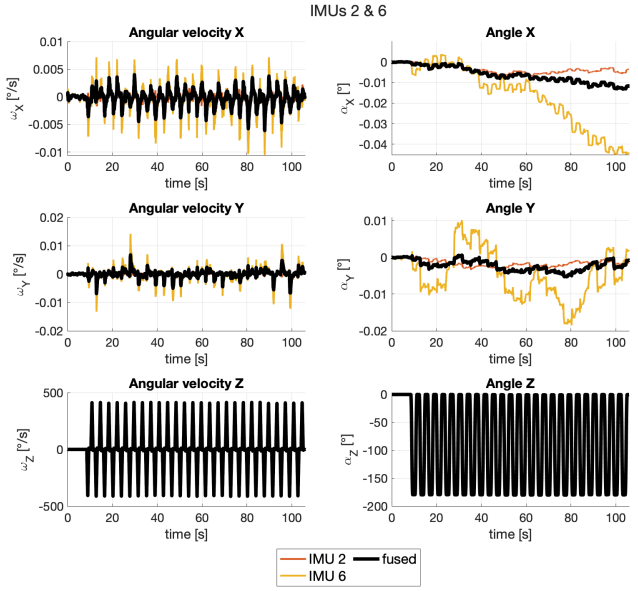


Figure 9.62.: Angular velocity and angle, IMUs 2 & 6

The position data for IMUs 2 and 6 are shown in Figure 9.63.

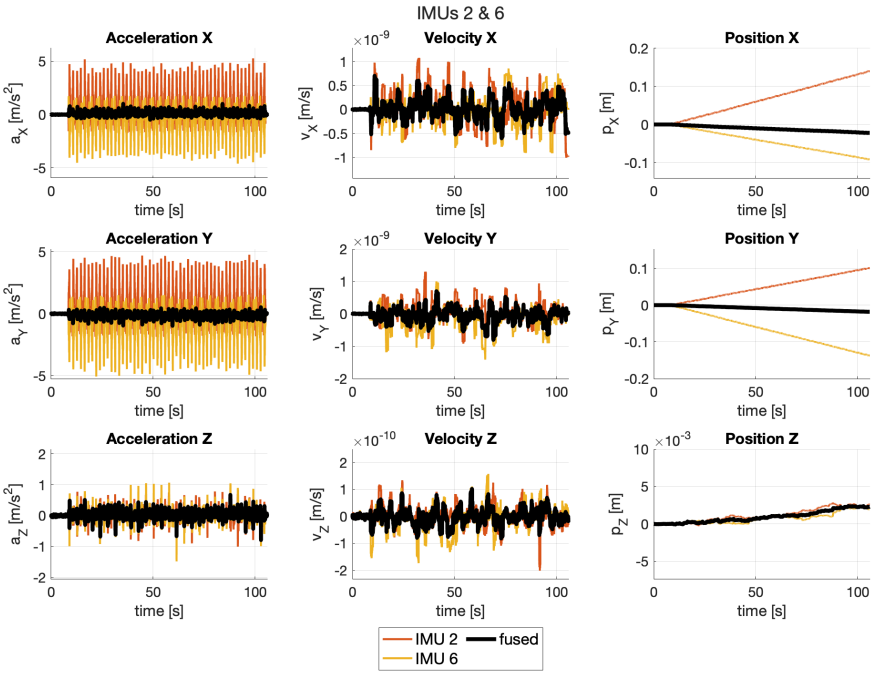


Figure 9.63.: Acceleration, velocity, and position, IMUs 2 & 6

The orientation data for IMUs 3 and 7 are shown in Figure 9.64.

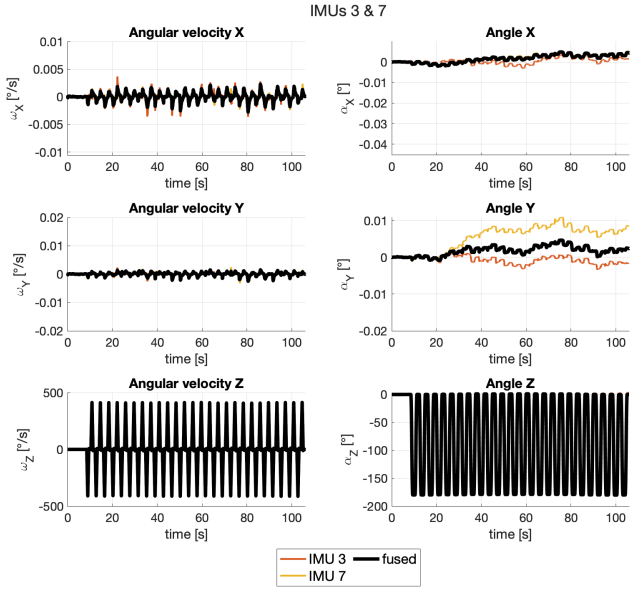


Figure 9.64.: Angular velocity and angle, IMUs 3 & 7

The position data for IMUs 3 and 7 are shown in Figure 9.65.

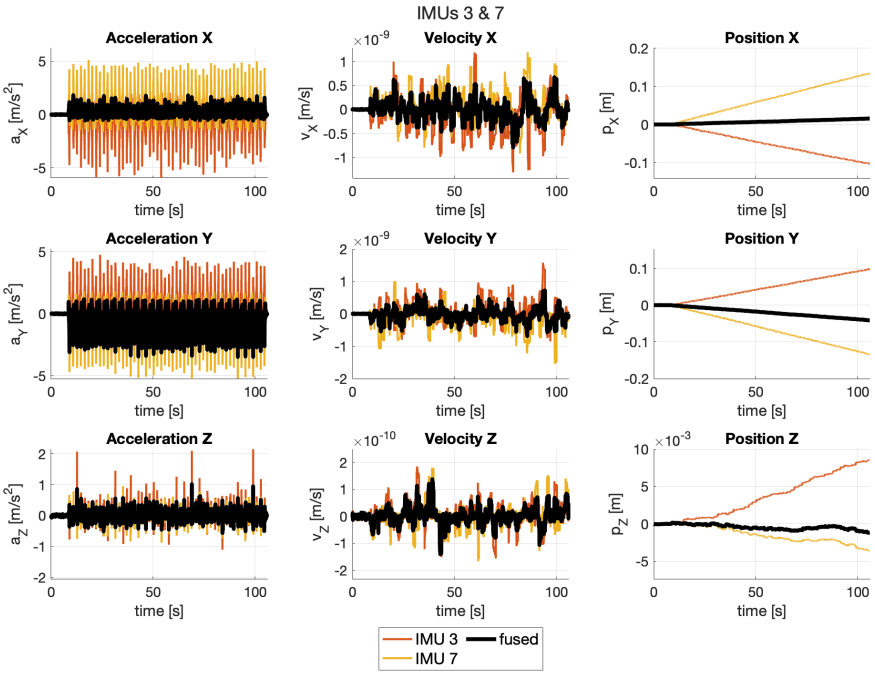


Figure 9.65.: Acceleration, velocity, and position, IMUs 3 & 7

The orientation data for IMUs 4 and 8 are shown in Figure 9.66.

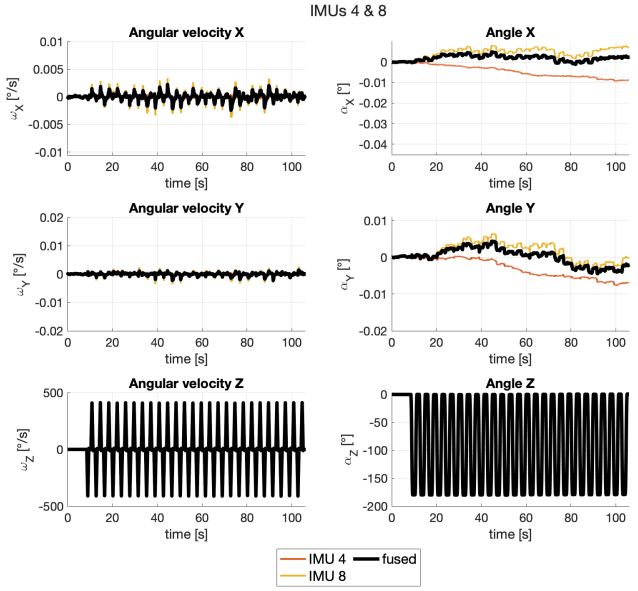


Figure 9.66.: Angular velocity and angle, IMUs 4 & 8

The position data for IMUs 4 and 8 are shown in Figure 9.67.

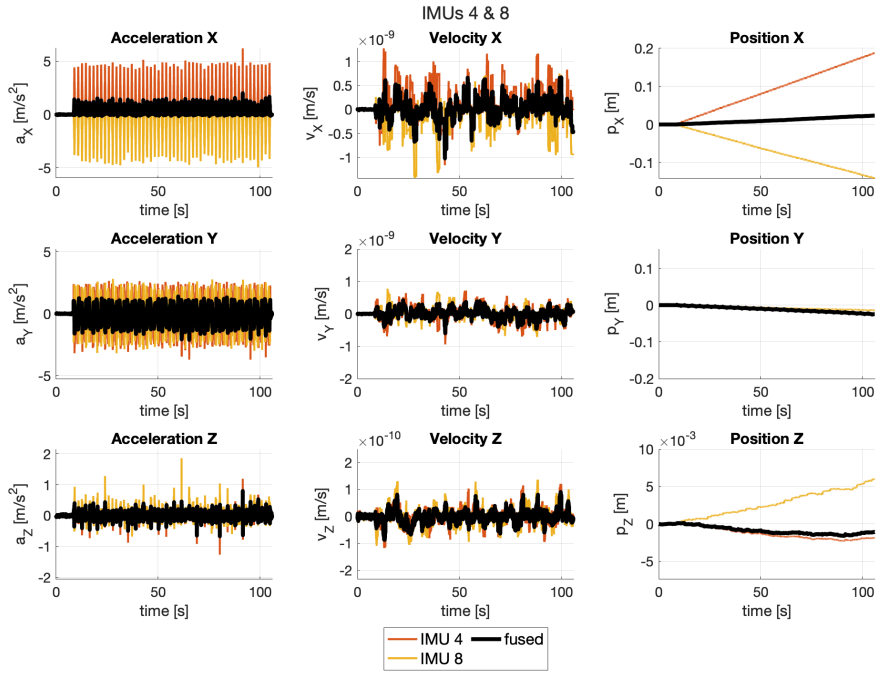


Figure 9.67.: Acceleration, velocity, and position, IMUs 4 & 8

Next, the IMUs are fused in fours, namely consider the group of IMUs 1, 3, 5, 7 and 2, 4, 6, 8.

The orientation data for IMUs 1, 3, 5, 7 are shown in Figure 9.68.

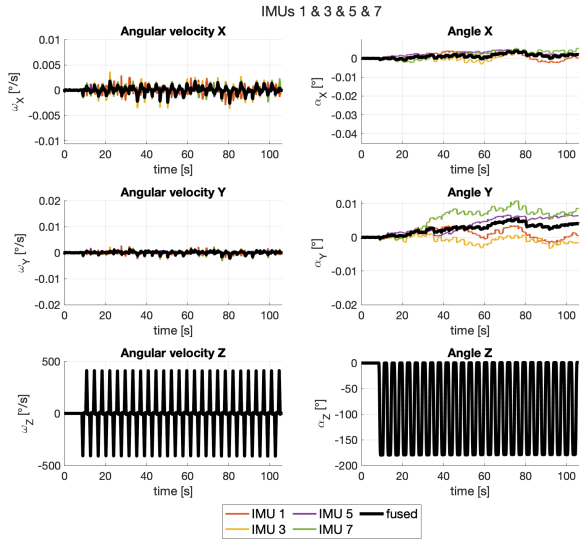


Figure 9.68.: Angular velocity and angle, IMUs 1&3&5&7

The position data for IMUs 1, 3, 5, 7 are shown in Figure 9.69.

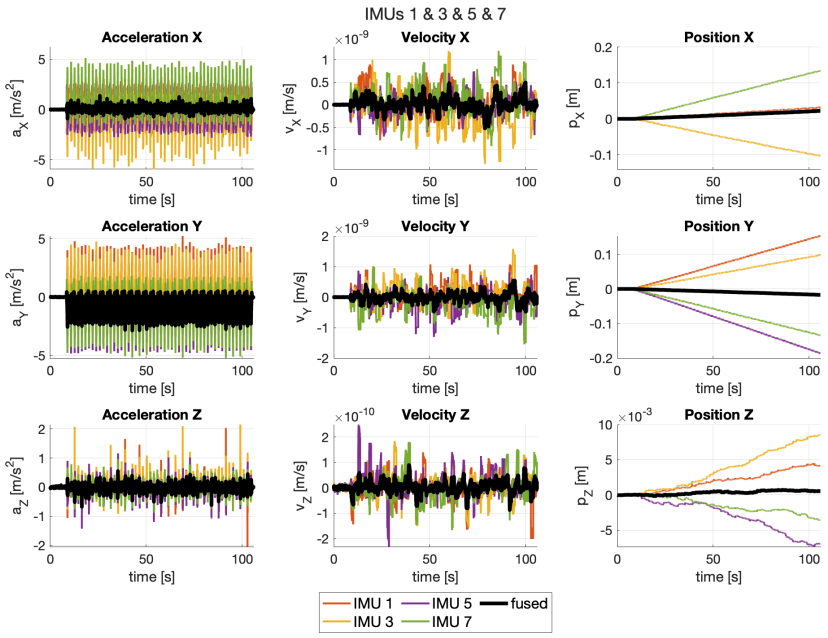


Figure 9.69.: Acceleration, velocity, and position, IMUs 1&3&5&7

The orientation data for IMUs 2, 4, 6, 8 are shown in Figure 9.70.

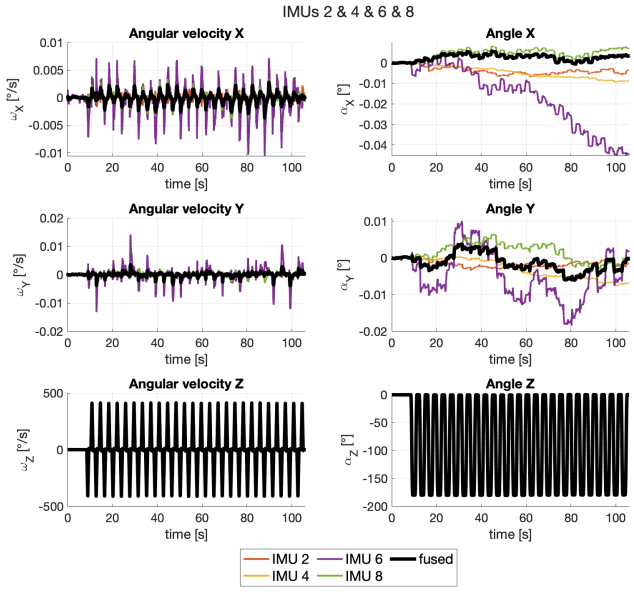


Figure 9.70.: Angular velocity and angle, IMUs 2&4&6&8

The position data for IMUs 2, 4, 6, 8 are shown in Figure 9.71.

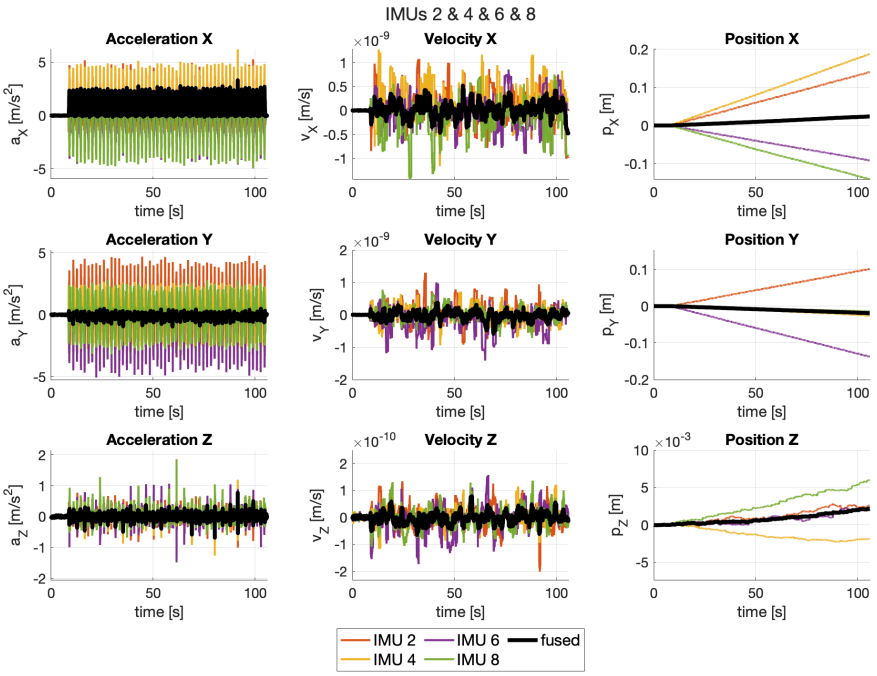


Figure 9.71.: Acceleration, velocity, and position, IMUs 2&4&6&8

Now, the fusion of all 8 IMUs is considered.
The orientation data for all 8 IMUs are shown in Figure 9.72.

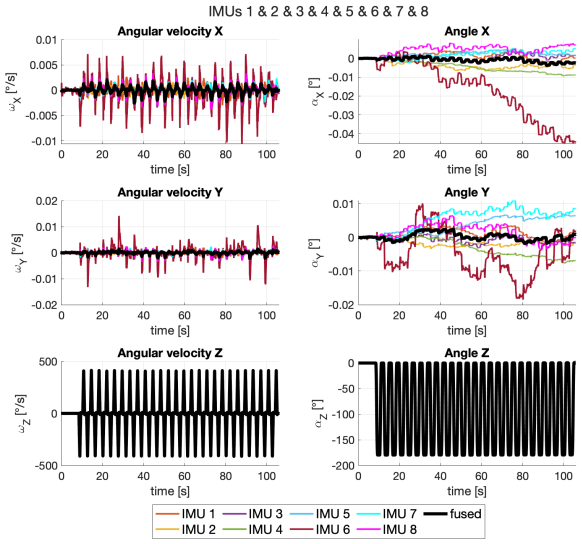


Figure 9.72.: Angular velocity and angle,
IMUs 1&2&3&4&5&6&7&8

The position data for all 8 IMUs are shown in Figure 9.73.

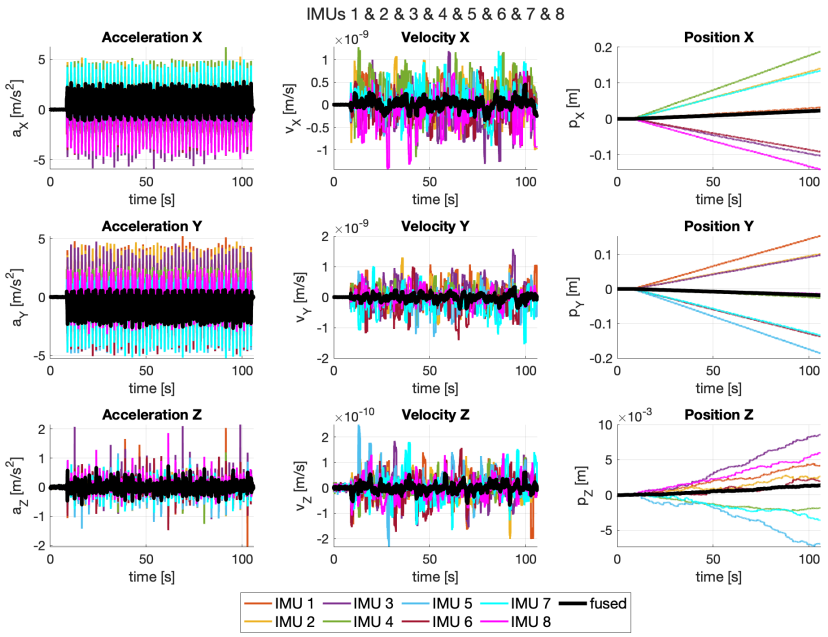


Figure 9.73.: Acceleration, velocity, and position,
IMUs 1&2&3&4&5&6&7&8

The presented experiment shows the advantages of fusing Multi-IMU data over the usage of single IMUs. The results are discussed further in Subsection 9.1.4.5.

9.1.4.4. Experiment: Translation

In this experiment, the data were collected from the Multi-IMU sensors while the tracked object was moved by hand. The tracking data obtained from the optical tracking system are presented in Figure 9.74.

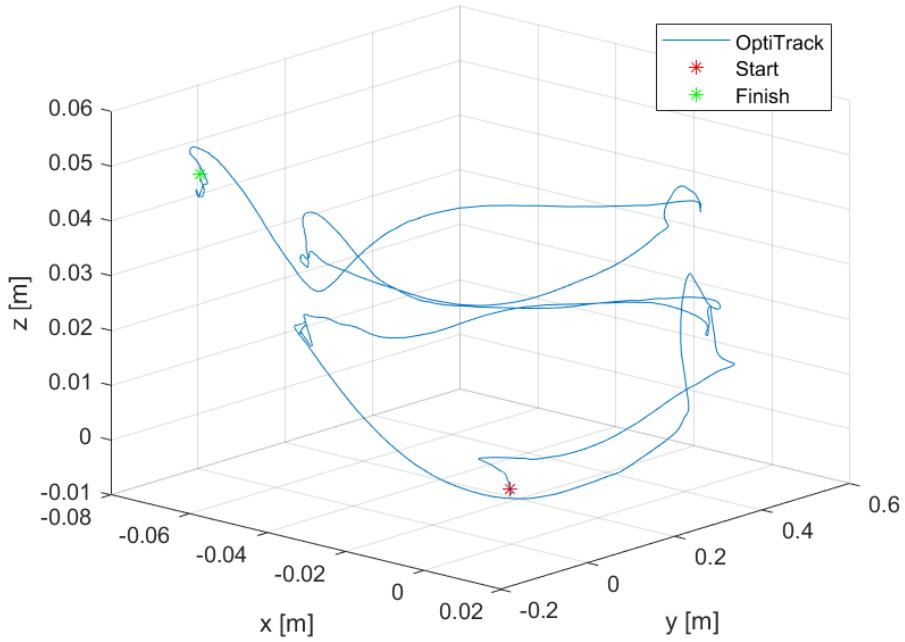


Figure 9.74.: Position tracked by the optical tracker

Because this experiment provides a more complex dataset, the data are additionally represented component-wise in Figure 9.75. As an interpretation of the hand movement, it can be said to resemble a boxing movement with three strokes being captured by the tracking systems.

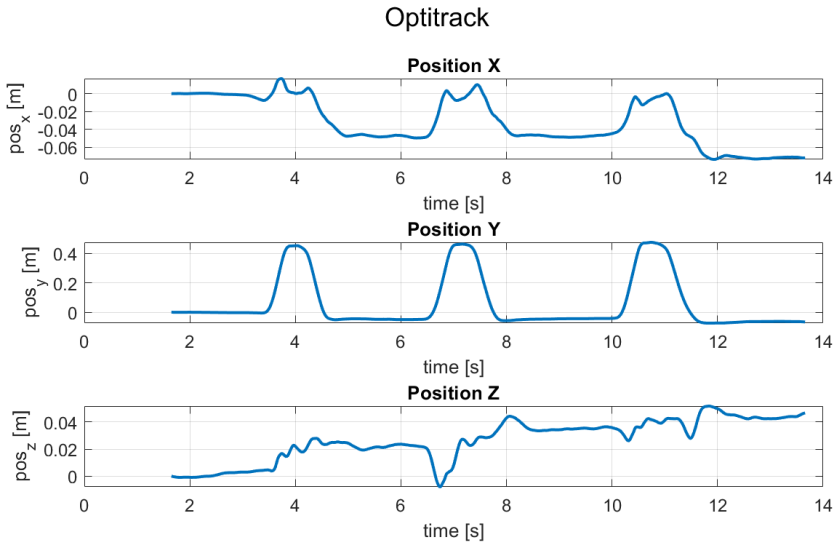


Figure 9.75.: Position tracked by the optical tracker (component-wise)

The results of the pose estimation are presented in the following series of figures. First, no fusion was performed, i.e., the data obtained from a single IMU is presented. Measured enhanced values are presented together with filtered values (for angular velocity and acceleration) along with angles, velocities, and positions, obtained by integration of the enhanced data. Note, that in this case gravity removal is not trivial and it was decided to present the measured acceleration as it is together with the enhanced acceleration data, i.e., with the acceleration data that were filtered, registered, and with the gravity removed.

The orientation data for IMU 1 are shown in Figure 9.76.

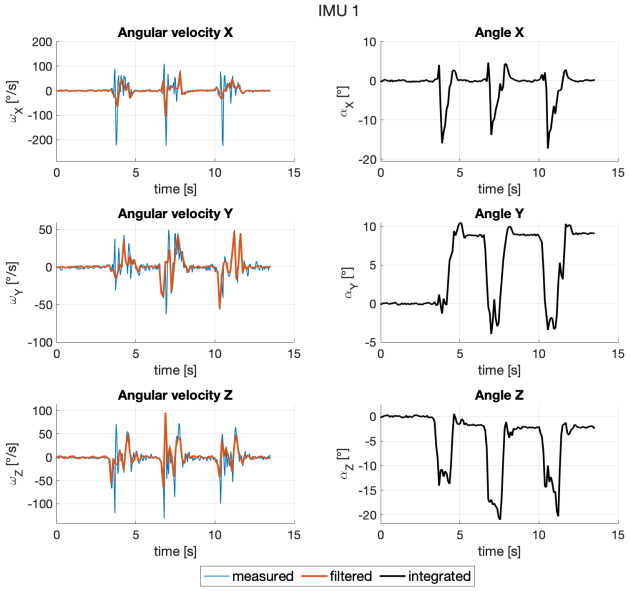


Figure 9.76.: Angular velocity and angle, IMU 1

The position data for IMU 1 are shown in Figure 9.77.

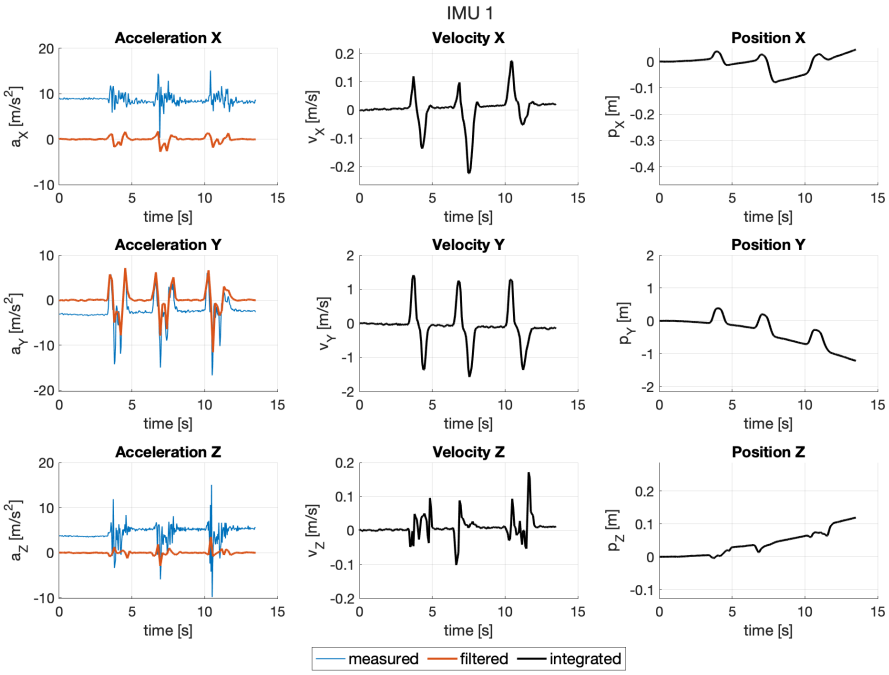


Figure 9.77.: Acceleration, velocity, and position, IMU 1

The orientation data for IMU 2 are shown in Figure 9.78.

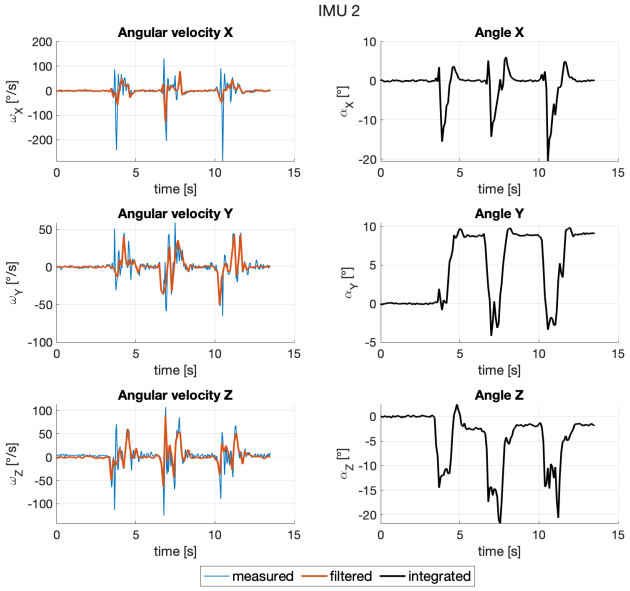


Figure 9.78.: Angular velocity and angle, IMU 2

The position data for IMU 2 are shown in Figure 9.79.

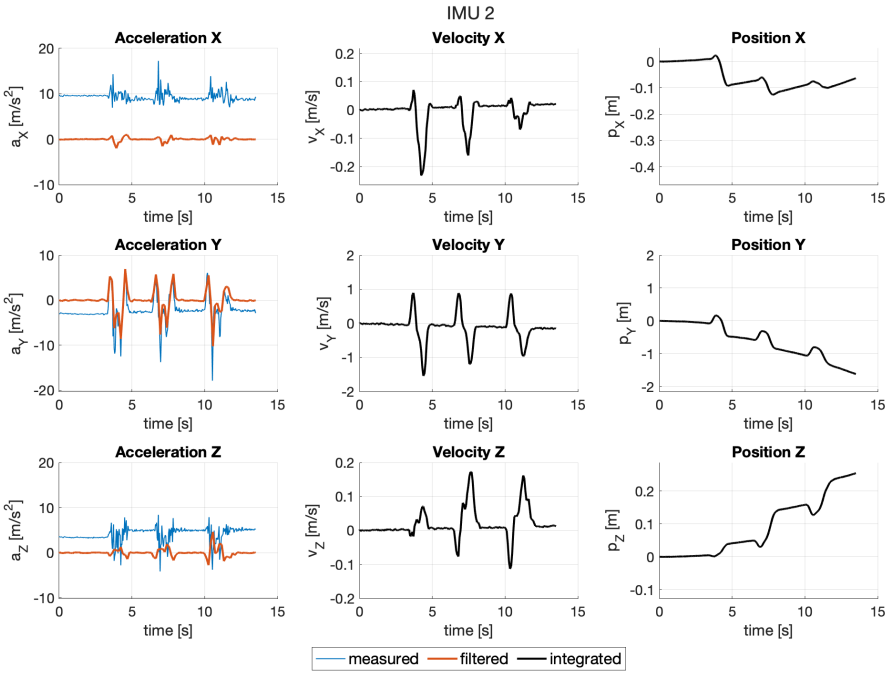


Figure 9.79.: Acceleration, velocity, and position, IMU 2

The orientation data for IMU 3 are shown in Figure 9.80.

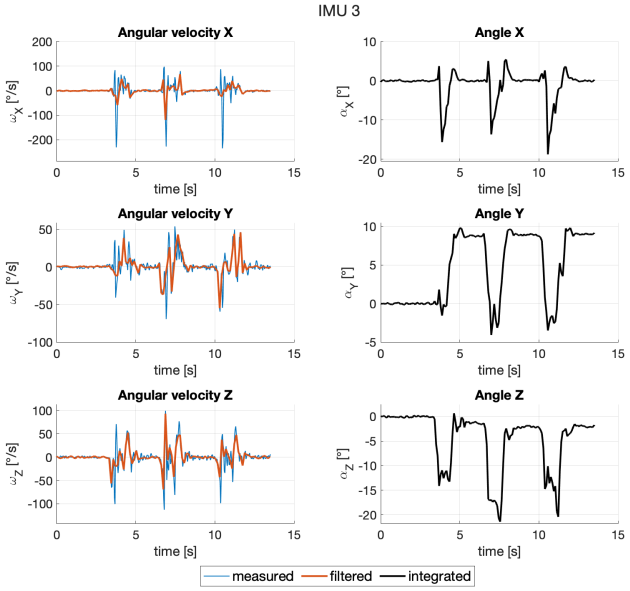


Figure 9.80.: Angular velocity and angle, IMU 3

The position data for IMU 3 are shown in Figure 9.81.

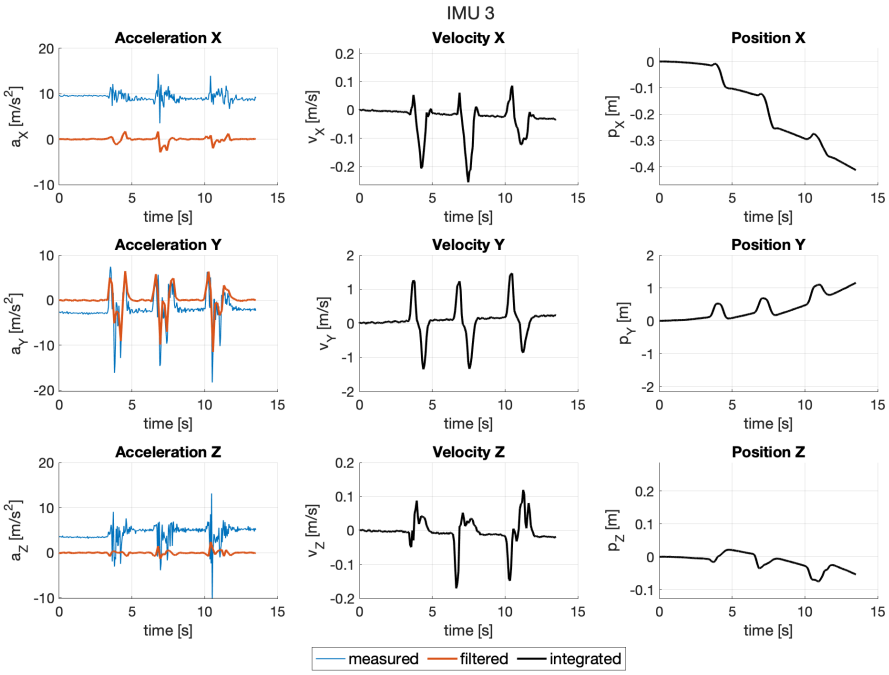


Figure 9.81.: Acceleration, velocity, and position, IMU 3

The orientation data for IMU 4 are shown in Figure 9.82.

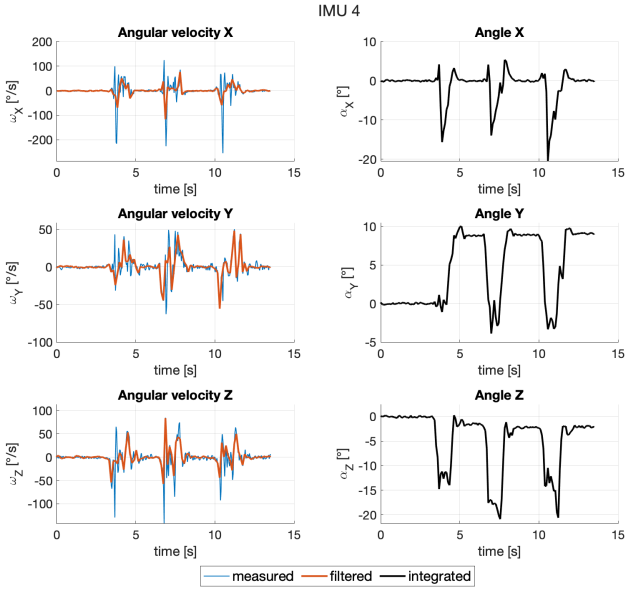


Figure 9.82.: Angular velocity and angle, IMU 4

The position data for IMU 4 are shown in Figure 9.83.

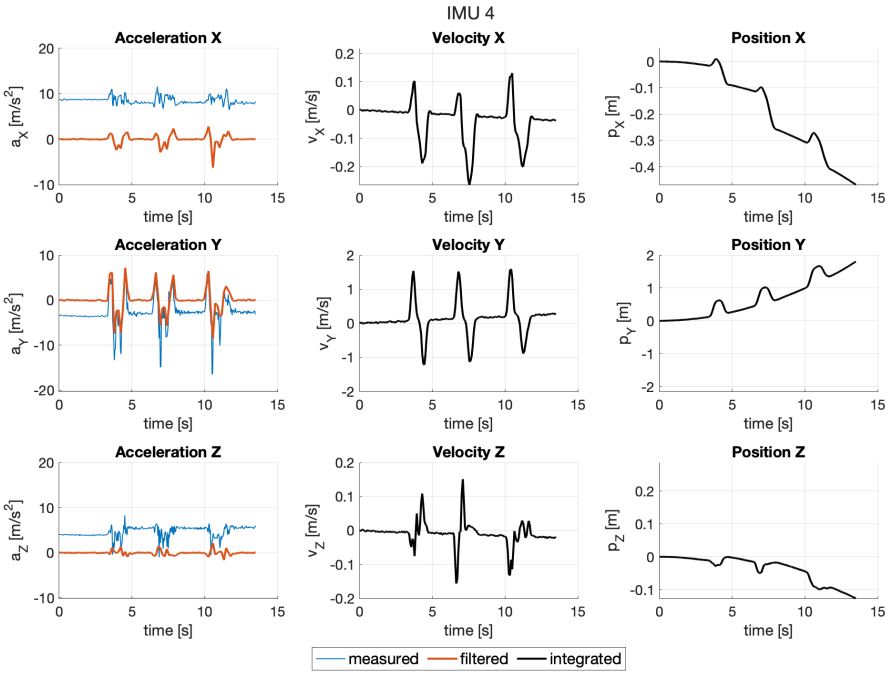


Figure 9.83.: Acceleration, velocity, and position, IMU 4

The orientation data for IMU 5 are shown in Figure 9.84.

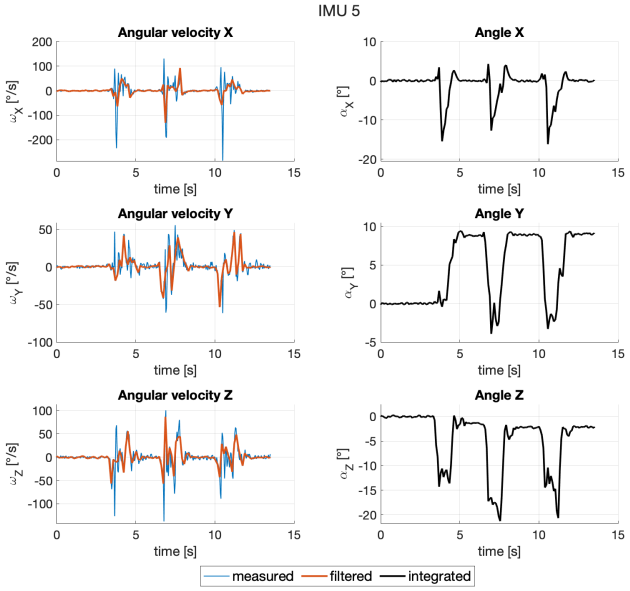


Figure 9.84.: Angular velocity and angle, IMU 5

The position data for IMU 5 are shown in Figure 9.85.

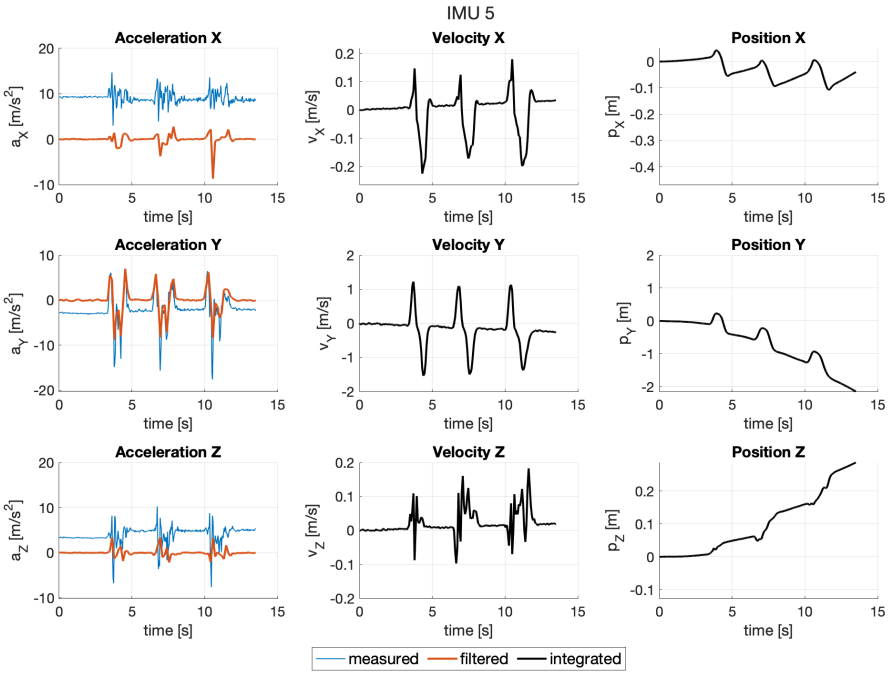


Figure 9.85.: Acceleration, velocity, and position, IMU 5

The orientation data for IMU 6 are shown in Figure 9.86.

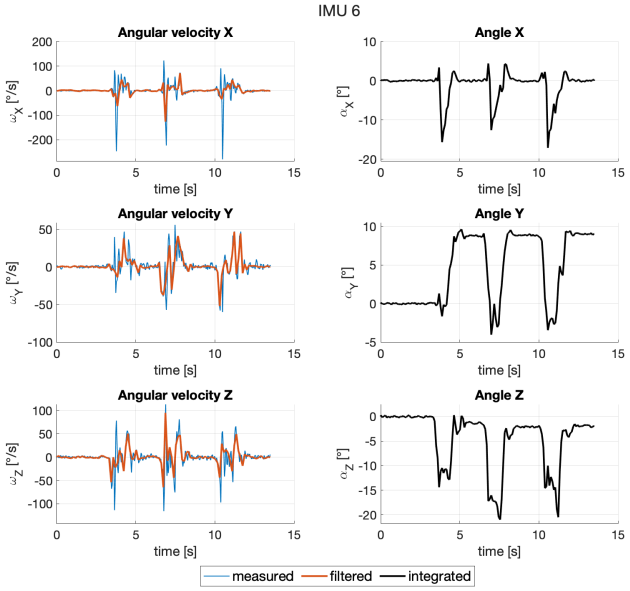


Figure 9.86.: Angular velocity and angle, IMU 6

The position data for IMU 6 are shown in Figure 9.87.

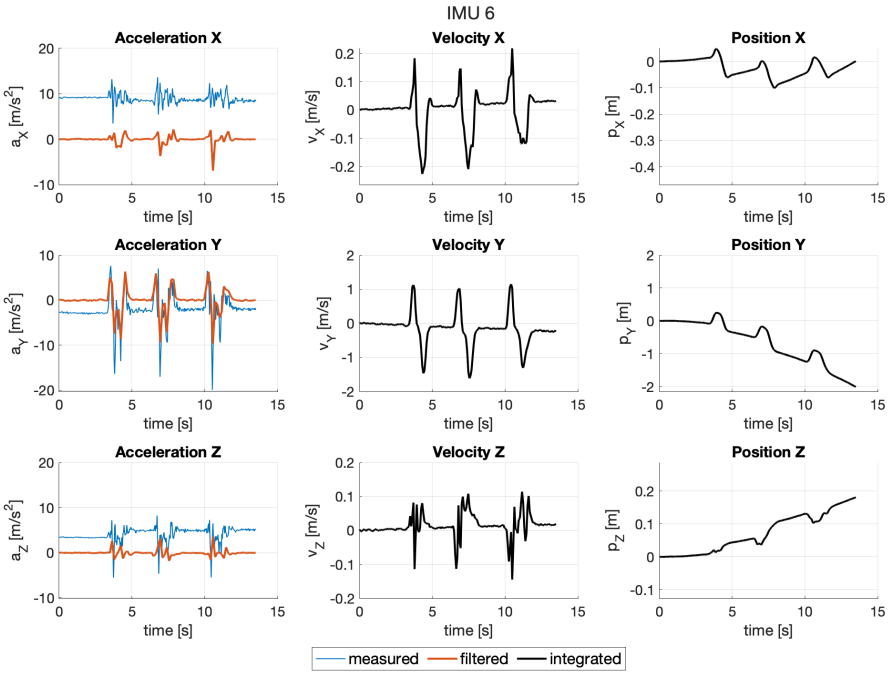


Figure 9.87.: Acceleration, velocity, and position, IMU 6

The orientation data for IMU 7 are shown in Figure 9.88.

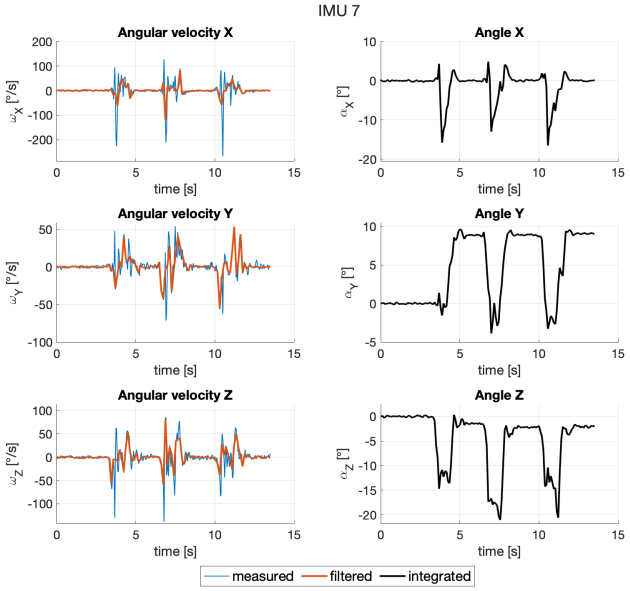


Figure 9.88.: Angular velocity and angle, IMU 7

The position data for IMU 7 are shown in Figure 9.89.

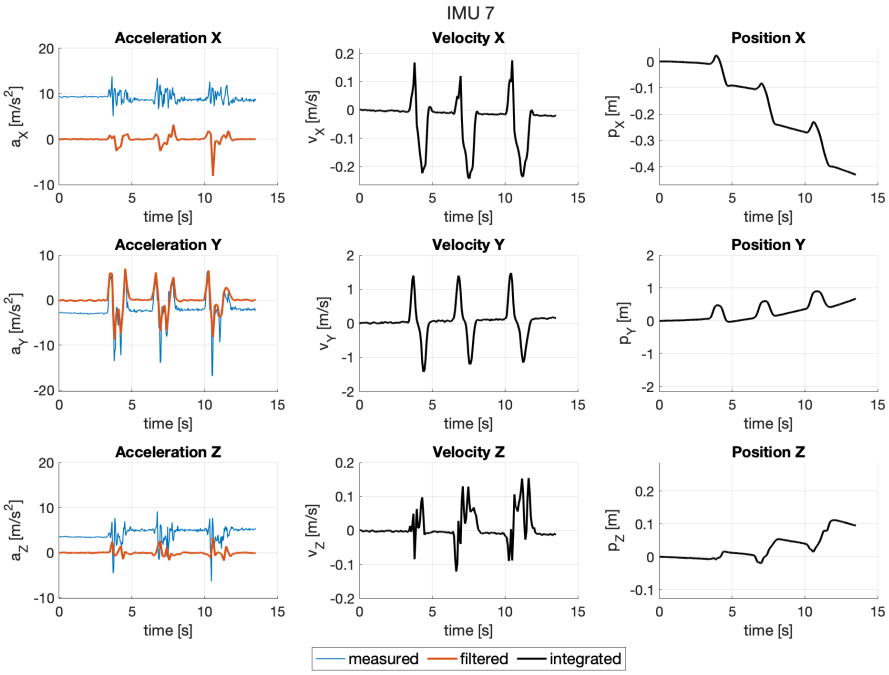


Figure 9.89.: Acceleration, velocity, and position, IMU 7

The orientation data for IMU 8 are shown in Figure 9.90.

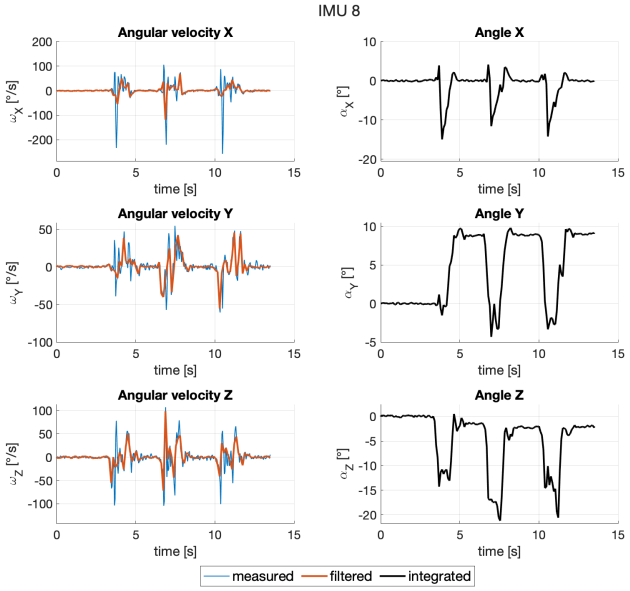


Figure 9.90.: Angular velocity and angle, IMU 8

The position data for IMU 8 are shown in Figure 9.91.

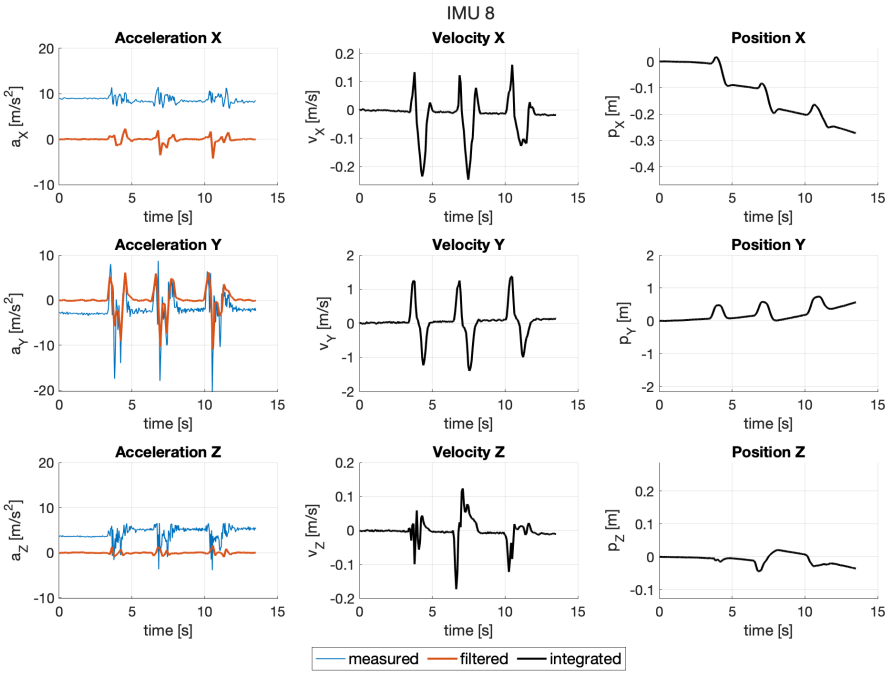


Figure 9.91.: Acceleration, velocity, and position, IMU 8

Next, the data is fused for the couples of paired IMUs: 1 and 5, 2 and 6, 3 and 7, and 4 and 8. These pairs of IMUs are chosen as they are located across from each other in the physical setting, meaning that with the implemented registration method the noise can be further reduced by fusion of the chosen IMUs.

The orientation data for IMUs 1 and 5 are shown in Figure 9.92.

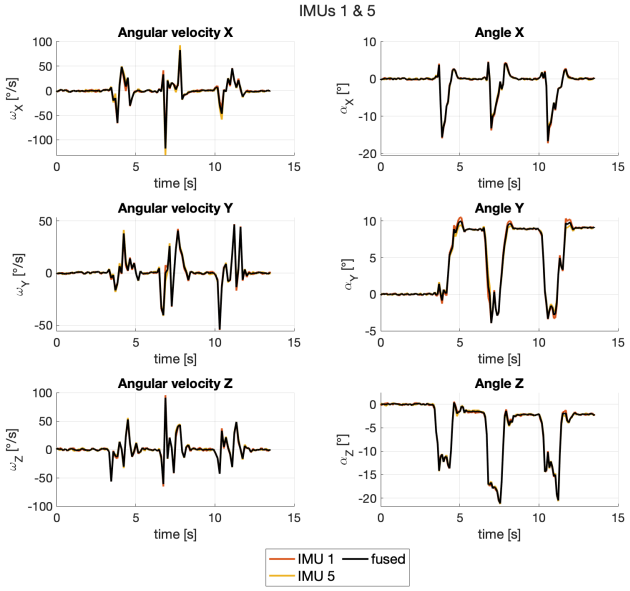


Figure 9.92.: Angular velocity and angle, IMUs 1 & 5

The position data for IMUs 1 and 5 are shown in Figure 9.93.

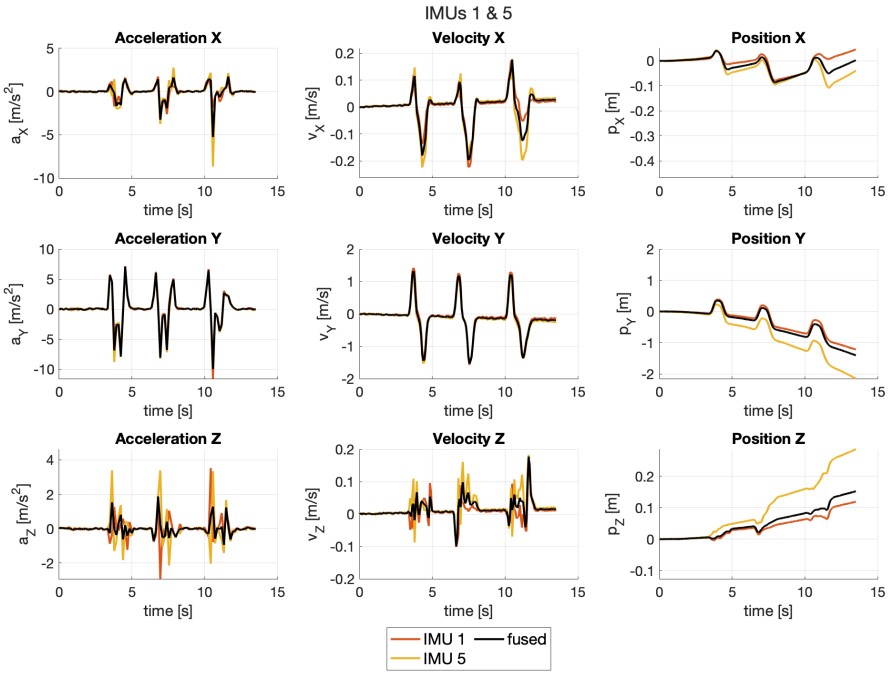


Figure 9.93.: Acceleration, velocity, and position, IMUs 1 & 5

The orientation data for IMUs 2 and 6 are shown in Figure 9.94.

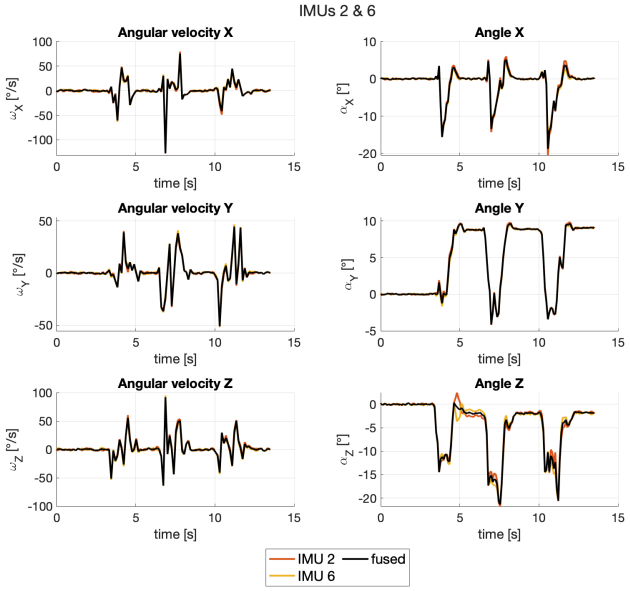


Figure 9.94.: Angular velocity and angle, IMUs 2 & 6

The position data for IMUs 2 and 6 are shown in Figure 9.95.

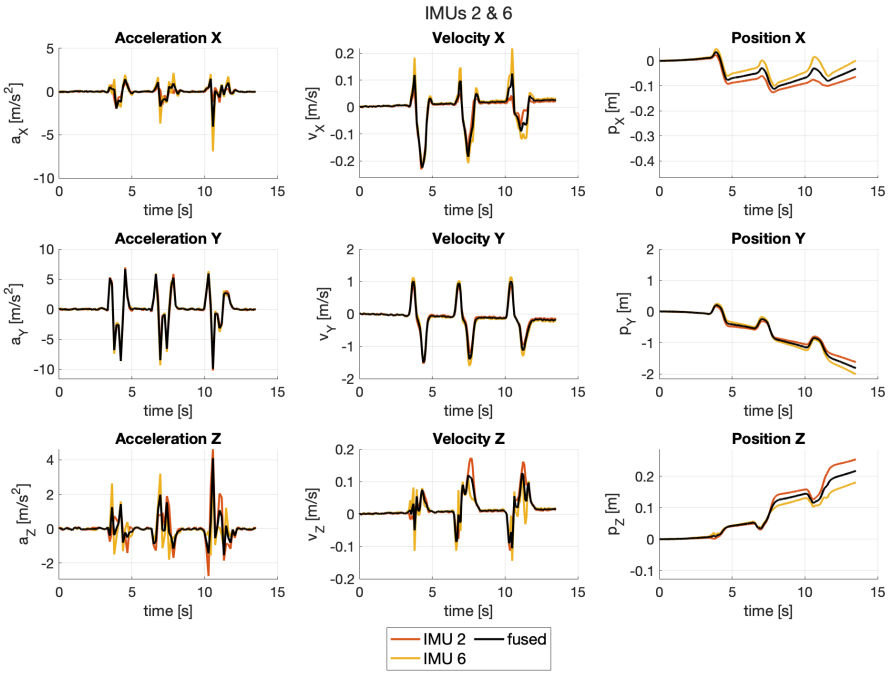


Figure 9.95.: Acceleration, velocity, and position, IMUs 2 & 6

The orientation data for IMUs 3 and 7 are shown in Figure 9.96.

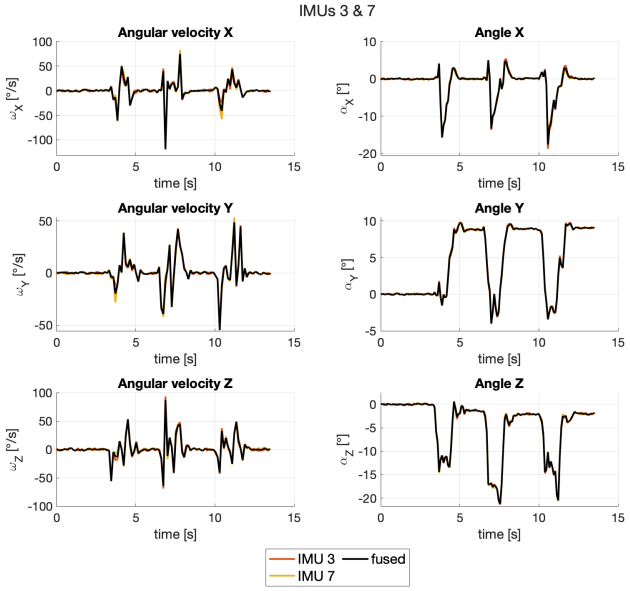


Figure 9.96.: Angular velocity and angle, IMUs 3 & 7

The position data for IMUs 3 and 7 are shown in Figure 9.97.

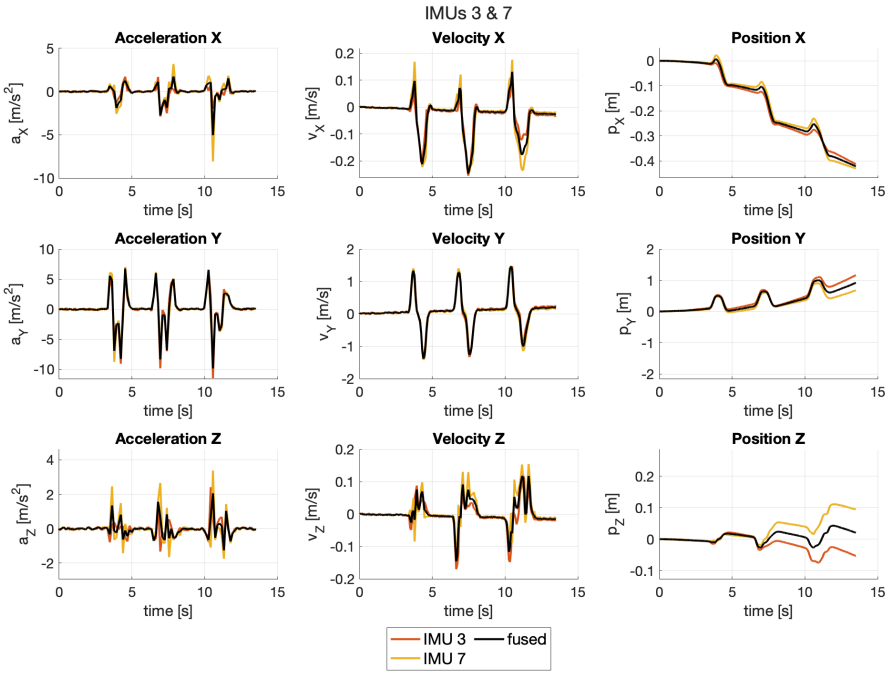


Figure 9.97.: Acceleration, velocity, and position, IMUs 3 & 7

The orientation data for IMUs 4 and 8 are shown in Figure 9.98.

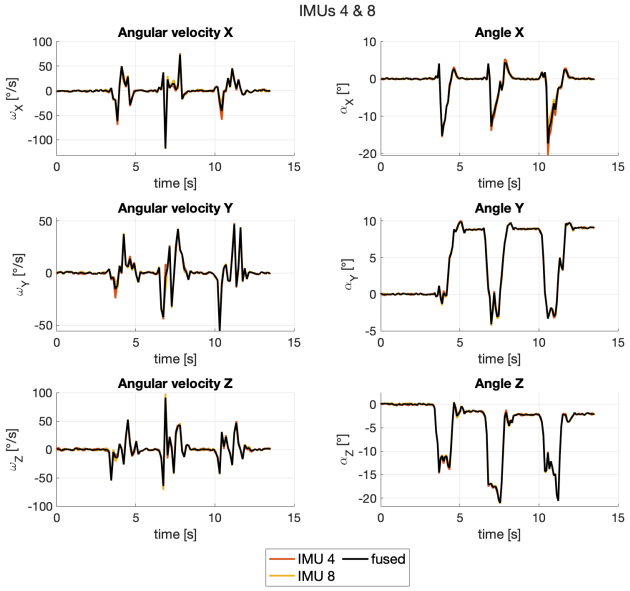


Figure 9.98.: Angular velocity and angle, IMUs 4 & 8

The position data for IMUs 4 and 8 are shown in Figure 9.99.

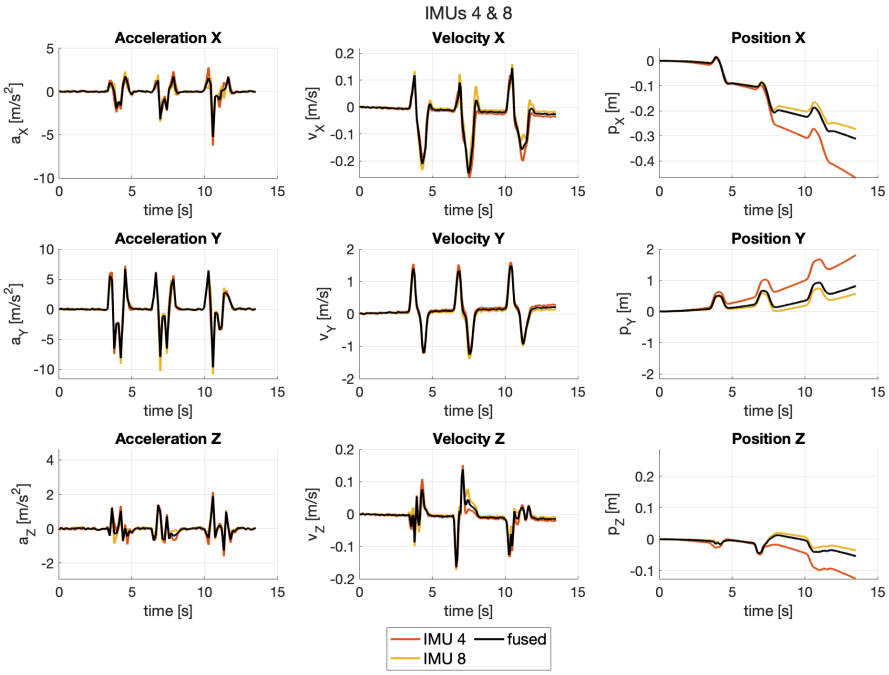


Figure 9.99.: Acceleration, velocity, and position, IMUs 4 & 8

Next, the IMUs are fused in fours, namely consider the group of IMUs 1, 3, 5, 7 and 2, 4, 6, 8.

The orientation data for IMUs 1, 3, 5, 7 are shown in Figure 9.100.

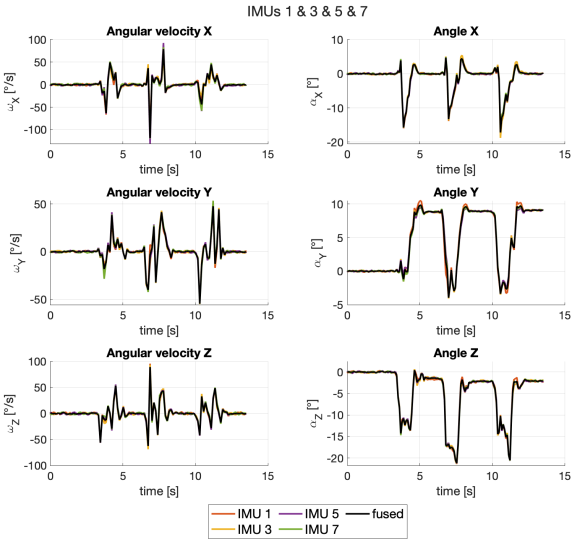


Figure 9.100.: Angular velocity and angle, IMUs 1&3&5&7

The position data for IMUs 1, 3, 5, 7 are shown in Figure 9.101.

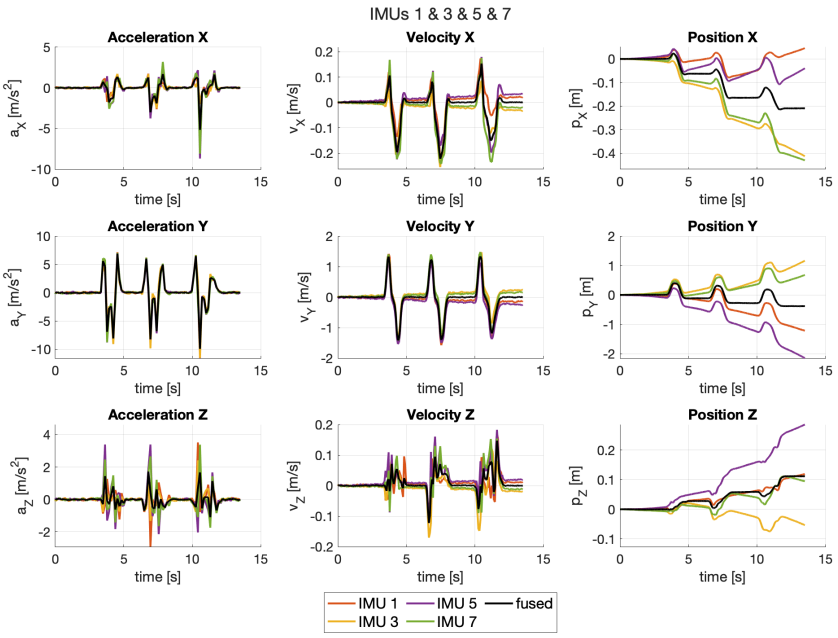


Figure 9.101.: Acceleration, velocity, and position, IMUs 1&3&5&7

The orientation data for IMUs 2, 4, 6, 8 are shown in Figure 9.102.

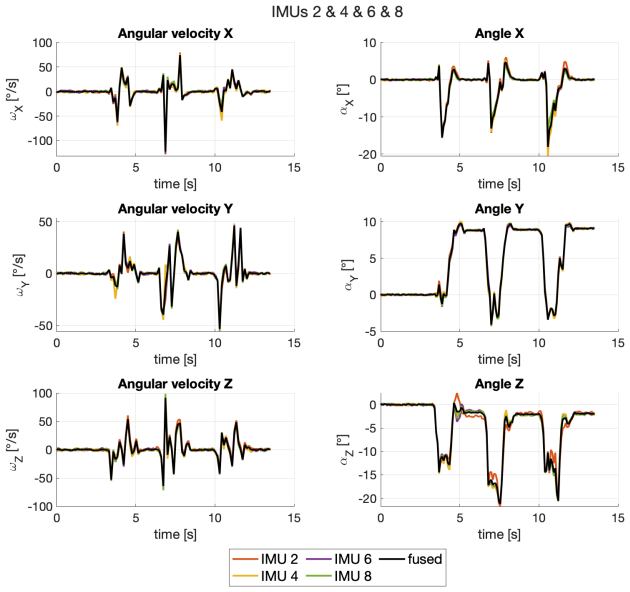


Figure 9.102.: Angular velocity and angle, IMUs 2&4&6&8

The position data for IMUs 2, 4, 6, 8 are shown in Figure 9.103.

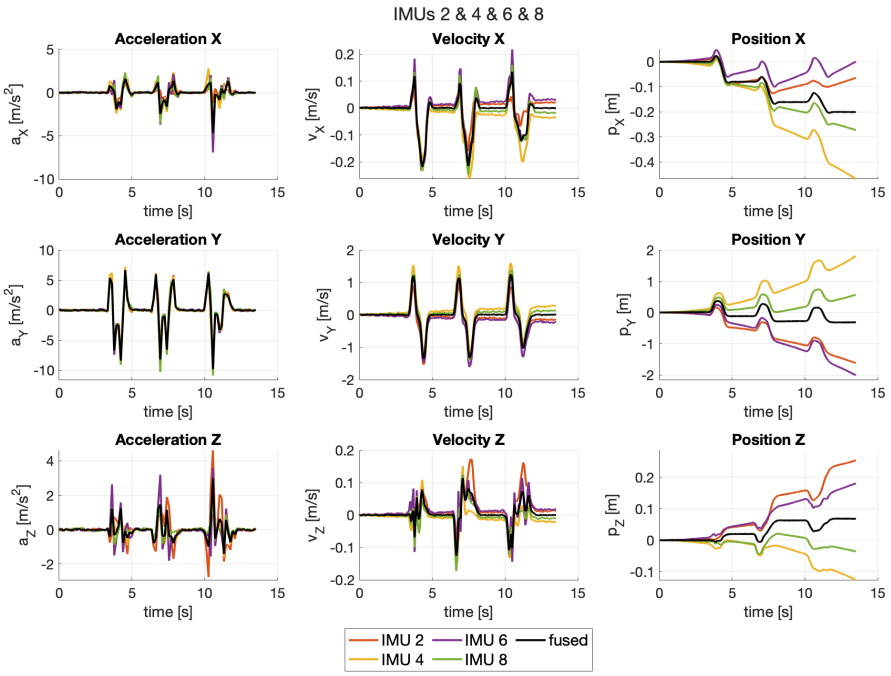


Figure 9.103.: Acceleration, velocity, and position, IMUs 2&4&6&8

Now, the fusion of all 8 IMUs is considered.
The orientation data for all 8 IMUs are shown in Figure 9.104.

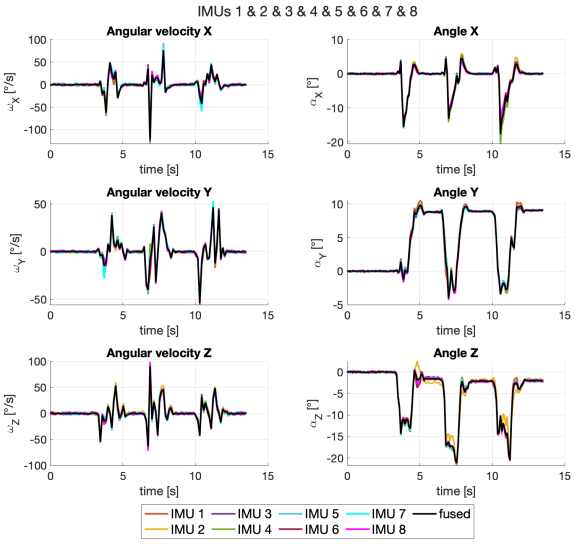


Figure 9.104.: Angular velocity and angle,
IMUs 1&2&3&4&5&6&7&8

The position data for all 8 IMUs are shown in Figure 9.105.

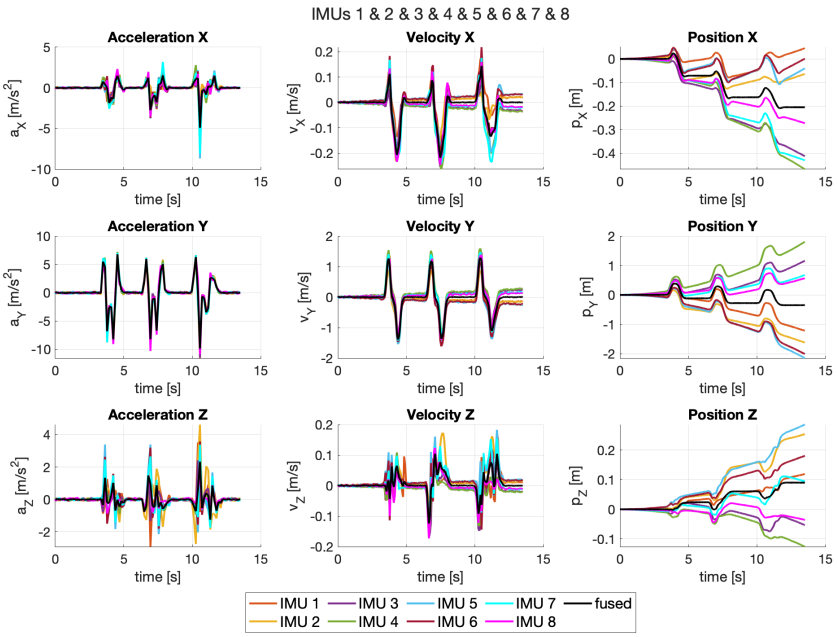


Figure 9.105.: Acceleration, velocity, and position,
IMUs 1&2&3&4&5&6&7&8

The presented experiment shows the advantages of fusing Multi-IMU data over the usage of single IMUs. The results are discussed further in Subsection 9.1.4.5.

9.1.4.5. Discussion

The proposed approach with a specifically designed physical setup and a modified pose estimation pipeline was tested in a series of three experiments. In this subsection, some important observations will be made based on the data presented earlier in the section, and some conclusions will be drawn from them.

First of all, consider the physical setup itself. The presented circular placement of the IMUs is advantageous from two very different points of view. On the one hand, it allows for easier data fusion (this will be discussed in more detail next), and on the other hand, it does not have a sensor (or any other physical element) placed in the center, meaning, that an application that requires an estimation of the pose in a point unavailable for direct placement of a sensor can use the proposed placement scheme and still get reasonable pose estimation in the unavailable location.

As can be seen by comparing the results of pose estimation using a single IMU with the results obtained by fusing Multi-IMU, the use of fused data helps reduce the noise and drift in the data. Furthermore, the fusion of 4 IMUs generally gives better results than the fusion of couples, and the results obtained by fusion of all 8 IMUs are better yet.

Consider, for example, the following comparison performed on the data obtained in the rotation experiment. The second norm of the position vectors obtained with the fusion of couples, fourths, and all eight IMUs is plotted in Figure 9.106.

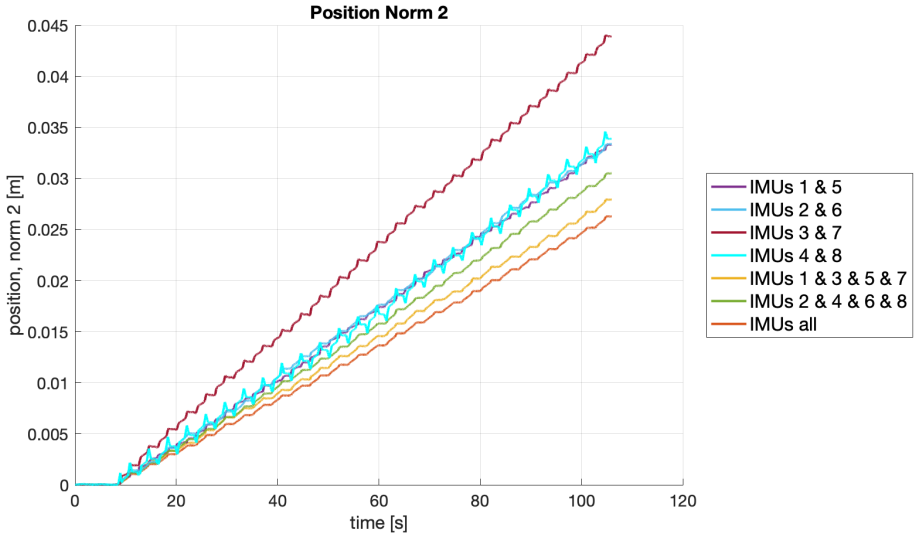


Figure 9.106.: Comparison of fusion patterns, $||p||_2$ [m]

Since in the rotation experiment the position doesn't change, the ground truth for this

case is a static position, i.e., all components of position stay 0 throughout the experiment. However, because of noise and drift, the estimated position changes with time. However, it can be seen from the presented plot that fusing 4 IMUs gives less deviation, and fusing 8 IMUs gives yet less deviation.

It should be noted that the data presented in the corresponding experiment sections are only fused according to selected patterns. The explanation for that is already given in the setup description (shortly, fusing counter-located sensors' data should allow for better noise and drift reduction); however, it can be of interest to compare the obtained results with the results of the fusion of different patterns.

Figure 9.107 shows the same metric (namely, the second norm of the position vector) for the rotation experiment for the results obtained by fusing IMU 1 consecutively with IMUs 2, 3, 4, 5, 6, 7, and 8.

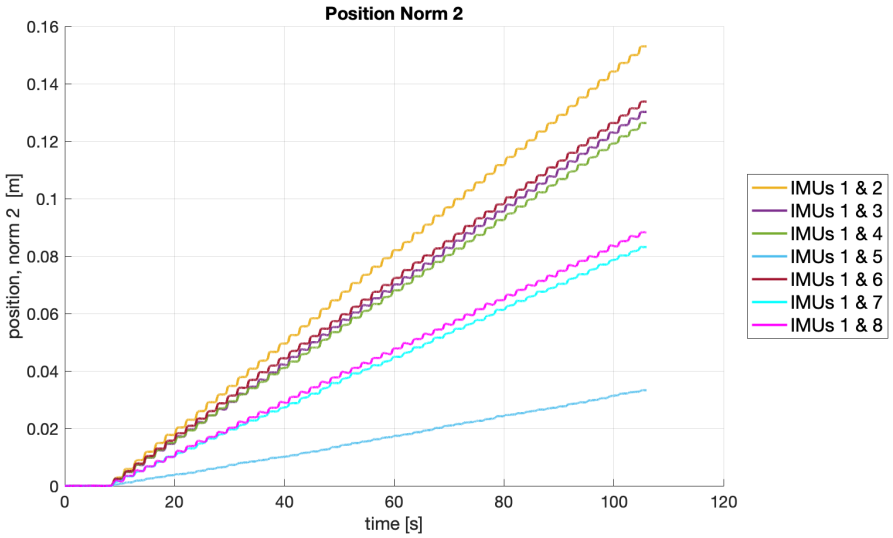


Figure 9.107.: Comparison: IMU 1 fused consecutively with other IMUs, $||p||_2$ [m]

As can be verified in Figure 9.5, IMU 1 is located across IMU 5, which corresponds to the best curve in Figure 9.107. The winning pair 1&5 is clear from the plot, and the other curves show more affection from drift and noise than the leading one.

A logical next comparison is for the fusion of different fours of IMUs. As shown in the figure above, fusing not-coupled IMUs leads to a decrease in accuracy, so only the fours consisting of coupled IMUs are considered. Figure 9.108 shows the results of this comparison.

As seen from the plot, all of the results are roughly in the same accuracy ballpark, meaning that any combination of 2 couples of IMUs can be successfully fused.

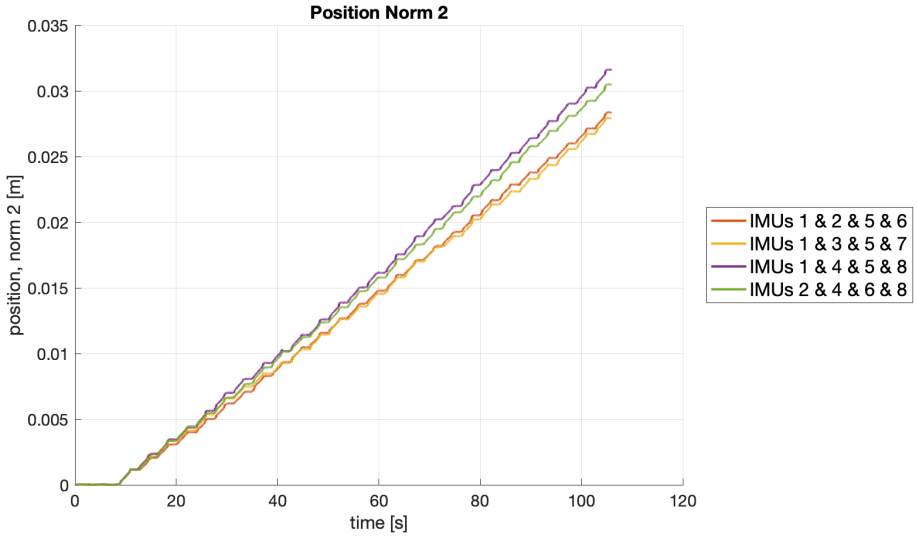


Figure 9.108.: Comparison: different fours of IMUs, $||p||_2$ [m]

Next, the quality of pose estimation should finally be assessed. For the static experiment the quality estimation can be easily viewed from the presented plots: in the ground truth, all output values should be 0 and any deviation clearly represents an error. Naturally, by studying the plots, it can be observed that they are not completely free from the noise, but the amplitudes of deviations are acceptably small.

Table 9.1 presents the worst cases (by IMUs considered) of the infinite norm of absolute error of estimated values for the static case.

The values presented in the table above allow concluding that for the static case, the proposed pipeline and the setup work fine; the table data also show quantitatively the reduction of the error when more IMUs are strategically fused.

Table 9.1.: Absolute error, infinite norm, worst case

Quantity	Meas. unit	Single IMU	2 IMUs	4 IMUs	8 IMUs
ω_x	$[°/s]$	0.0509	0.0430	0.0339	0.0258
ω_y	$[°/s]$	0.0381	0.0312	0.0165	0.0137
ω_z	$[°/s]$	0.0339	0.0244	0.0143	0.0120
α_x	$[°]$	3.3639	0.4546	0.2203	0.1983
α_y	$[°]$	0.7381	0.4583	0.1451	0.0572
α_z	$[°]$	1.9524	1.2164	0.2323	0.0845
a_x	$[m/s^2]$	0.0133	0.0074	0.0036	0.0034
a_y	$[m/s^2]$	0.0073	0.0056	0.0037	0.0036
a_z	$[m/s^2]$	0.0144	0.0126	0.0082	0.0081
v_x	$[m/s]$	2.72e-10	1.42e-10	9.02e-11	3.59e-11
v_y	$[m/s]$	1.46e-10	7.27e-11	3.53e-11	2.61e-11
v_z	$[m/s]$	1.64e-10	8.24e-11	6.71e-11	4.32e-11
p_x	$[m]$	7.98e-4	3.62e-4	9.80e-5	1.40e-5
p_y	$[m]$	8.88e-4	3.31e-4	1.19e-4	7.76e-5
p_z	$[m]$	1.21e-3	7.37e-4	1.87e-4	7.87e-5

For the rotation experiment, the ground truth data are not available for some of the estimated quantities. The optical tracker only collects the data for the designated point, and in this case, the orientation data can be easily interpreted for the considered scheme, but the position data are not comparable. The reason is the specific accelerometer registration procedure that basically gives position data in the center of the plate, which (as it corresponds to the application point of rotation in the orthogonal axis) is static. In the experiment, as stated in the respective section, the sensors are also getting influenced by the vibration of the engine. However, the angles can still easily be compared with the ground truth. Another meaningful comparison would be for the velocity and position data, as the presented results are believed to show the position of the center of the plate which should be static. The velocity is basically zero for all of the presented cases, but the positions can be compared. Again, this comparison is only meaningful for the fused data (because of the registration strategy employed). Table 9.2 shows the errors for the specified quantities.

Table 9.2.: Absolute error, infinite norm, worst case

Quantity	Meas. unit	Single IMU	2 IMUs	4 IMUs	8 IMUs
α_x	$[°]$	0.0491	0.0128	0.0119	0.0042
α_y	$[°]$	0.0192	0.0057	0.0041	0.0023
α_z	$[°]$	2.1342	1.0391	0.8812	0.1944
p_x	$[m]$	-	3.19e-2	2.71e-2	2.29e-2
p_y	$[m]$	-	4.91e-2	1.23e-2	1.15e-2
p_z	$[m]$	-	2.80e-3	2.71e-3	1.71e-3

Again, the table shows that the drift is reduced when the data are fused from several sensors and confirm the applicability of the proposed method for pose estimation.

The translation (hand movement) experiment faces the same problem as the rotation experiment: no proper ground truth data is available for some of the estimated quantities. In this case, the position data is available from the optical tracker, but it can clearly be seen that even the best available results (in the presented data, the fusion of all 8 IMUs gives the best result) are not corresponding exactly to the optical tracking data presented in Figure 9.75. However, consider a visual comparison of the position components obtained from the optical tracker and from the 8 fused IMUs, presented in Figure 9.109.

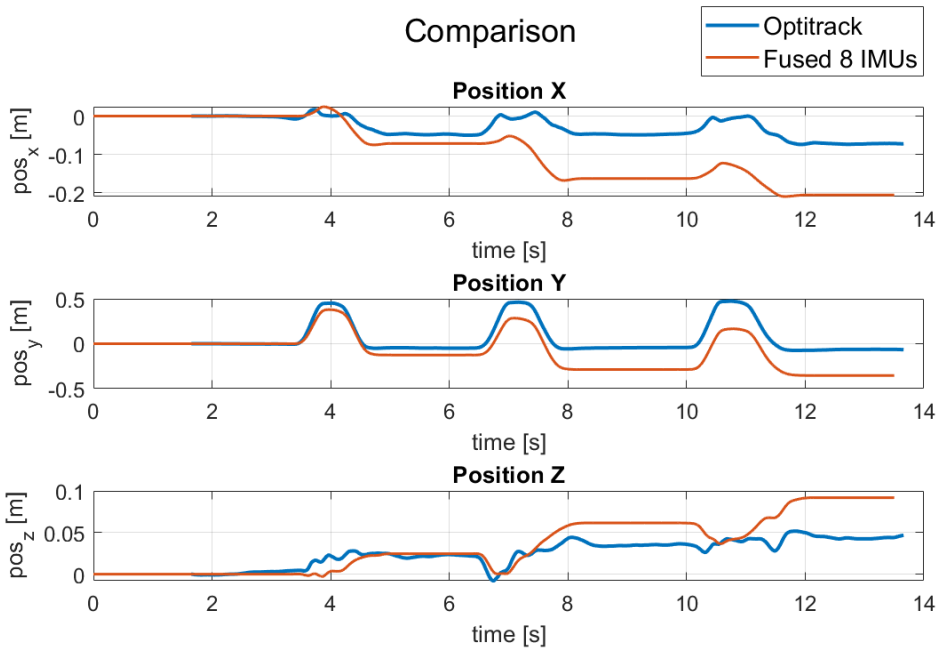


Figure 9.109.: Comparison with optical tracker: position components

The data obtained with fusion show clear correspondence to the optical tracking data: the locations of the features (peaks, bumps, valleys, etc.), the general structure, and the amplitudes are similar if not exactly coinciding. The data obtained via Multi-IMUs clearly exhibit drift which is the expected behavior of inertial sensors. Nevertheless, the presented comparison shows the potential in the use of Multi-IMUs.

As expected, the orientation can be estimated rather well with the use of IMUs, and the fusion of Multi-IMUs enhances the data; the position data can also be estimated, but the quality is somewhat lacking. However, fusion still helps reduce some if not all of the drift and noise.

For that reason, pattern recognition strategy should be employed on the design stage of the Multi-IMU based device, meaning, that Multi-IMUs can be supported with either specific algorithms (as ZUPT, for example, for applications that can use rotation-over-translation

simplification) or extra sensors. As for the latter, it could be a magnetometer, a GPS navigator, a wideband-based indoor tracking system, a camera, etc. depending on the particular applications that the developer has in mind.

Summing up, the following conclusions can be drawn from the discussed observations:

- The proposed physical setup, which does not require a tracker to be placed in the center of the setup, is suitable for a few real-life applications, such as head-mounted displays, wrist or ankle bracelets, or any other applications with similar spatial restrictions.
- Fusing *specifically* selected multiple IMUs reduces drift and noise. Moreover, fusing 4 IMUs gives generally better results than fusing 2 IMUs, and fusing 8 IMUs gives better results than fusing 4 IMUs.
- Fusing *randomly* selected IMUs does not have the same noise- and drift-reduction effect.
- The proposed adapted pipeline allows for pose estimation with Multi-IMU.
- Orientation estimation is reasonable with a single IMU and can be further enhanced by fusion. Position estimation with fused IMUs is reasonable but requires further enhancement, especially in cases when long periods of tracking are needed.
- Pattern recognition strategy is advised when constructing a Multi-IMU-based tracking device: different applications require different enhancement strategies for position estimation.

All in all, the presented approach together with a specific physical setup allow estimating the pose of the object by following the steps of the proposed pipeline and fusing the data from Multi-IMUs. The obtained results show rather good correspondence to the ground truth values. The use of zero-velocity update (or other enhancement techniques based on additional knowledge about the intended application of the system) allows reducing the usually present drift in accelerometer data, which is then further reduced by the use of fusion from multiple IMUs.

9.2. Pose Estimation with Machine Learning

In this part of the study, the focus is on choosing a suitable artificial neural network algorithm for pose estimation with Multi-IMU, in terms of accuracy and performance, by using simulated Multi-IMU sensor data. The project was initiated by the author and performed in cooperation with A. Rahman [226] and Z. Khan [144].

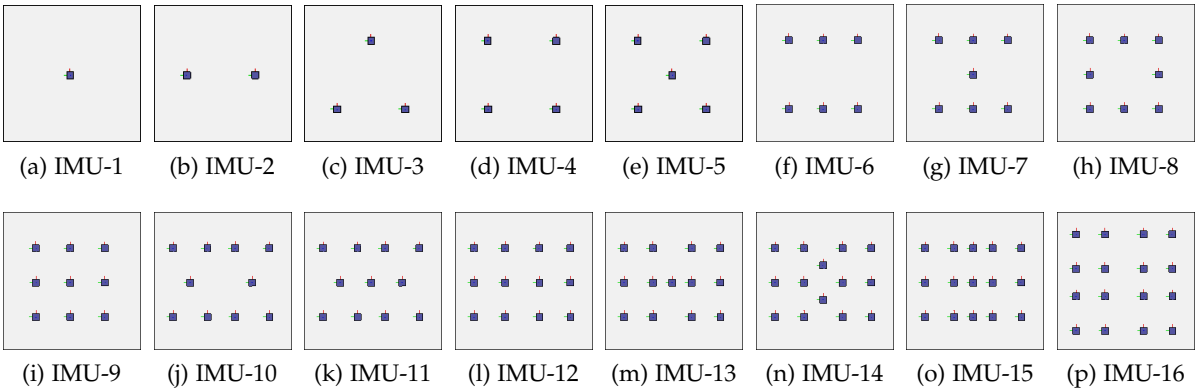
Generally, the major problems of Multi-IMU development are associated with the time and the cost of construction. To make predictions of an object's pose in a Mixed Reality scenario, a large number of experiments is usually needed in order to get as close as possible to the actual (ground truth) values. To test multiple potential configurations, the device has to be reconstructed for each configuration, then for each of them the data needs to be

collected multiple times, with all the underlying data enhancement and corrections performed anew each time (to minimize the errors present in real sensor data). The latter, at least for experimental reasons (e.g., obtaining the ground truth), requires a presence of an additional tracking system, usually an optical one. Again, using the additional tracker (that also generally needs enhancement) for Multi-IMU evaluation increases the time needed to perform and register the setup. Therefore, the data collection steps take up most of the time of the experiment, and still, the obtained results depend deeply on the quality of enhancement, thus diminishing the experiment value the goal of which is configuration comparison.

When it comes to the cost, the construction can also not be called inexpensive. For instance, a single IMU can cost from 1 € up to 50 € for general market devices with even higher prices for special devices with increased accuracy. As already mentioned before, low-cost IMUs present higher error levels, which means that by saving in the cost of construction, the time needed for enhancement is inevitably increased. Again, introducing additional trackers for validation further increases the cost of the setup.

9.2.1. Experimental setup

To avoid all the enhancement procedures required for physically constructed systems and forego the purchase of a large number of trackers, simulator software is used in this experiment to generate the data of the IMUs. In the simulator, a virtual planar object with an option of adding multiple IMUs on its surface is used. The object can move freely along the designated paths maintaining 6 DoF. The experimental setups considered in the project include from 1 to 16 IMUs; the configurations are presented in Figure 9.110.



(a) IMU-1 (b) IMU-2 (c) IMU-3 (d) IMU-4 (e) IMU-5 (f) IMU-6 (g) IMU-7 (h) IMU-8

(i) IMU-9 (j) IMU-10 (k) IMU-11 (l) IMU-12 (m) IMU-13 (n) IMU-14 (o) IMU-15 (p) IMU-16

Figure 9.110.: IMU positioning patterns. Source: [226]

The use of a simulator for data generation allows for easy setup of new configurations. Various software exists for sensor data simulation. The requirements for the simulation are high precision and accuracy of the output data, 6 DoF movement, and adjustable parameters for real-life scenarios simulation. In this experiment, CoppeliaSim (previously known as V-Rep) by Coppelia Robotics is used (see Appendix A for the description of simulation

model development). This software is developed to simulate real-life scenarios for different robotic components for instance robot arms, hexapods, etc. It also has a wide range of other components for simulation purposes, e.g., infrastructure, furniture, household items, office items, and even humans. This software also offers a range of usable virtual sensors, for instance, accelerometers, gyroscopes, vision sensors, laser scanners, GPS sensors, etc. On top of that, the tool provides online insight into the system state.

The simulation path represents the object's movement in an MR scenario. Therefore, creating a simulation path to be as realistic as possible is one of the key goals for this experiment. The path controls the pose of the object (in the case of this experiment, the pose of the plane with attached IMUs). For the experiment, multiple paths needed to be created to make sure the dataset is diverse and neural networks can train on different data points. To ensure the integrity of the paths the object should follow the rules of 6 DoF.

Two distinct test paths for evaluating the models were generated: a short Nascar-track-shaped simple path (see Figure 9.111) and a more complex long random path (see Figure 9.112). Both of the test paths maintain the properties of 6 DoF.

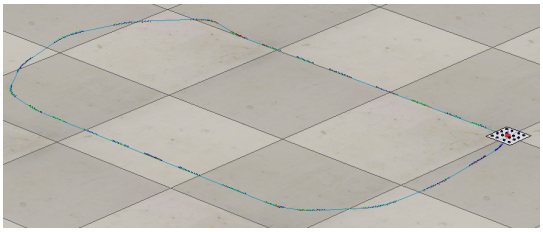


Figure 9.111.: Nascar test path. Source: [226]

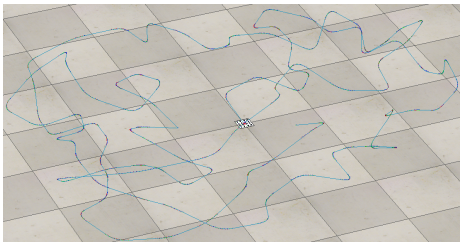


Figure 9.112.: Random test path. Source: [226]

The data generated from the virtual IMUs can be fed into the neural networks to predict the object's pose. In this project, the main focus is on developing Recurrent Neural Network (RNN) [226] and Convolutional Neural Network (CNN) [144] models, which are trained using the fused sensory data coming from the IMUs. The goal of the models is to predict object pose as accurately as possible. Multiple instances of RNNs and CNNs are built to assess and compare the quality of pose estimation.

9.2.2. Neural Network Architecture

This subsection is devoted to the description and explanation of neural network design choices that were made within the scope of this experiment.

The neural network models were build, trained, and analysed with *Google Colaboratory (Google Colab)*, a free cloud computational service that provides access to Google cloud server instances, GPU (Graphics Processing Unit) and TPU (Tensor Processing Unit) instances with a browser-based Python notebook interface [99].

An *optimizer* in the context of neural networks is an algorithm that is used to find the (global) minimum/minima of the loss function. There are multiple optimizers available for

neural networks, including but not limited to Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSProp), Adaptive Moment Estimation (Adam), Nesterov-accelerated Adaptive Moment Estimation (Nadam). In this project, Adam was used for optimization. The concept of this method is to gradually add momentum to the solution. Initially, the learning process is slow, but after some steps, it picks up the speed. Adam allows making different time steps for different parameters to update. With the help of momentum, it leads to faster convergence.

Loss functions are the means to achieve the training: they evaluate how well does the network model a given dataset. The use of a loss function allows rewarding or penalizing a particular prediction to achieve the principal goal, i.e., to minimize the error. Loss functions are often referred to as cost functions and objective functions [98].

Since the problem considered in this research is a regression problem, there are two viable loss functions to optimize the algorithms: Mean Absolute Error (MAE) and Mean Squared Error (MSE). The definitions for MAE and MSE respectively are as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}_i|, \tag{9.7}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}_i|^2, \tag{9.8}$$

where y_i is the actual label value and $\bar{y}_i$ is the predicted value.

Both of these loss functions have advantages and disadvantages. MAE is a more robust loss function, in the sense that it works better in presence of a large number of outliers. On the other hand, MSE penalizes long distances or large errors, because it involves a square operation. MSE has a higher gradient for large values. MAE allows the network to learn at a constant speed, which is not the case for MSE [98].

One of the obvious choices for *accuracy metrics* is the accuracy of the prediction. It can provide an overview of how the model is predicting a new unseen set of data after the training is completed on a different dataset.

Another accuracy metric that can be used is MSE vs. MAE comparison: If one of the loss functions is chosen for computations, the other one is used in the accuracy metrics. This metric allows to monitor the whole training and predicting process.

Next, to evaluate to performance of the models and select the best performing ones, the mean and variance of the predicted errors had also been taken into account.

For the *model fitting*, following the classic approach of train-test-split, the available data were used as follows: 85% of the data for training, 10% for validation, and the other 5% for testing purposes.

The number of epochs was set to 20 and the dataset was divided into 32 batches for better memory usage.

Callback functions are used to monitor the performance. Validation loss is used as the monitoring unit with a patience of 10, i.e., the training stops automatically to save time and

resources if there is no improvement in validation loss after 10 iterations. *TensorBoard* had been used to visualize the network architectures, information flow and most importantly having the logs permanently saved for later use.

TensorBoard is a visualization toolkit of TensorFlow. TensorBoard offers a wide variety of visualization options [267], including: tracking and visualizing metrics such as loss and accuracy; visualizing the model graph (Optimization and Layers); visualizing histograms of weights, biases, etc. as they change over each iteration; visualizing custom images, texts related to the models; custom plots.

The CNN model overview is presented in Figure 9.113. The training of around 100 different CNN models was performed by using simulation data from the simulation software.

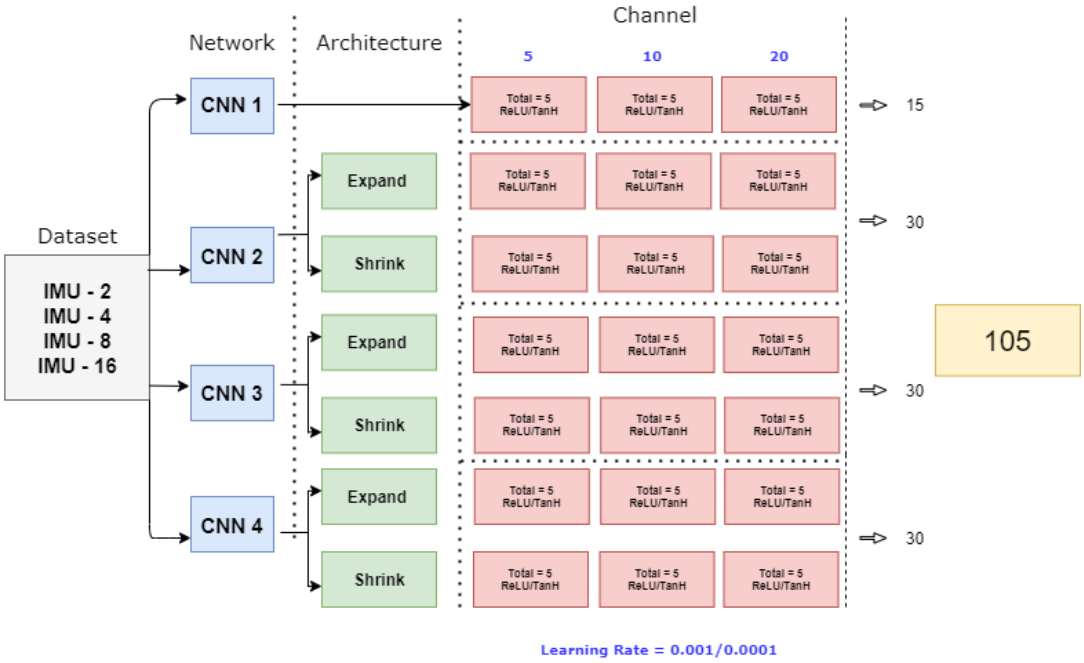


Figure 9.113.: CNN model overview. Source: [144]

The CNNs designed for the experiment consist of a different number of layers (from 1 to 4) with shrinking or expanding architecture using 5, 10, or 20 data channels as input stream. Two different activation functions were used for CNNs: ReLU (Rectified Linear Unit) and Tanh.

As for the number of channels, a traditional approach is to get an approximation of the pose from 1 measurement of the data and compare them to the ground truth collected for the same time step. However, the principal of CNNs calls for convolution of several input data for proper functioning. That is, N measurements are collected into a set and the pose is predicted based on the combination of them and can be compared with the ground truth obtained for measurement number N. Then, the next N data are provided and compared with $2N$ and so on.

By channelling the data it is possible to distinguish between the two common accelerometer data states: zero acceleration with constant (non-zero) velocity and zero acceleration with zero velocity. The channelling in this case provides a history of measurements that allows understanding, which state does the object have.

In this experiment, 1D convolution layers are used instead of the 2D convolution layers. While 2D convolution works well for image and video data, 1D convolution is best suitable for time series data (which is the case study). The main difference between 1D and 2D convolution is that in 1D convolution the kernel slides along one dimension wherein in 2D convolution the kernel slides along 2 dimensions (width and height). Figures 9.114 and 9.115 show the difference between 1D and 2D convolutions.

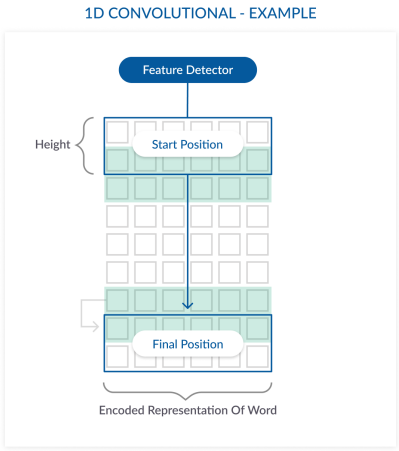


Figure 9.114.: 1D convolution architecture. Source: [143]

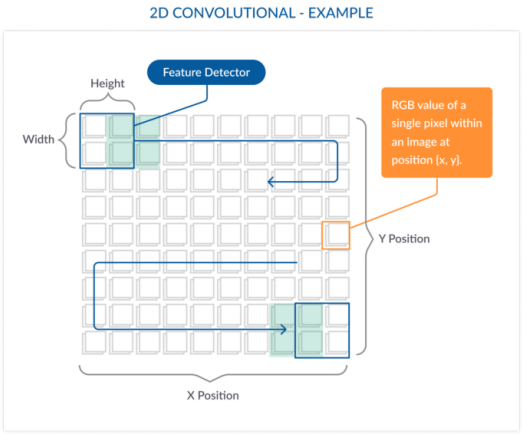


Figure 9.115.: 2D convolution architecture. Source: [143]

The general RNN model overview is presented in Figure 9.116. Three types of recurrent networks were chosen for this experiment: Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bi-Directional LSTM (Bi-LSTM). During the experiments, the caviar approach is chosen for selecting the best model, i.e., the structures of the networks are fixed and only configurations (number of hidden layers, recurrent units, learning rate, etc.) are varied.

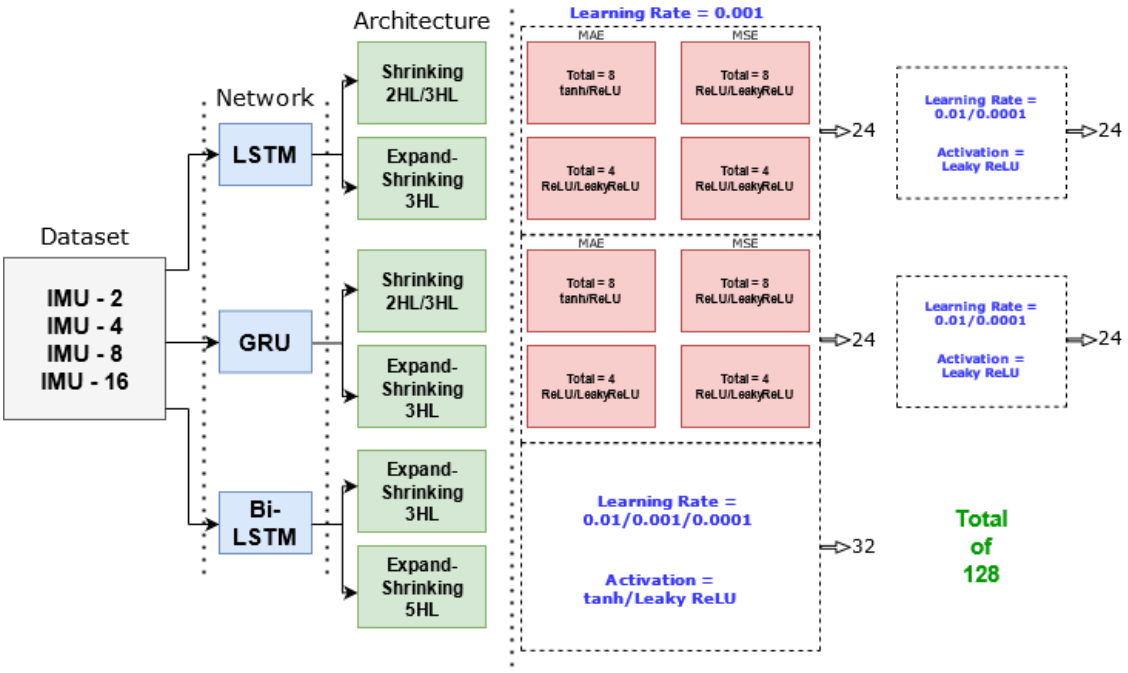


Figure 9.116.: RNN model overview. Source: [226]

For RNN, the expand-shrinking models are chosen to be built on three hidden layers, because two hidden layers are not enough to employ the expand-shrinking mechanism. For the Bi-LSTM models, models on 3 and 5 hidden layers are built to have better prediction performance, because Bi-LSTMs are known to perform better in sequence learning problems with more parameters [98].

The comparison was performed for various loss functions, activation functions, and learning rates. MAE and MSE are used as the loss functions interchangeably with Tanh, ReLU, and Leaky RelU as activation functions. The intermediate results can be found in [144, 226]. Here only a short summary of findings is provided to show that Machine Learning approach has a potential for pose estimation.

9.2.3. Conclusions

The research described in this section was focused on comparing artificial neural network algorithms for pose estimation with Multi-IMU. The goal was to develop a system that could test the models of potential Multi-IMU systems and select the best available neural network for the data fusion with provided estimates of the configuration quality.

The neural networks of two types were compared in this project: convolutional and recurrent, with different configurations, loss functions, activation functions, and learning rates.

Based on the observation of the results of the training and tests performed for 2 different test path the following conclusions can be drawn:

- There is no single NN configuration that outperforms others on every dataset.
- Training dataset needs to be diverse, including data of different paths.
- Expanding NN architecture generally performs well in all datasets.
- Networks with larger amounts of hidden layers generally perform better.
- The choice of loss function affects the results; in general, networks with MAE outperformed those with MSE in the observed experiments; however, some of the better performing networks are MSE-based.
- For RNN: LSTM and GRU perform similarly for the given problem.
- For CNN: The usage of multiple-channel input data provides better results than single-channel data.
- RNN generally performs better than CNN in the conducted experiments.
- A combination of CNN and RNN can be used in a sequential or parallel manner to further improve the pose estimation quality.

The research was only conducted with simulated ideal sensor data. However, measurement noise can be modelled by adding new layers to the designed networks or data preprocessing, so it constitutes no principal drawback. The presented results show that the underlying system can be used as a framework for designing the Multi-IMU systems, including the number of IMUs as well as their placement. The framework can then be used for testing particular configurations before the physical development of a working prototype.

"Interaction is the essence of all user experiences. It is the conversation between your product and your user, and if the conversation is boring, your user will leave and talk to someone more interesting."

— Jerry Cao

10. Multi-IMU Interaction Applications

As the research on which this work is based was done with the view of potential applications in Mixed Reality, this chapter is devoted to existing and potential interaction applications of Multi-IMU.

As stated in the previous chapters of this work, applications that use IMUs (including those using multiple IMUs) usually either only use the orientation-related information from the obtained results of pose estimation or incorporate additional sensors of different nature (e.g., acoustic, GPS, optical) to enhance tracking. This chapter is therefore divided into two sections corresponding to a review of these two classes of applications.

It should be noted that some applications don't require tracking as defined earlier in the work; instead, the system only needs to *classify* the action of the user for interaction. That is, after obtaining the data the task is to tell the type of the action. After the type of the action is identified, the system can then give some response, thus performing the interaction (see for example [241, 154]). Although this class of problems also studies interaction with (potentially) the use of IMUs, it does not rely on pose estimation as such, and therefore is not considered further in this work.

10.1. Applications with additional sensors

A very common direction of research is fusion of Multi-IMU with GPS to track the object outdoors or indoors. The quality of GPS tracking is generally good in zones with good signal reception, but when the signal is affected GPS becomes unreliable. Furthermore, the use of GPS only allows for tracking on long trajectories, because the precision of tracking in small quarters is lacking. However, by fusing the data obtained from GPS with additional sensor data, tracking can be performed also indoors or in close quarters. For example, the authors of [18] consider tracking of pedestrians in a stadium (see Figure 10.1). The authors compare several configurations of sensors and then – several fusion techniques, including a stacked filter (centralized Kalman filter) and Federated No Reset filter (master filter performing least squares).

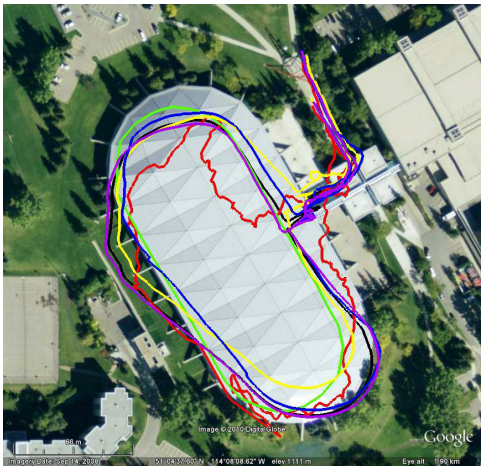


Figure 10.1.: Map view of best performing filters. True solution (green), Standalone GPS solution (red), Single IMU (yellow), Fused IMUs (blue), Stacked Filter (black), Federated No Reset solution (purple). Source: [18]

The model and numerical method developed in this paper allow fusing GPS and Multi-IMU data to obtain reliable tracking of a moving person even in zones with flawed satellite reception. The provided comparison also shows the advantage of using multiple sensors as compared to single ones.

Another research that uses GPS in combination with Multi-IMU is presented in [243]. The setup is additionally equipped with a camera that is used for tracking as well as for obtaining images for superimposing virtual objects after processing. The setup is presented in Figure 10.2.

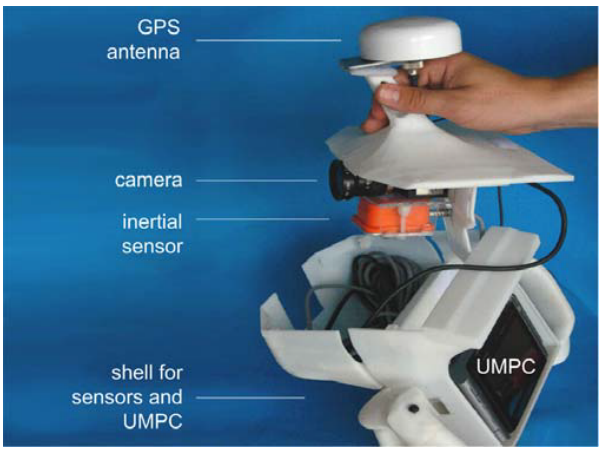


Figure 10.2.: Outdoor AR platform. Source: [243]

The research provides a detailed description of the developed fusion algorithm. The architecture is presented in Figure 10.3. Although the base of the fusion algorithm is a series

of Kalman filters, the data acquisition and preprocessing are also considered in detail in the paper.

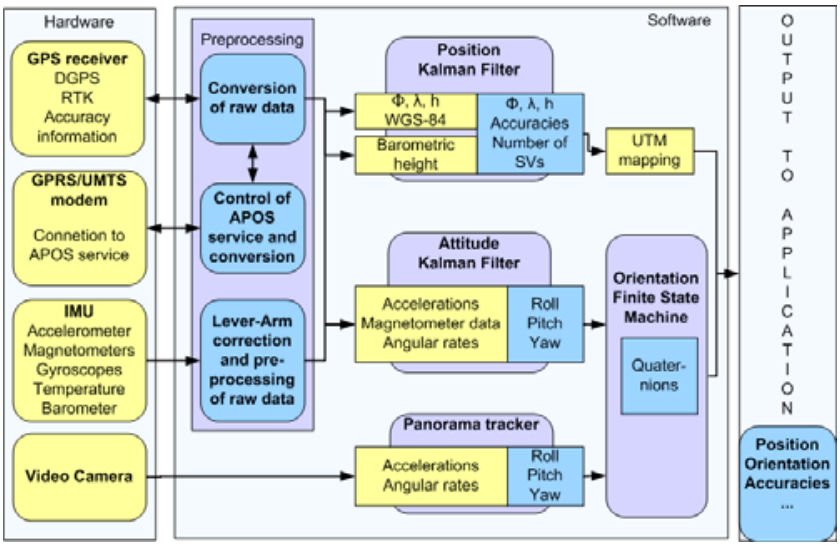


Figure 10.3.: Multi-sensor fusion system architecture. Source: [243]

The developed multi-sensor fusion system allows presenting virtual objects overlaid on top of the real environment (within the platform framework). An example is shown in Figure 10.4.

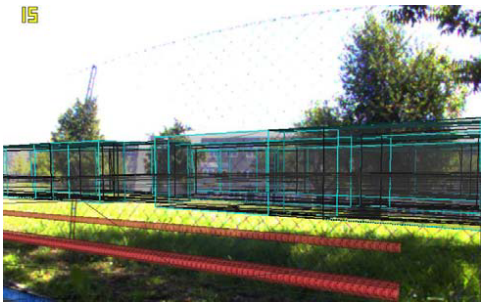


Figure 10.4.: Screenshot taken on AR platform showing underground assets (water mains in blue, electricity lines in red) and wire frames of buildings. Source: [243]

Paper [63] focuses on fusion of multiple IMUs and multiple cameras in order to perform high-quality tracking. The authors first consider fusion in a system consisting of one IMU and one camera with an Extended Kalman Filter. They then extend this approach for a system consisting of multiple IMU-camera pairs. They further advance the method by adding online calibration and registration of the sensors and also considering sensor failure cases. They later test the proposed method on a series of simulated examples and some real-life examples, both times showing promising results even in presence of sensor dropouts (see Figure 10.5).

The authors consider the application of their developed method for Augmented Reality and robotics, specifically for multi-robot systems.

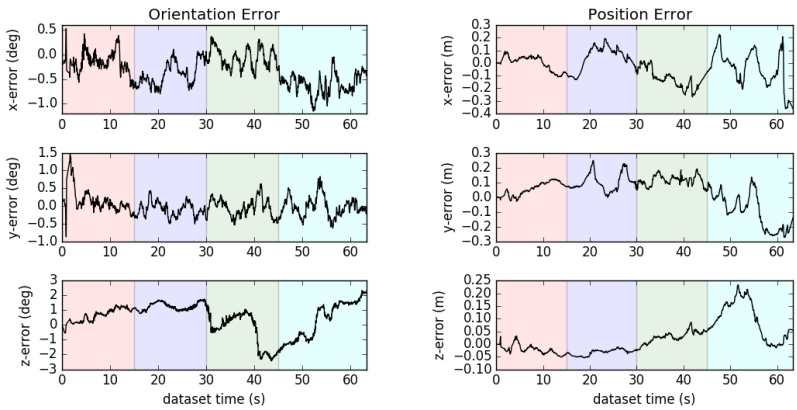


Figure 10.5.: Pose estimation errors of the proposed Multi-IMU Multi-Camera Visual Inertial Navigation System in the presence of sensor dropouts. Source: [63]

Paper [218] describes a body sensor network from Xsens. The system consists of several MEMS-based trackers placed independently on different parts of the body (see Figure 10.6). The trackers consist of an accelerometer, a gyroscope, a magnetometer, a barometer, and a thermometer. The authors only estimate the orientation of the tracked parts in relationship to each other. The focus of the paper is on the wireless transmission of the data and on the fusion of multiple sensors.

Figure 10.6.: Components of Xsens body sensor network placed on the body. Source: [218]

There are also approaches that use the inverse kinematics models (which are the main topic of the next subsection) together with additional sensors or equipment. As they don't always only rely on orientation, but can also incorporate positional data from IMUs, they are considered in this section.

For example, the authors of [250] propose an approach based on a kinematic model. The schematic of the underlying model is presented in Figure 10.7. Here the shoulder joint is assigned 3 degrees of freedom, elbow joint 1 DoF, and wrist again 3 degrees of freedom. The link lengths parameters are associated with the upper arm and the forearm.

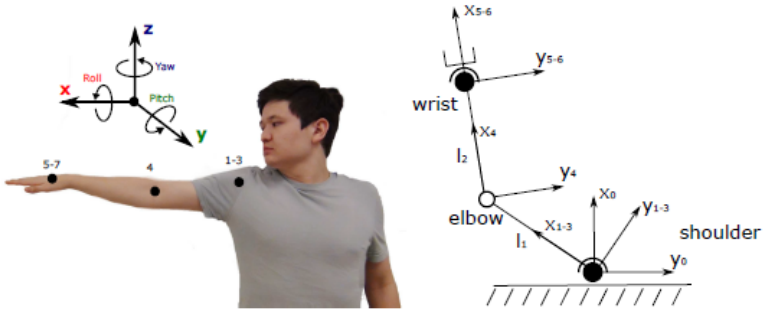


Figure 10.7.: A human arm relative to a fixed global frame with indicated DoF (left) and a simplified human arm kinematic model (right). Source: [250]

The complete tracking system consists of several IMUs completed with multiturn potentiometers that help determine the angle between the forearm and the upper arm. The paper also provides an experimental case study on robot teleoperation, although the authors conclude that the quality of interaction could be improved by introducing additional constraints implied by the selected kinematic model.

10.2. Applications only using orientation

The idea of using only orientation, as opposed to using the full pose (i.e. orientation and position), is motivated by the fact that the orientation estimation with IMUs is easier and gives better results. Even the use of Multi-IMU with additional enhancement techniques leaves more to be wanted from the position estimation. However, the usage of only the orientation limits the potential applications, as, for example, some gestures that could be used for interaction, consist more recognizably of translational movement, and can therefore not be traced well when only relying on orientation.

To overcome the potential disadvantages of the approach, researchers usually tend to enhance the tracking by introducing underlying motion models.

Paper [189] considers tracking of an arm with Multi-IMUs using a simplified kinematic model of the human arm with 7 degrees of freedom (orientations), as represented in Figure 10.8. There are 3 degrees of freedom in the shoulder joint, one degree of freedom in the elbow, and 3 degrees of freedom in the wrist. The lengths of the upper arm and the forearm are constant and considered as parameters in the model. The system implies some constraints that help reduce the drift present when estimating the movement with inertial sensors. Based on the orientation of the joints and the lengths of the denoted segments it is possible to determine the arm posture.

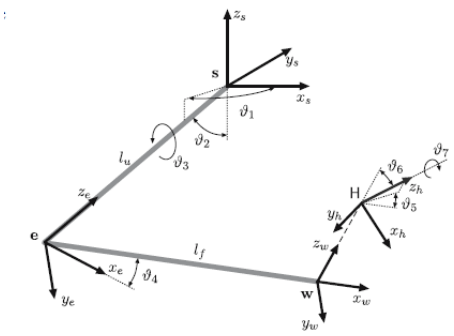


Figure 10.8.: Simplified kinematic model of the human arm. Source: [189]

The author performs simulations to verify the pose estimation method and also provides verification through a series of experiments. A rehabilitation robot and an optical tracking system are used to assess the quality of tracking: the user's hand is fixed in the robot's grip and the interaction forces are measured to provide controlled input (see Figure 10.9), while the optical tracker registers the movements. The author concludes that the quality of tracking could be further enhanced by either physically restricting the motions of the user or by developing a more complex kinematic model.

Figure 10.9.: Simplified kinematic model of the human arm. Source: [189]

The authors of [305] use a similar model and apply it to full-body tracking. The human body is represented by 15 segments connected by joints (see Figure 10.10).

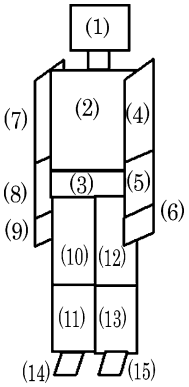


Figure 10.10.: Model of the human body with 15 segments. Source: [305]

A 9-DoF IMU is placed on each of the segments at its midsection. The segment is represented by an orthogonal frame corresponding to the sensitive axes of the IMU (see Figure 10.11). The orientation between the two connective segments is parametrised with angle θ. That angle is estimated during the tracking.

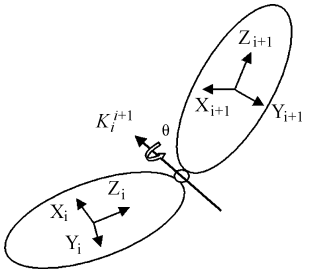


Figure 10.11.: Schematic of the segment model. Source: [305]

The results presented in the paper show that the developed model with the algorithm allows for pose estimation with the accuracy suitable for potential applications.

A similar model is used in [252] to track the upper body. The authors propose the usage of a wearable Multi-IMU system for controlling an industrial robot. As seen in Figure 10.12, the developed system aims at preserving the operator's movement capabilities in order to increase the usability of the control system.

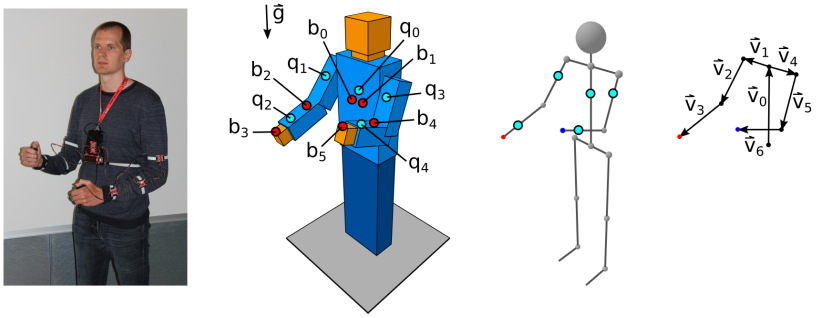


Figure 10.12.: Developed upper body tracking system: (left to right) the wearable Multi-IMU system; the positions of sensors; the skeleton abstraction; the vector abstraction. Source: [252]

The underlying kinematic model is similar to the ones described before; the paper is more focused on the application of the prototype for controlling an actual collaborative robot, taking into account such potential pitfalls as communication delays and package loss along with calibration and registration of the system.

Having observed the presented solutions in the field, the author in collaboration with J. Oehm considered the development of a tracking device consisting of a smartphone and an Arduino Nano 33 with an embedded 9-axis IMU [205]. The sensor network, presented in Figure 10.13, is developed for tracking the orientation of the arm segments for Mixed Reality.

Figure 10.13.: The sensor network consisting of a smartphone and an Arduino Sensor. Source: [205]

The schematic representation of the arm with sensors placed is presented in Figure 10.14. The system uses the kinematic model, described earlier in this chapter, and employs geometric constraint to prevent the drift in the system. The forearm is only described with the angle θ from the upper arm because the abduction/adduction and pronation/supination are limited.

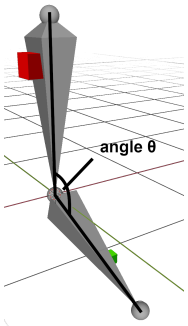


Figure 10.14.: Schematic representation of upper arm and forearm. Source: [205]

Various aspects of tracking device development were considered in the process: selection of the tracking model, choice of materials and components, implementation of software, design of the prototype, control of the hardware-software con-function, and development of interaction applications. The latter, represented by 3 games where the developed prototype is used for control, allow for testing and verification of the device by assessing such aspects of interaction quality as precision, speed, and spatial perception. Screenshots from two of the games, Fruit Ninja (inspired by [107, 108]) and Robot game, are presented respectively in Figures 10.15 and 10.16.

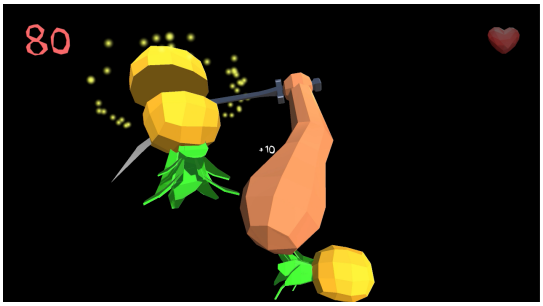


Figure 10.15.: Fruit Ninja game. Source: [205]

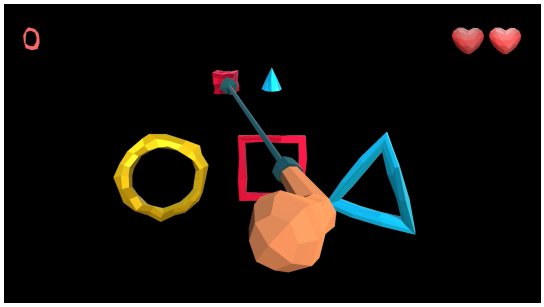


Figure 10.16.: Robot game. Grasping the object. Source: [205]

This chapter was devoted to a short review of applications of Multi-IMU systems for interaction. Most of the applications either use selectively orientation or enhance the tracking by the use of additional sensors of a different type. It nevertheless shows that Multi-IMUs can be used for interaction in applications for Mixed Reality.

"The purpose of life is to contribute in some way to making things better."

— Robert F. Kennedy

11. Conclusion

This chapter gives a summary of the research exhibited in this thesis, draws conclusions based on the achieved results, and gives an outlook of future research opportunities.

The major focus of this research was on Multi-IMU tracking and its application to mixed reality. MR applications naturally require good quality of tracking and that, in turn, requires proper usage of the Multi-IMU systems, i.e., getting reliable and accurate tracking results from the system sensors. That process is complex and consists of several majorly important steps, which were investigated and reviewed in this work.

Chapter 2 was devoted to a survey of the necessary background. A major part of this chapter was devoted to human-machine interaction and immersive technologies. An understanding of the application sphere, in the case of this study - of mixed reality, and of the related topics helps see open questions, better understand the existing technologies and techniques, and find new focus points. Another major part of the chapter was devoted to artificial intelligence. Artificial intelligence plays an important part in many spheres of modern research and everyday life, and by employing it in the developed solutions the new levels of result quality can be reached.

Chapter 3 was concentrated on the inertial measurement systems. Efficient software cannot be developed without an understanding of the hardware design or at least of its main features. Therefore, the first part of the chapter reviewed the principal components of the IMUs. The second part of the chapter was devoted to the types of errors typically exhibited by the IMUs. As the presence of errors in the sensor data is one of the main reasons that can lead to a failure of tracking, it is very important to understand the sources and the nature of these errors.

Chapter 4 focused on calibration, i.e., on elimination of deterministic errors. The calibration in general was briefly discussed, and then three main classes of IMU calibration were reviewed: calibration with auxiliary testing tools, calibration with auxiliary sensors, and calibration without any auxiliary tools. Methods from any of the three classes can generally be used in practice; however, the main motivation of the study of Multi-IMU was its accessibility and comparatively low cost. Therefore, non-aided calibration methods are desirable for practical use with Multi-IMUs. One of the non-aided calibration methods was used in an experiment with Multi-IMU to compare several combined low-cost IMUs with a single high-cost IMU. With appropriate calibration, a low-cost Multi-IMU system is shown to be able to outperform the single high-cost IMU.

Chapter 5 was devoted to the filtering techniques, i.e., on the techniques that eliminate stochastic errors. Frequency-based filtering and Bayesian filtering were considered in the chapter. In general, frequency-based techniques are recommended to be used as a module of an advanced filter. As for Bayesian filters, an accent is made on the applicability of the filters for different models. A special focus is made on nonlinear filtering, which is often undeservingly neglected by the researchers.

Chapter 6 was concentrated on the registration of IMUs to other sensors. IMU to IMU registration was considered first in this chapter. As a common usage of IMU is the enhancement of optical tracking systems, especially in MR applications, IMU to optical system registration was considered next in the chapter. It should also be noted that the advanced MR devices can use Multi-IMU together with cameras to achieve even further enhancement of the results. An experiment showing the importance of registration in Multi-IMU systems is provided in this chapter.

Chapter 7 focused on synchronization. Hardware synchronization and software synchronization were considered in this chapter. A multiplexer was described as a hardware synchronization tool used for Multi-IMU. As for software synchronization, two main branches were considered: time delay estimation + interpolation and SCAAT fusion. The latter technology basically consists of relying on a SCAAT fusion algorithm for dealing with synchronization. The former approach has various practical implementations that were reviewed in the respective sections. A practical problem of Multi-IMU synchronization is considered in this chapter.

Chapter 8 was devoted to a survey of the vast field of fusion algorithms. As the focus of the research in this work is on Multi-IMU systems, data fusion is a major point that should be considered thoroughly and carefully. Centralized and decentralized approaches to fusion were reviewed in this chapter, and fusion by machine learning was discussed and surveyed. An experiment comparing common fusion algorithms in a practical problem is presented in this chapter.

Chapter 9 explained how to bring together all the stages described in previous chapters to obtain pose from the raw sensor data. This chapter, inherently, contains a recipe for pose estimation with multiple sensors and, in particular, with Multi-IMU. A series of experiments is provided for pose estimation with a specifically developed setup through an adapted pose estimation pipeline. The experiments focus on Multi-IMU fusion and comparison of different fusion patterns. The results show a large potential in the use of Multi-IMU, although position estimation can be recommended to be enhanced with pattern recognition. The chapter also presents the results of the research on pose estimation with machine learning for Multi-IMU. The research focused on choosing a suitable artificial neural network in terms of performance and accuracy for a particular Multi-IMU configuration. The provided results show the potential in the proposed approach, and the developed system can be used for Multi-IMU design and virtual prototyping.

Chapter 10 was devoted to the review and discussion of the existing and potential applications of Multi-IMU for interaction in Mixed Reality.

11.1. Future Work

This section is devoted to the suggestions for future studies. These suggestions arose in the course of research and were not brought to life because of the lack of time and because the underlying needed research was not directly aligned with the main objectives of this study.

- Dimension of Multi-IMU
  Additional research is needed to investigate how the number of IMUs constituting a Multi-IMU system affects the computational complexity and accuracy of pose estimation. Specifically, if there is such a number of trackers that if another one is added, no significant accuracy gain can be achieved, while the computational complexity increases.

- SCAAT Fusion Vs. Decentralized fusion
  As reviewed in the corresponding sections of Chapter 8, a SCAAT algorithm allows for automation of synchronization. In decentralized fusion architecture, on the other hand, synchronization has to be performed separately. Although both approaches are extensively used in practice, no conclusive comparison has been performed for Multi-IMU.

- Noise IMU data fusion by Machine Learning
  Data fusion with ML is a point of interest for many researches, including this work. However, primarily because of the time constraints, the proposed approach was only validated on simulated data and not on measured ones. The results obtained on simulated data are particularly promising, so a real-data validation is of major interest.

- Comparison of numerical integration approaches
  Pose estimation with inertial tracking relies on numerical integration and therefore depends heavily on the choice of a particular integration method. The field of numerical integration approaches is vast and some of the approaches that were not used in this study can be of interest, e.g., the Riemann integration technique.

- IMU arrangement
  One of the evaluation experiments was devoted to a comparison between two arrangement schemes of Multi-IMU: linear and planar. The experiment showed that the pose estimation result depends rather much on the relative arrangement of system components. Therefore, other arrangement schemes should be designed and studied, in particular, the non-planar ones, that were not considered in this work at all.

- Full body tracking
  As a far-fetching goal, full-body tracking can be performed with several Multi-IMU composites. This goal deserves its own special study that should include hardware design, usability, and ergonomics testing, analysis of scaling possibilities as well as, clearly, registration, synchronization, and fusion algorithm development.

The goals, stated in the introduction of this thesis, were achieved. New potential research questions were discovered in the course of the study. As the end result, this work can be seen as a recipe for getting reliable and accurate measurements from Multi-IMU systems.

"All laws are simulations of reality."

— John C. Lilly

A. Multi-IMU Simulation

In this appendix, the creation of a simulation setup of a Multi-IMU system in CoppeliaSim (previously known as V-Rep) by Coppelia Robotics is demonstrated.

A.1. Scene Object

To simulate a Mixed Reality scenario, first, an object for the scene has to be created that will embed the characteristics of a moving component in the MR application.

For the considered application, an object is modelled to contain Multi-IMUs that move on a pre-defined path. For proper simulation, 6 DoF has to be ensured to correctly represent the real-world motion. The object can be modelled from available primitive shapes: plane, disc, cuboid, sphere, and cylinder as the options. In this scenario, a two-dimensional plane was chosen to allow for 16 IMUs to be potentially placed on to it in a planar grid.

To add the desired object to the scene, following steps should to be followed (see Figures A.1, A.2):

1. Click on 'Add' on the menu bar.
2. Go to the first option; 'Primitive Shape'.
3. Click on 'Plane' to add a two-dimensional plane to the scene.
4. Put the desired parameters in the 'Primitive Plane' dialog box and click 'Ok'.

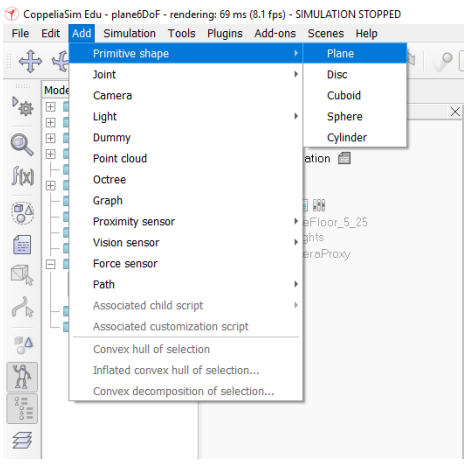


Figure A.1.: Adding 2D plane to the scene

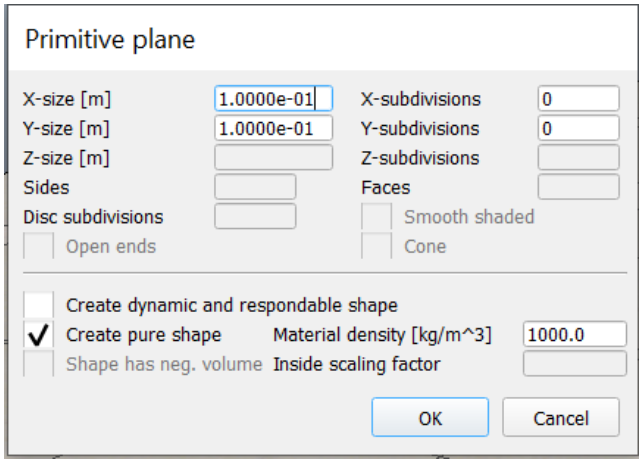


Figure A.2.: Plane Configuration dialog

The plane newly added to the scene looks as shown in Figure A.3.

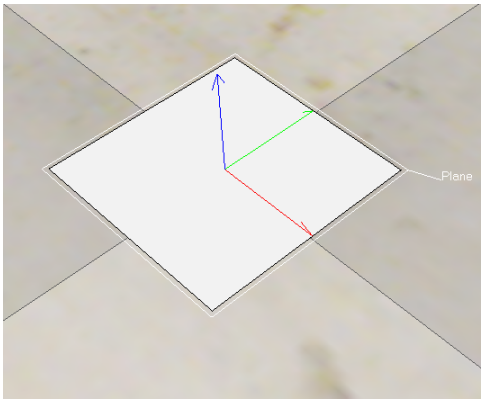


Figure A.3.: Primitive Plane

A.2. Inertial Measurement Units

The next step is adding the IMUs to the plane. Adding IMUs in CoppeliaSim means adding the IMU components, in this case, accelerometers and gyroscopes – one of each to simulate a single IMU. To add the virtual sensors to the scene, the following steps should be followed:

1. Click on 'Components' in the 'Model browser' pane. See Figure A.4.
2. Click on 'Sensors' to open a new pane for all the available sensors.
3. Scroll down the pane to select the desired sensors (accelerometer, gyroscope), see Figure A.5.

4. Drag and drop the sensors onto the plane in the scene.

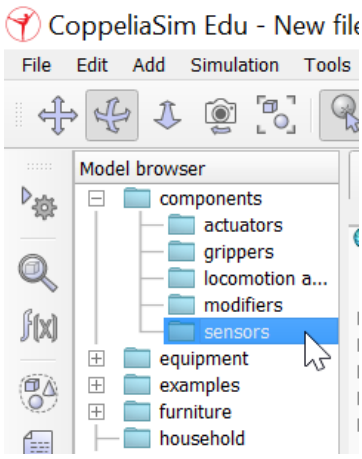


Figure A.4.: Select sensor components

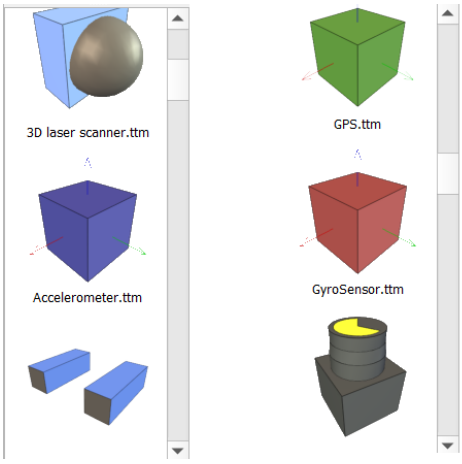


Figure A.5.: Select sensors

As the goal of the simulation is to model a Multi-IMU system, the two sensors (accelerometer and gyroscope) need to be attached together or merged as a single IMU. This can be either done by drag-and-dropping or by explicitly setting the positions of both sensors to be the same. Another option is to create a custom object consisting of the two sensors through the control panel. The description of the panel is provided in Section A.4.

A.2.1. IMU Creation

By default, the simulation software provides a non-threaded child script for each sensor. The default language for the scripts is Lua; however, the scripts can also be written in C++. The default code for the used sensors can sense the acceleration and angular velocity of the IMUs for each millisecond. The data sensed by the IMUs were written to files in dynamic mode; the script creates a designated number of text files for the accelerometer and gyroscope, which is defined by the control panel when the simulation starts. To implement this, the non-threaded script attached to the sensor needs to be modified, see Figure A.6.

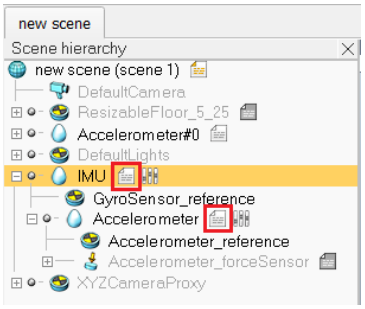


Figure A.6.: Opening child script

If the script is not specified as non-threaded, the parent sensor object can terminate the thread prematurely.

For easier creation of a sensor combination representing an IMU, a separate scene should be used. Then, one of the sensors should be added (e.g., a gyroscope) and a Lua script should be created for it, specifying that another sensor will be attached. Then, the other sensor should be added directly to the gyroscope in the *Scene hierarchy* pane, with a specified Lua script. After that, the parent sensor (in this case, the gyroscope) has to be saved as a new model, that can then be used in the global scene through the custom control panel.

A.3. Simulation Path

This section explains the generation and usage of a simulation path.

First, a dummy object has to be created to specify the point of entry of the simulation process and control the process itself. The following steps need to be undertaken (see Figure A.7):

1. Navigate to the 'Scene hierarchy' pane.
2. Open context menu of the scene name.
3. In 'Add' click on 'Dummy'.

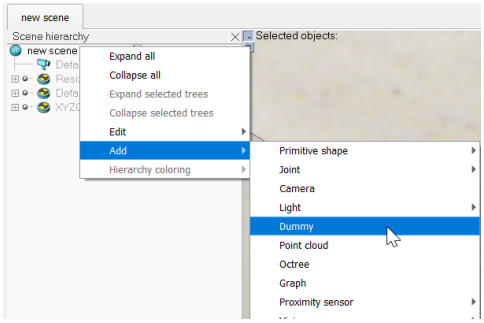


Figure A.7.: Adding dummy object to the scene

For future references, the object should get a recognizable name. In this simulation, it was named IMU_Simulation. The object is placed at the center of the plane for easier registration (see Figure A.8).

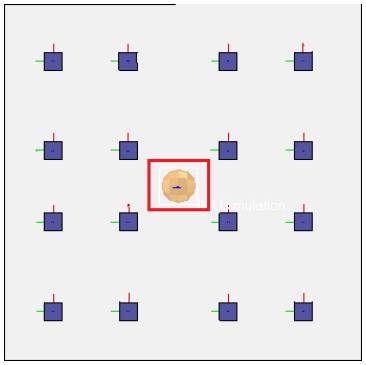


Figure A.8.: IMU_Simulation object on the plane

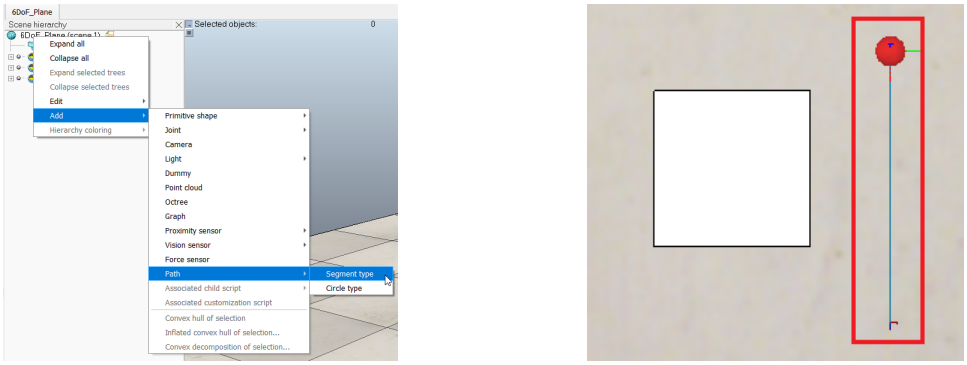
Administration of the simulation can be made easier by adding customization scripts to the IMU_Simulation object. That allows for the control of, e.g., the number of IMUs in the setup, as well as the acceleration and velocity of the plane.

The starting point of the paths was chosen to be at the location of IMU_Simulation for easier registration.

CoppeliaSim provides two default path types — segment type and circle type; both path types were used for the data collection. To add a segment type path to the scene, the following steps are needed (see Figure A.9):

1. Go to 'Scene hierarchy' pane.
2. Open context menu of IMU_Simulation.
3. Go to 'Add' and then go to 'Path'.
4. Click on 'Segment type' to create the primitive path.

A straight line with two control points was created, shown in Figure A.9(b). The newly created path can also be renamed by double-clicking the path in the 'Scene hierarchy' pane.



(a) Adding a segment type path

(b) Segment type path

Figure A.9.: Adding a segment type path to the scene

After the path was added to the scene, it can be modified and the needed parameters can be set.

For the purpose of the Multi-IMU simulation that will be used in the Machine Learning experiment, the paths should satisfy the 6 DoF requirement and be rather diverse to show off the machinery.

For the described experiment, two short paths and two long paths were created. For the shorter paths, the chosen path shapes are:

1. Circular path with constant elevation gain and loss.
2. Vertical rectangular path.

The longer paths are intended to be used for NN training. For that reason, it was decided to make them random.

Editing of a path is performed in Path edit mode, which allows adding path control points and modifying the pose parameters. The Path edit mode can be entered as shown in Figure A.10.

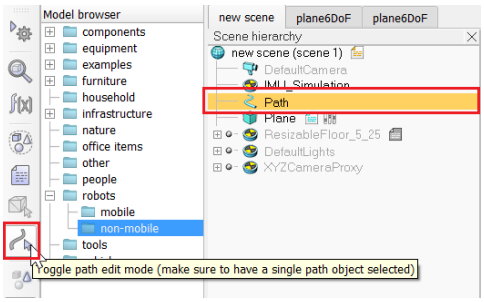


Figure A.10.: Entering the Path edit mode

When in the Path edit mode, a pane containing the control points for the selected path is opened, as shown in Figure A.11. Initially, the pane contains only two control points since the created path is a segment-type path (i.e., a straight line).

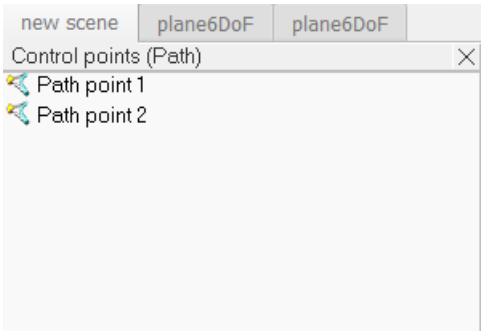


Figure A.11.: Path control points pane

Next, the path should be made longer. This can be achieved by adding more control points. To do that, the user needs to navigate to the last control point, open the context menu and choose 'Insert new control point after selection'. The new point is created right on top of the last point by default; this behavior can be changed by first selecting the point and then toggling the 'Object/Item shift' button on the top control panel. This option allows moving any object in the scene along all three axes. Two types of movement are allowed through this option (see Figure A.12):

1. Move an object along the X-axes and Y-axes by simple drag-and-drop.
2. Press 'CTRL' in the keyboard and move the cursor up and down to move the object along Z-axis.

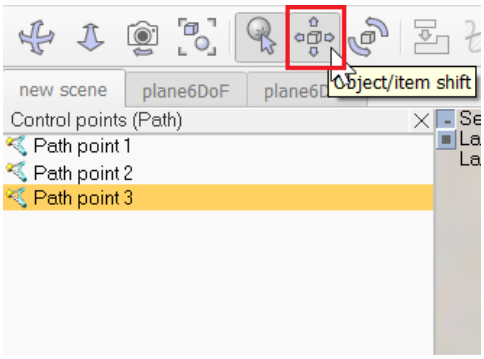


Figure A.12.: Toggle item shifting

To ensure 6 DoF properties for the simulation plane, the orientation of the path control points needs to be modified.

By default, if there is a long distance between two control points, the object tends not to change its orientation while following the path; rather. the object keeps the same angle from point A to point B. Each control point has three orientation angles for three axes: α, β, and γ. These angles (set in degrees) control the orientation of the object. For example, consider the

three angles for control point A: $\alpha = 45°$, $\beta = 90°$, and $\gamma = 0°$. For control point B, consider $\alpha = 0°$, $\beta = 0°$, and $\gamma = 90°$. Now, when the simulation is running, the object moving from point A retains the angles defined in A until it reaches point B and only then changes the angles to the new values. This does not align with real-life or MR scenarios, because generally, the object needs to change its orientation frequently along the path. Therefore, the change of orientation along the path has to be specified manually.

The suggested solution for this situation is to create long paths using several control points that have multiple orientation values. It is important to note that, 'Automatic Orientation' for the path needs to be unchecked to ensure 6 DoF for the moving object (see Figure A.13).

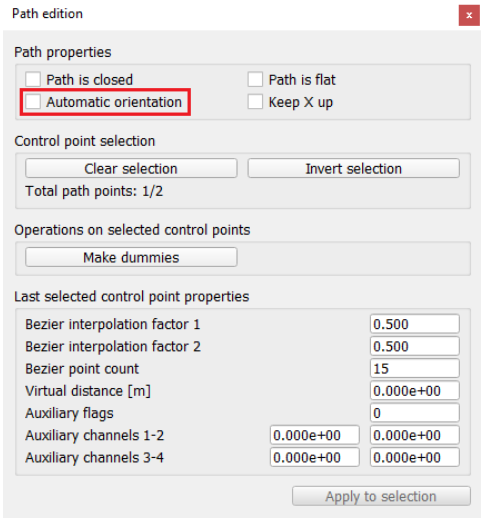


Figure A.13.: Unchecking 'Automatic orientation'

When the 'Automatic orientation' checkbox is unchecked, and the path has several control points with defined orientations, the object can move along the path freely maintaining 6 DoF properties. By creating control points in close proximity with varying orientation values, the object can be forced to change the orientation frequently, replicating an actual MR scenario.

To set the orientation values by changing the default angle values of the control points the following steps need to be followed:

1. Select the latest point on the 'Control points' pane.
2. Click the 'Object/Item' rotate on the top control pane (see Figure A.14). Doing this opens a new control panel for orientation and rotation for that particular control point.
3. Navigate to the 'Orientation' tab.
4. Make sure orientation is relative to the world frame.
5. Change the angle values (α, β, and γ) and click 'Apply to selection' (see Figure A.15).

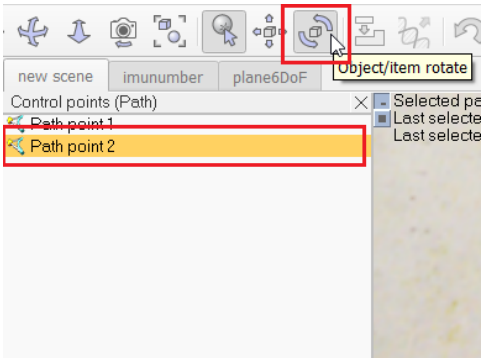


Figure A.14.: Toggling 'Object/Item rotation'

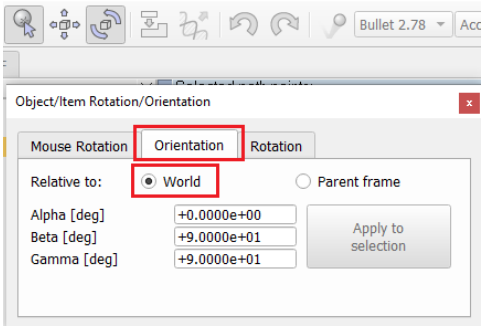


Figure A.15.: Changing control point angles

Note that any item or object in the simulation scene can be rotated using this control panel.

A.4. Simulation Control Panel

After preparing the components of the scene, data collection can be performed from the simulation. To make simulation control easier and more flexible, a custom control panel can be implemented. The control panel can allow changing the basic configurations, start the simulation, and interrupt the simulation if needed. The simulation control panel is shown in Figure A.16].

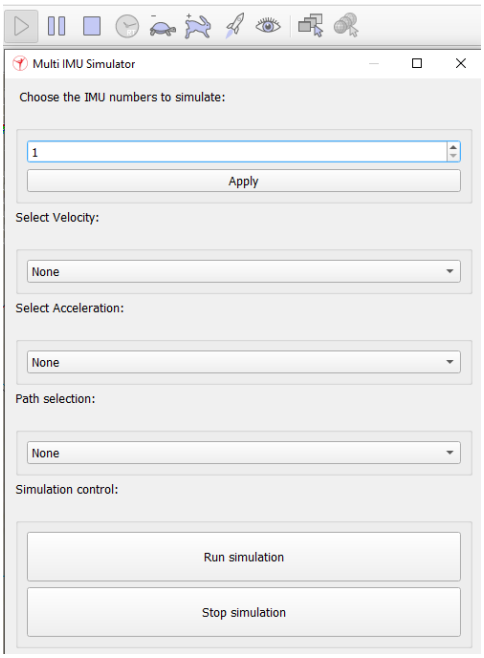


Figure A.16.: Simulation control panel

As the simulation starts at IMU_Simulation by design, a customization script should be added to it. To add the script to IMU_Simulation, the following steps are needed to be followed (see Figure A.17):

1. Right-click in 'IMU_Simulation' on the 'Scene hierarchy' pane.
2. Navigate to 'Add'.
3. Click on 'Associated customization script'.

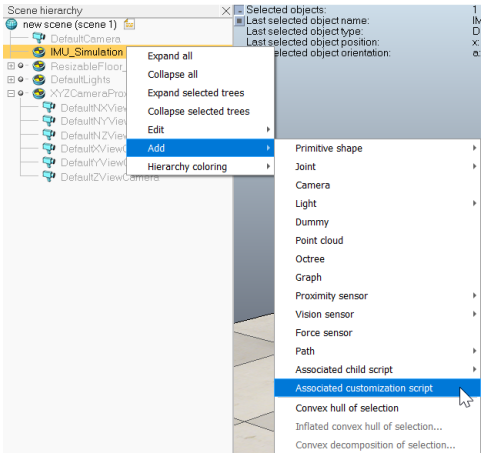


Figure A.17.: Customization script for the control panel

The customization script has predefined methods for the simulation by default. More functions can be added; in this case, custom spin boxed and combo boxed were added to the script. The spin box allows adding IMUs on the plane ranging from 1 to 16. The velocity combo box allows selecting the initial velocity (in m/s). Similar to velocity, the acceleration combo box allows to change the acceleration of the object (in m/s^2). It is also possible to select a path from the paths combo box to make the plane follow a particular path. The values selected from the control panel need to be transferred to the plane to force it to follow the path with the given parameters.

To pass the user-defined values from the control panel, the plane needs to have the list of parameters already defined to receive and use the given values. In the considered case, the parameters are the number of IMUs (noIMU), the acceleration of the plane (Accel), the velocity of the plane (Velocity), and the id of the path the plane should follow (Path_Number). Again, a control panel can be created to make the process easier.

Each object in the scene that can have an associated child script is automatically bound to have 'Use Parameters' control. This control option is attached right beside the object in the 'Scene hierarchy' pane, see Figure A.18.

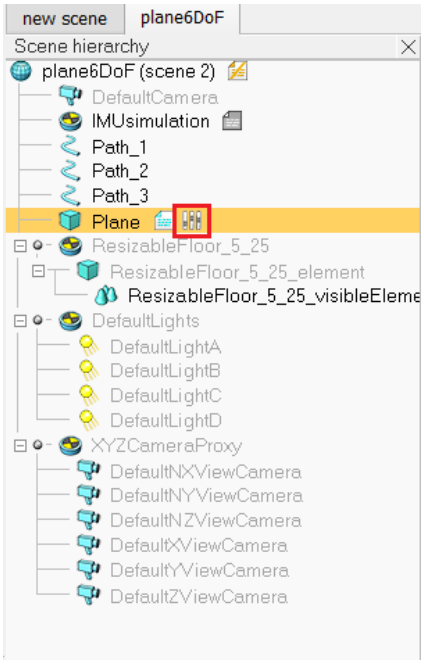


Figure A.18.: User parameters icon

Double-clicking this icon opens up a new control panel (see Figure A.19) that allows adding the parameters the experiment needs for the simulation. For this experiment, only four custom parameters were needed whose values were coming from the simulation control panel.

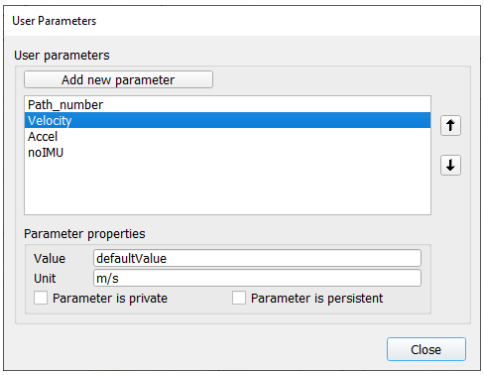


Figure A.19.: User parameters control panel

To add a new parameter, click the 'Add new parameter' button on the top of the control panel. The name of the parameter has to be changed to match with the script by double-clicking the newly created parameter.

Following the described steps, a simulation setup can be created for Multi-IMU simulation in CoppeliaSim software.

"After you understand about the sun and the stars and the rotation of the earth, you may still miss the radiance of the sunset."

— Alfred North Whitehead

B. Rotation Representation

Various formalisms exist to describe rotations in 3D space as a mathematical transformation. The orientation of an object is also described through the same transformations as a rotation can be viewed as the rotation from a reference placement in space.

Although other approaches exist, the three formalisms presented in this appendix can be arguably considered the most used in the fields of mechanics and tracking.

B.1. Rotation Matrix

Rotation matrices $\mathcal{R} \in \mathbb{R}^{3\times3}$ have the following properties:

$$\mathcal{R}\mathcal{R}^\top = \mathcal{R}^\top\mathcal{R} = \mathcal{I}_3, \quad \det\mathcal{R} = 1. \tag{B.1}$$

The properties of (B.1) provide an interpretation of the name of a special orthogonal group $SO(3)$. The orthogonal matrices of dimension 3×3 have the property $\mathcal{R}\mathcal{R}^\top = \mathcal{R}^\top\mathcal{R} = \mathcal{I}_3$ and are part of the orthogonal group $O(3)$. The special notation of $SO(3)$ specifies that only matrices with $\det\mathcal{R} = 1$ are considered rotations.

A vector v can be rotated to u by the use of a rotation matrix as

$$\vec{v} = \mathcal{R}\vec{u}, \tag{B.2}$$

and u to v as

$$\vec{u} = \mathcal{R}^\top\vec{v}. \tag{B.3}$$

B.2. Euler Angles

Rotation can also be defined as a consecutive rotation around three axes in terms of the so-called Euler angles [71].

By considering an angle α for rotating around the x-axis, an angle β for rotating around the y-axis and an angle γ for rotating around the z-axis, the rotation matrix can be expressed as follows:

$$\mathcal{R} = \mathcal{R}_x\mathcal{R}_y\mathcal{R}_z \quad = \\ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & \end{bmatrix} = \\ = \begin{bmatrix} \cos\beta\cos\gamma & \cos\beta\sin\gamma & -\sin\beta \\ \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\cos\beta \\ \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{bmatrix}. \quad \text{(B.4)}$$

Angles α, β and γ are usually called roll, pitch and yaw, respectively.

B.3. Quaternions

Quaternions offer another point of view on rotations. As compared with rotation matrices, the use of quaternions allows saving memory during computations, concatenation of quaternions can be performed faster, and interpolation can be performed more easily.

Quaternions were first introduced by [109] and are widely used in orientation estimation algorithms. A unit quaternion use a 4-dimensional representation of the orientation according to

$$\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^\mathsf{T}, \quad \boldsymbol{q} \in \mathbb{R}^4, \quad \|\boldsymbol{q}\|_2 = 1. \quad \text{(B.5)}$$

A quaternion can be viewed as a four-dimensional complex number.

The quaternion conjugate, denoted by $\cdot^\star$, can be used to swap the relative frames described by an orientation

$$\boldsymbol{q}^\star = \begin{pmatrix} q_1 & -q_2 & -q_3 & -q_4 \end{pmatrix}^\mathsf{T}. \quad \text{(B.6)}$$

The quaternion product, denoted by $\otimes$, can be used to define compound orientations. For two quaternions, $\boldsymbol{p}$ and $\boldsymbol{q}$, the quaternion product can be determined using the Hamilton rule and defined as

$$\boldsymbol{p} \otimes \boldsymbol{q} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix} \otimes \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} = \begin{bmatrix} p_1q_1 - p_2q_2 - p_3q_3 - p_4q_4 \\ p_1q_2 + p_2q_1 + p_3q_4 - p_4q_3 \\ p_1q_3 - p_2q_4 + p_3q_1 + p_4q_2 \\ p_1q_4 + p_2q_3 - p_3q_2 + p_4q_1 \end{bmatrix}. \quad \text{(B.7)}$$

A quaternion product is not commutative: $\boldsymbol{p} \otimes \boldsymbol{q} \neq \boldsymbol{q} \otimes \boldsymbol{p}$.

A three-dimensional vector $\vec{v}$ can be rotated by a quaternion using the relationship described as follows. Let the modified vector $\tilde{v}$ contain 0 inserted as the first element to make it into a 4 element vector:

$$\tilde{v} = \begin{pmatrix} 0 & v_1 & v_2 & v_3 \end{pmatrix}^\mathsf{T}. \quad \text{(B.8)}$$

Then the rotation can be written as

$$\tilde{v}_q = \boldsymbol{q} \otimes \tilde{v} \otimes \boldsymbol{q}^\star \quad . \quad \text{(B.9)}$$

The orientation described by a quaternion $\boldsymbol{q}$ can be represented as the rotation matrix $\boldsymbol{R}$ defined by

$$\boldsymbol{R} = \begin{bmatrix} 2q_1^2 - 1 + 2q_2^2 & 2\left(q_2q_3 + q_1q_4\right) & 2\left(q_2q_4 - q_1q_3\right) \\ 2\left(q_2q_3 - q_1q_4\right) & 2q_1^2 - 1 + 2q_3^2 & 2\left(q_3q_4 + q_1q_2\right) \\ 2\left(q_2q_4 + q_1q_3\right) & 2\left(q_3q_4 - q_1q_2\right) & 2q_1^2 - 1 + 2q_4^2 \end{bmatrix}. \tag{B.10}$$

Euler angles α, β and γ can be obtained as:

$$\begin{aligned} \alpha &= \operatorname{Atan2}\left(2q_3q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1\right), \\ \beta &= -\sin^{-1}\left(2q_2q_4 + 2q_1q_3\right), \\ \gamma &= \operatorname{Atan2}\left(2q_2q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1\right). \end{aligned} \tag{B.11}$$

Given three angles α, β, γ, representing roll, pitch, and yaw respectively, the corresponding quaternion $\boldsymbol{q}$ is:

$$\boldsymbol{q} = \begin{bmatrix} \sin\left(\frac{1}{2}\theta_x\right)\cos\left(\frac{1}{2}\beta\right)\cos\left(\frac{1}{2}\gamma\right) - \cos\left(\frac{1}{2}\alpha\right)\sin\left(\frac{1}{2}\beta\right)\sin\left(\frac{1}{2}\gamma\right) \\ \cos\left(\frac{1}{2}\alpha\right)\sin\left(\frac{1}{2}\beta\right)\cos\left(\frac{1}{2}\gamma\right) + \sin\left(\frac{1}{2}\alpha\right)\cos\left(\frac{1}{2}\beta\right)\sin\left(\frac{1}{2}\gamma\right) \\ \cos\left(\frac{1}{2}\alpha\right)\cos\left(\frac{1}{2}\beta\right)\sin\left(\frac{1}{2}\gamma\right) - \sin\left(\frac{1}{2}\alpha\right)\sin\left(\frac{1}{2}\beta\right)\cos\left(\frac{1}{2}\gamma\right) \\ \cos\left(\frac{1}{2}\alpha\right)\cos\left(\frac{1}{2}\beta\right)\cos\left(\frac{1}{2}\gamma\right) + \sin\left(\frac{1}{2}\alpha\right)\sin\left(\frac{1}{2}\beta\right)\sin\left(\frac{1}{2}\gamma\right) \end{bmatrix}. \tag{B.12}$$

List of Figures

List of Tables

Acronyms

AFSA	Artificial Fish Swarm Algorithm
AI	Artificial Intelligence
ANN	Artificial Neural Network
AR	Augmented Reality
ARMA	Autoregressive Moving Average
AV	Augmented Virtuality
CAD	Computer-Aided Design
CG	Computer Graphics
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning
DNN	Deep Neural Network
DoF	Degree of Freedom
DWT	Discrete Wavelet Transform
EDA	Exploratory Data Analysis
EKF	Extended Kalman Filter
FT	Fourier Transform
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HMD	Head-mounted Display
HMI	Human-machine Interaction
I2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IRU	Inertial Reference Unit

ISO	Inside-out (tracking)
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MEMS	Micro-electromechanical System
ML	Machine Learning
MLP	Multilayer Perception
MR	Mixed Reality
MSE	Mean Squared Error
MUX	Multiplexer
NN	Neural Network
OSI	Outside-in (tracking)
pdf	Probability Density Function
PSD	Power Spectral Density
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RLS	Recursive Least Squares
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RV	Reality-Virtuality
SAA	Systems Application Architecture
SCAAT	Single Constraint at a Time (approach)
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TDE	Time Delay Estimation
TPU	Tensor Processing Unit
UKF	Unscented Kalman Filter
UWB	Ultra Wide Band
VR	Virtual Reality
ZUPT	Zero-velocity Update

Bibliography

[1] A. Abraham. "Artificial Neural Networks". In: *Handbook of Measuring System Design* 2 (2005). Ed. by P. H. Sydenham and R. Thorn, pp. 233–293.

[2] A. Agarwal. *Loss Functions and Optimization Algorithms. Demystified.* URL: https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c/.

[3] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy. "A standard testing and calibration procedure for low cost MEMS inertial sensors and units". In: *The Journal of Navigation* 61.2 (2008), pp. 323–336.

[4] R. Aggarwal. *Bi-LSTM*. URL: https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0.

[5] G. Agresti, L. Minto, G. Marin, and P. Zanuttigh. "Deep Learning for Confidence Information in Stereo and ToF Data Fusion". In: *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017* 2018-Janua (2017), pp. 697–705. DOI: 10.1109/ICCVW.2017.88.

[6] H. Alipour, D. Zeng, and D. C. Derrick. "AdaBoost-based sensor fusion for credibility assessment". In: *ISI 2012 - 2012 IEEE International Conference on Intelligence and Security Informatics: Cyberspace, Border, and Immigration Securities* (2012), pp. 224–226. DOI: 10.1109/ISI.2012.6284314.

[7] D. W. Allan. "Should the classical variance be used as a basic measure in standards metrology?" In: *IEEE Transactions on instrumentation and measurement* 2 (1987), pp. 646–654.

[8] J. Alves, J. Lobo, and J. Dias. "Camera-inertial sensor modelling and alignment for visual navigation". In: *Machine Intelligence and Robotic Control* 5.3 (2003), pp. 103–112.

[9] AMI. *Aptio V UEFI BIOS Firmware*. URL: https://ami.com/en/products/bios-uefi-firmware/aptio-v/.

[10] M. S. Andrle and J. L. Crassidis. "Geometric integration of quaternions". In: *Journal of Guidance, Control, and Dynamics* 36.6 (2013), pp. 1762–1767.

[11] Apollo Computer. *Getting Started With Your DOMAIN System*. 1983.

[12] G. Artese, A. Trecroci, et al. "Calibration of a low cost MEMS INS sensor for an integrated navigation system". In: *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci* (2008), pp. 877–882.

[13] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Transactions on signal processing* 50.2 (2002), pp. 174–188.

[14] S. Aukstakalnis and D. Blatner. *Silicon Mirage; The Art and Science of Virtual Reality*. Berkeley, CA, USA: Peachpit Press, 1992.

[15] R. T. Azuma. "A survey of augmented reality". In: *Presence: Teleoperators & Virtual Environments* 6.4 (1997), pp. 355–385. DOI: 10.1162/pres.1997.6.4.355.

[16] R. T. Azuma. "Predictive tracking for augmented reality". PhD thesis. University of North Carolina at Chapel Hill, 1995.

[17] O. Bamodu and X. M. Ye. "Virtual reality and virtual reality system components". In: *Advanced materials research*. Vol. 765. Trans Tech Publ. 2013, pp. 1169–1172.

[18] J. B. Bancroft and G. Lachapelle. "Data fusion algorithms for multiple inertial measurement units". In: *Sensors* 11.7 (2011), pp. 6771–6798.

[19] J. B. Bancroft and G. Lachapelle. "Data Fusion Algorithms for Multiple Inertial Measurement Units". In: *Sensors* 11.7 (2011), pp. 6771–6798. DOI: 10.3390/s110706771. URL: http://www.mdpi.com/1424-8220/11/7/6771.

[20] Z. Bao, G. Lu, Y. Wang, and D. Tian. "A calibration method for misalignment angle of vehicle-mounted IMU". In: *Procedia-Social and Behavioral Sciences* 96 (2013), pp. 1853–1860.

[21] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

[22] G. Baratoff and S. Blanksteen. "Tracking devices". In: *Encyclopedia of Virtual Environments* (1993).

[23] J. A. Barnes, A. R. Chi, L. S. Cutler, D. J. Healey, D. B. Leeson, T. E. McGunigal, J. A. Mullen, W. L. Smith, R. L. Sydnor, R. F. Vessot, et al. "Characterization of frequency stability". In: *IEEE transactions on instrumentation and measurement* 2 (1971), pp. 105–120.

[24] M. Bauer. "Tracking Errors in Augmented Reality". PhD thesis. Technische Universität München, 2007.

[25] T. Bayes. "An essay towards solving a problem in the doctrine of chances". In: *Philosophical Transactions of the Royal Society of London* 53 (1763), pp. 370–418.

[26] J. K. Bekkeng. "Calibration of a novel MEMS inertial reference unit". In: *IEEE Transactions on instrumentation and measurement* 58.6 (2008), pp. 1967–1974.

[27] J. Belzer, A. Holzman, and A. Kent. *Encyclopedia of Computer Science and Technology*. Vol. 3. Marcel Dekker, Inc., 1976.

[28] Y. Bengio. "Learning Deep Architectures for AI". In: *Freescale Semiconductor* 2 (2009), pp. 1–127.

[29] Y. Bengio, P. Simard, and P. Frasconi. "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: *Freescale Semiconductor* 5 (1994), pp. 1–127.

[30] L. P. Berg and J. M. Vance. "Industry use of virtual reality in product design and manufacturing: a survey". In: *Virtual reality* 21.1 (2017), pp. 1–17.

[31] N. Bergman. "Recursive Bayesian estimation: Navigation and tracking applications". PhD thesis. Linköping University, 1999.

[32] S. Bhatt. *5 Things You Need to Know about Reinforcement Learning*. URL: https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html.

[33] G. Bishop, G. Welch, and B. D. Allen. "Tracking: Beyond 15 minutes of thought". In: *SIGGRAPH Course Pack* 11 (2001).

[34] L. Yi-Bo, K. Shao-Peng, Q. Zhi-Hua, and Z. Qiong. "Development actuality and application of registration technology in augmented reality". In: *2008 international symposium on computational intelligence and design*. Vol. 2. IEEE. 2008, pp. 69–74.

[35] R. Bos, S. De Waele, and P. M. Broersen. "Autoregressive spectral estimation by application of the Burg algorithm to irregularly sampled data". In: *IEEE Transactions on Instrumentation and Measurement* 51.6 (2002), pp. 1289–1294.

[36] N. Bostrom. *What happens when our computers get smarter than we are?* 2015. URL: https://www.ted.com/talks/nick_bostrom_what_happens_when_our_computers_get_smarter_than_we_are.

[37] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. San Francisco, Holden-Day, 1970.

[38] D. Brewster. *The Stereoscope; Its History, Theory and Construction, with Its Application to the Fine and Useful Arts and to Education, Etc*. John Murray, 1856.

[39] P. J. Brockwell and R. A. Davis. *Introduction to time series and forecasting*. Springer, 2016.

[40] N. A. Carlson. "Federated square root filter for decentralized parallel processors". In: *IEEE Transactions on Aerospace and Electronic Systems* 26.3 (1990), pp. 517–525.

[41] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic. "Augmented reality technologies, systems and applications". In: *Multimedia tools and applications* 51.1 (2011), pp. 341–377. DOI: 10.1007/s11042-010-0660-6.

[42] G. Carter. "Time delay estimation for passive sonar signal processing". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.3 (1981), pp. 463–470.

[43] J. Catzel. *Is your HMI ergonomically correct?* 2004. URL: https://www.controleng.com/articles/is-your-hmi-ergonomically-correct/.

[44] T. Caudell and D. Mizell. "Augmented reality: An application of heads-up display technology to manual manufacturing processes". In: *Hawaii international conference on system sciences*. 1992, pp. 659–669.

[45] C. Chatfield. *The analysis of time series. 5th Edn London*. New York, Chapman and Hall, 1996.

[46] C. Chatfield. "Model uncertainty, data mining and statistical inference". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 158.3 (1995), pp. 419–444.

[47] Z. Chen et al. “Bayesian filtering: From Kalman filters to particle filters, and beyond”. In: *Statistics* 182.1 (2003), pp. 1–69.

[48] C. M. Cheuk, T. K. Lau, K. W. Lin, and Y. Liu. “Automatic calibration for inertial measurement unit”. In: *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE. 2012, pp. 1341–1346.

[49] K.-Y. Choi, S.-A. Jang, and Y.-H. Kim. “Calibration of inertial measurement units using pendulum motion”. In: *International Journal of Aeronautical and Space Sciences* 11.3 (2010), pp. 234–239.

[50] K. H. Chon and R. J. Cohen. “Linear and nonlinear ARMA model parameter estimation using an artificial neural network”. In: *IEEE transactions on biomedical engineering* 44.3 (1997), pp. 168–174.

[51] R. D. Christ and R. L. Wernli. “Chapter 17 - Navigational Sensors”. In: *The ROV Manual (Second Edition)*. Oxford: Butterworth-Heinemann, 2014, pp. 453–475. DOI: https://doi.org/10.1016/B978-0-08-098288-5.00017-8.

[52] Computer History Museum. *ENIAC*. URL: https://www.computerhistory.org/revolution/birth-of-the-computer/4/78.

[53] S. Condino, G. Turini, P. D. Parchi, R. M. Viglialoro, N. Piolanti, M. Gesi, M. Ferrari, and V. Ferrari. “How to build a patient-specific hybrid simulator for Orthopaedic open surgery: benefits and limits of mixed-reality using the Microsoft HoloLens”. In: *Journal of healthcare engineering* 2018 (2018).

[54] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. “The CAVE: audio visual experience automatic virtual environment”. In: *Communications of the ACM* 35.6 (1992), pp. 64–73.

[55] I. Daubechies. *Ten lectures on wavelets*. Philadelphia, SIAM, 1992.

[56] A. G. De Sa and G. Zachmann. “Virtual reality as a tool for verification of assembly and maintenance processes”. In: *Computers & Graphics* 23.3 (1999), pp. 389–403.

[57] M. Dearborn. *Make Way For Holograms: New Mixed Reality Technology Meets Car Design As Ford Tests Microsoft Hololens Globally*. 2017. URL: https://media.ford.com/content/fordmedia/fna/us/en/news/2017/09/21/ford-tests-microsoft-hololens-globally.html.

[58] M. El-Diasty and S. Pagiatakis. “Calibration and stochastic modelling of inertial navigation sensor errors”. In: *Journal of Global Positioning Systems* 7.2 (2008), pp. 170–182.

[59] J. Ding, V. Tarokh, and Y. Yang. “Bridging AIC and BIC: a new criterion for autoregression”. In: *IEEE Transactions on Information Theory* 64.6 (2017), pp. 4024–4043.

[60] Z. Ding, H. Cai, and H. Yang. “An improved multi-position calibration method for low cost micro-electro mechanical systems inertial measurement units”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 229.10 (2015), pp. 1919–1930.

[61] Z. Dong, G. Zhang, Y. Luo, C. C. Tsang, G. Shi, S. Y. Kwok, W. J. Li, P. H. Leong, and M. Y. Wong. "A calibration method for MEMS inertial sensors based on optical tracking". In: *2007 2nd IEEE International Conference on Nano/Micro Engineered and Molecular Systems*. IEEE. 2007, pp. 542–547.

[62] A. Doucet, S. Godsill, and C. Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and computing* 10.3 (2000), pp. 197–208.

[63] K. Eckenhoff, P. Geneva, and G. Huang. "Mimc-vins: A versatile and resilient multi-imu multi-camera visual-inertial navigation system". In: *IEEE Transactions on Robotics* (2021).

[64] A. Edelstein. "Advances in magnetometry". In: *Journal of Physics: Condensed Matter* 19.16 (2007), p. 165217.

[65] C. Eichhorn, A. Jadid, D. A. Plecher, S. Weber, G. Klinker, and Y. Itoh. "Catching the Drone-A Tangible Augmented Reality Game in Superhuman Sports". In: *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2020, pp. 24–29.

[66] W. Elmenreich. "An Introduction to Sensor Fusion". In: *Research Report 47/2001* (2002). DOI: https://doi.org/10.1073/pnas.1201800109.

[67] D. C. Engelbart. "Augmenting human intellect: A conceptual framework". In: *Menlo Park, CA* (1962).

[68] D. C. Engelbart. *Xy position indicator for a display system*. US Patent 3,541,541. 1970.

[69] W. K. English, D. C. Engelbart, and M. L. Berman. "Display-selection techniques for text manipulation". In: *IEEE Transactions on Human Factors in Electronics* 1 (1967), pp. 5–15.

[70] G. Eshel. *The Yule Walker Equations for the AR Coefficients*. 2015. URL: http://www-stat.wharton.upenn.edu/%5C~steele/Courses/956/Resource/YWSourceFiles/YW-Eshel.pdf.

[71] L. Euler. "Formulae generales pro translatione quacunque corporum rigidorum". In: *Novi Commentarii academiae scientiarum Petropolitanae* (1776), pp. 189–207.

[72] Extend3D. *Werklicht Pro L*. URL: https://www.extend3d.de/werklichtpro.php.

[73] K. Faceli, A. C. L. de Carvalho, and S. O. Rezende. "Experiments on machine learning techniques for sensor fusion". In: *Proceedings Fourth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2001*. IEEE. 2001, pp. 395–399.

[74] A. S. Al-Fahoum and M. S. Abadir. "Design of a Modified Madgwick Filter for Quaternion-Based Orientation Estimation Using AHRS". In: *Int. J. Comput. Electrical Eng.* 10.3 (2018), pp. 174–186.

[75] D. Feng, C. Wang, C. He, Y. Zhuang, and X.-G. Xia. "Kalman-filter-based integration of IMU and UWB for high-accuracy indoor positioning and navigation". In: *IEEE Internet of Things Journal* 7.4 (2020), pp. 3133–3146.

[76] J. Ferguson. *Calibration of Deterministic IMU Errors*. Tech. rep. Embry-Riddle Aeronautical University, 2015.

[77] V. Ferrari, G. Klinker, and F. Cutolo. "Augmented Reality in Healthcare". In: *Journal of healthcare engineering* 2019 (2019).

[78] F. Ferraris, U. Grimaldi, and M. Parvis. "Procedure for effortless in-field calibration of three-axis rate gyros and accelerometers". In: *Sensors and Materials* 7 (1995), pp. 311–311.

[79] J. Filliben and A. Heckert. "Exploratory Data Analysis". In: *NIST/SEMATECH e-Handbook of Statistical Methods*. 2003. DOI: https://doi.org/10.18434/M32189.

[80] P. Fischer, R. Daniel, and K. V. Siva. "Specification and design of input devices for teleoperation". In: *Proceedings., IEEE International Conference on Robotics and Automation*. May 1990, 540–545 vol.1. DOI: 10.1109/ROBOT.1990.126036.

[81] M. Fleps, E. Mair, O. Ruepp, M. Suppa, and D. Burschka. "Optimization based IMU camera calibration". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 3297–3304.

[82] C. A. Fowler. "Comments on the Cost and Performance of Military Systems". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-15.1 (1979), pp. 2–10. DOI: 10.1109/TAES.1979.308790.

[83] J. Frankenfield. *Artificial Intelligince (AI)*. URL: https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp.

[84] B. Friedlander and B. Porat. "The modified Yule-Walker method of ARMA spectral estimation". In: *IEEE Transactions on Aerospace and Electronic Systems* 2 (1984), pp. 158–173.

[85] P. Furgale, J. Rehder, and R. Siegwart. "Unified temporal and spatial calibration for multi-sensor systems". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1280–1286.

[86] D. Galar and U. Kumar. "Chapter 1 - Sensors and Data Acquisition". In: *eMaintenance*. Ed. by D. Galar and U. Kumar. Academic Press, 2017, pp. 1–72.

[87] F. Galton. "Presidential address, section H, anthropology". In: *Report of the British Association for the Advancement of Science* 55 (1885), pp. 1206–14.

[88] Y. Gao, E. Krakiwsky, M. Abousalem, and J. McLellan. "Comparison and analysis of centralized, decentralized, and federated filters". In: *Navigation* 40.1 (1993), pp. 69–86.

[89] Y. Gao, L. Guan, and T. Wang. "Optimal artificial fish swarm algorithm for the field calibration on marine navigation". In: *Measurement* 50 (2014), pp. 297–304.

[90] Y. Gao, L. Guan, T. Wang, and Y. Sun. "A novel artificial fish swarm algorithm for recalibration of fiber optic gyroscope error parameters". In: *Sensors* 15.5 (2015), pp. 10547–10568.

[91] C. F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Frid. Perthes et J.H. Besser, 1809.

[92] C. F. Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*. 1821.

[93] F. A. Gers and J. Schmidhuber. "Recurrent Nets that Time and Count". In: *Neural Computation* 12 (2000), pp. 2451–2471.

[94] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12 (2000), pp. 2451–2471.

[95] J. M. Giron-Sierra. *Digital Signal Processing with Matlab Examples, Volume 3: Model-Based Actions and Sparse Representation*. Springer, 2016.

[96] P. Gloor. "Hans Berger on electroencephalography". In: *American Journal of EEG Technology* 9.1 (1969), pp. 1–8.

[97] R. H. Goddard. *Mechanism for directing flight*. US1879187A. 1932.

[98] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016. URL: http://www.deeplearningbook.org.

[99] Google. *Google Colaboratory*. URL: https://colab.research.google.com/.

[100] A. Graves and J. Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks* 18.5-6 (2005), pp. 602–610.

[101] J. Grudin. "From tool to partner: The evolution of human-computer interaction". In: *Synthesis Lectures on Human-Centered Interaction* 10.1 (2017), pp. i–183.

[102] J. Gu, M. Meng, A. Cook, and M. G. Faulkner. "Micro sensor based eye movement detection and neural network based sensor fusion and fault detection and recovery". In: *Canadian Conference on Electrical and Computer Engineering* 1 (2000), pp. 518–522.

[103] P. Gu, Z. Jing, and L. Wu. "Adaptive fading factor unscented Kalman filter with application to target tracking". In: *Aerospace Systems* (2020), pp. 1–6.

[104] D. Gupta. *Fundamentals of Deep Learning – Activation Functions and When to Use Them*. URL: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/.

[105] W. Guthrie, J. Filliben, and A. Heckert. "Process Modeling". In: *NIST/SEMATECH e-Handbook of Statistical Methods*. 2003.

[106] S. Ha and S. Choi. "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, pp. 381–388.

[107] Halfbrick. *Fruit Ninja*. URL: https://halfbrick.com/our-games/fruit-ninja/.

[108] Halfbrick. *Fruit Ninja VR*. URL: https://halfbrick.com/our-games/fruit-ninja-vr/.

[109] W. R. Hamilton. "On quaternions; or on a new system of imaginaries in algebra". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 25.163 (1844), pp. 10–13.

[110] A. Harrison and P. Newman. "TICSync: Knowing when things happened". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 356–363.

[111] D. Hartmann and H. Van der Auweraer. "Digital Twins". In: *arXiv preprint arXiv:2001.09747* (2020).

[112] A. G. Hayal. "Static calibration of the tactical grade inertial measurement units". PhD thesis. The Ohio State University, 2010.

[113] C. He, P. Kazanzides, H. T. Sen, S. Kim, and Y. Liu. "An inertial and optical sensor fusion approach for six degree-of-freedom pose estimation". In: *Sensors* 15.7 (2015), pp. 16448–16465.

[114] M. L. Heilig. *Sensorama simulator*. US Patent 3,050,870. 1962.

[115] F. J. Hellings. *Application of extended Kalman filtering to a dynamic laboratory calibration of an inertial navigation system*. Tech. rep. Aerospace Corp el Segundo ca Engineering Science Operations, 1973.

[116] G. Hendeby, R. Karlsson, and F. Gustafsson. "Particle filtering: the need for speed". In: *EURASIP Journal on Advances in Signal processing* 2010 (2010), pp. 1–9.

[117] Y. Ho and R. Lee. "A Bayesian approach to problems in stochastic estimation and control". In: *IEEE transactions on automatic control* 9.4 (1964), pp. 333–339.

[118] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[119] J. D. Hol, F. Dijkstra, H. Luinge, and T. B. Schon. "Tightly coupled UWB/IMU pose estimation". In: *2009 IEEE international conference on ultra-wideband*. IEEE. 2009, pp. 688–692.

[120] J. D. Hol, T. B. Schön, and F. Gustafsson. "Relative pose calibration of a spherical camera and an IMU". In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2008, pp. 21–24.

[121] J. D. Hol, T. B. Schön, and F. Gustafsson. "Modeling and calibration of inertial and vision sensors". In: *The international journal of robotics research* 29.2-3 (2010), pp. 231–244.

[122] G. Hu, S. Gao, Y. Zhong, B. Gao, and A. Subic. "Modified federated Kalman filter for INS/GNSS/CNS integration". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 230.1 (2016), pp. 30–44.

[123] M. Huber, M. Schlegel, and G. Klinker. "Application of time-delay estimation to mixed reality multisensor tracking". In: *Journal of Virtual Reality and Broadcasting* 11.3 (2014).

[124] M. Hwangbo and T. Kanade. "Factorization-based calibration method for MEMS inertial measurement unit". In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 1306–1311.

[125] "IEEE Standard for Inertial Sensor Terminology". In: *IEEE Std 528-2019* (2019), pp. 1–35. DOI: 10.1109/IEEESTD.2019.8863799.

[126] "IEEE Standard Specification Format Guide and Test Procedure for Linear, Single-Axis, Non-Gyroscopic Accelerometers". In: *IEEE Std 1293-1998* (1999), pp. 1–252. DOI: 10.1109/IEEESTD.1999.89857.

[127] "IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros". In: *IEEE Std 952-1997* (1998), pp. 1–84. DOI: 10.1109/IEEESTD.1998.86153.

[128] G. Jacovitti and G. Scarano. "Discrete time techniques for time delay estimation". In: *IEEE Transactions on signal processing* 41.2 (1993), pp. 525–533.

[129] A. Jadid, M. Koplin, S. Siegert, M. Hering-Bertram, V. Paelke, T. Teschke, and H. Eirund. "Express Yourself/City—Smart Participation Culture Technologies". In: *Smart Cities in the Mediterranean*. Springer, 2017, pp. 175–194.

[130] A. Jadid, L. Rudolph, F. Pankratz, and G. Klinker. "Utilizing Multiple Calibrated IMUs for Enhanced Mixed Reality Tracking". In: *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE. 2019, pp. 384–386. DOI: 10.1109/ISMAR-Adjunct.2019.000-5.

[131] A. Jadid, L. Rudolph, F. Pankratz, K. Wu, P. Wang, and G. Klinker. *MultiIMU: Fusing Multiple Calibrated IMUs For Enhanced Mixed Reality Tracking*. Technical report. Technical University Munich, 2019.

[132] J. Jankowski and C. Sucksdorff. *Guide for magnetic measurements and observatory practice*. International Association of Geomagnetism and Aeronomy Warsaw, 1996, p. 235.

[133] A. H. Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[134] Y.-B. Jia. *Recursive Least Squares Estimation*. Tech. rep. 2015.

[135] E. S. Jones and S. Soatto. "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach". In: *The International Journal of Robotics Research* 30.4 (2011), pp. 407–430.

[136] S. J. Julier and J. K. Uhlmann. "Fusion of time delayed measurements with uncertain time delays". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 4028–4033.

[137] D. Jurman, M. Jankovec, R. Kamnik, and M. Topič. "Calibration and data fusion solution for the miniature attitude and heading reference system". In: *Sensors and Actuators A: Physical* 138.2 (2007), pp. 411–420.

[138] R. E. Kalman and R. S. Bucy. "New results in linear filtering and prediction theory". In: *Journal of Basic Engineering* (1961).

[139] R. E. Kalman. "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* (1960), pp. 34–45.

[140] A. Kaur. *Neural Network & Hyperparameter Tuning In A Nutshell*. URL: https://medium.com/@arshdeepkaur/neural-network-hyperparameter-tuning-in-a-nutshell-8c2b9a14a2c0.

[141] P. Keitler. "Management of Tracking". Dissertation. Munich: Technical University of Munich, 2011. URL: https://mediatum.ub.tum.de/node?id=1007293.

[142] J. Kelly, N. Roy, and G. S. Sukhatme. "Determining the time delay between inertial and visual sensor measurements". In: *IEEE Transactions on Robotics* 30.6 (2014), pp. 1514–1523.

[143] Keras. *Keras Conv1D: Working with 1D Convolutional Neural Networks in Keras*. URL: https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras.

[144] Z. H. Khan. "Multiple IMU Sensor Fusion using Convolutional Neural Network for predicting an object's position and orientation". Master's thesis. Technische Universität München, 2020.

[145] Y. Khazanov. *MEMS accelerometer device*. US Patent 8,700,353. 2014.

[146] G. Kitagawa. "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models". In: *Journal of computational and graphical statistics* 5.1 (1996), pp. 1–25.

[147] R. A. Kitzerow. *Design, development, analysis, and laboratory test results of a Kalman filter system-level IMU calibration technique*. Tech. rep. AIR FORCE AVIONICS LAB WRIGHT-PATTERSONAFB OH, 1977.

[148] G. Klinker. "Ubiquitous, Dynamic Inclusion and Fusion of Tracking Data from Various Sources for Mobile AR Applications in "AR-ready Environments"". In: *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications, Volume 1, Lisbon, Portugal, 5-8 January, 2014*. 2014, IS–11.

[149] G. Klinker, H. Najafi, T. Sielhorst, F. Sturm, F. Echtler, M. Isik, W. Wein, and C. Trübswetter. "FixIt: An Approach towards Assisting Workers in Diagnosing Machine Malfunctions." In: *MIXER*. Citeseer. 2004.

[150] C. Knapp and G. Carter. "The generalized correlation method for estimation of time delay". In: *IEEE transactions on acoustics, speech, and signal processing* 24.4 (1976), pp. 320–327.

[151] J. Koutnıék, K. Greff, F. Gomez, and J. Schmidhuber. "A Clockwork RNN". In: *Neural Computation* 12 (2014), pp. 2451–2471. DOI: 10.1002/andp.19053221004.

[152] C. Krebs. "Generic IMU-camera calibration algorithm: Influence of IMU-axis on each other". In: *Autonomous Systems Lab, ETH Zurich, Tech. Rep* (2012).

[153] M. V. Kulikova and G. Y. Kulikov. "Numerical methods for nonlinear filtering of signals and measurements". In: *Computational Technologies* 21.4 (2016), pp. 64–98.

[154] R. Kuni, Y. Prathivadi, J. Wu, T. R. Bennett, and R. Jafari. "Exploration of interactions detectable by wearable IMU sensors". In: *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE. 2015, pp. 1–6.

[155] P. Lang, A. Kusej, A. Pinz, and G. Brasseur. "Inertial tracking for mobile augmented reality". In: *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No. 00CH37276)*. Vol. 2. IEEE. 2002, pp. 1583–1587.

[156] P. Lang and A. Pinz. "Calibration of hybrid vision/inertial tracking systems". In: *InerVis, Barcelona, Spain* (2005).

[157] P. Laplace. *Theorie Analytique des Probabilité*. Courcier, 1812.

[158] A. G. Leal-Junior, L. Vargas-Valencia, W. M. dos Santos, F. B. Schneider, A. A. Siqueira, M. J. Pontes, and A. Frizera. "POF-IMU sensor system: A fusion between inertial measurement units and POF sensors for low-cost and highly reliable systems". In: *Optical Fiber Technology* 43 (2018), pp. 82–89.

[159] D. Lee, S. Lee, S. Park, and S. Ko. "Test and error parameter estimation for MEMS - based low cost IMU calibration". In: *Int. J. Precis. Eng. Manuf. 12* 12 (2011), pp. 597–603.

[160] H. Lee, T. H. Jung, M. C. tom Dieck, and N. Chung. "Experiencing immersive virtual reality in museums". In: *Information & Management* 57.5 (2020), p. 103229.

[161] T. S. Lee. "Theory and application of adaptive fading memory Kalman filters". In: *IEEE transactions on circuits and systems* 35.4 (1988), pp. 474–477.

[162] A. M. Legendre. "Nouvelles méthodes pour la détermination des orbites des comètes". In: *Firmin Didot, Paris* (1805).

[163] M. C. Leu, H. A. ElMaraghy, A. Y. Nee, S. K. Ong, M. Lanzetta, M. Putz, W. Zhu, and A. Bernard. "CAD model based virtual assembly simulation, planning and training". In: *CIRP Annals* 62.2 (2013), pp. 799–822.

[164] J. Li and J. Fang. "Not Fully Overlapping Allan Variance and Total Variance for Inertial Sensor Stochastic Error Analysis". In: *IEEE Transactions on Instrumentation and Measurement* 62.10 (Oct. 2013), pp. 2659–2672. DOI: 10.1109/TIM.2013.2258769. URL: http://ieeexplore.ieee.org/document/6576249/.

[165] M. Li and A. I. Mourikis. "Online temporal calibration for camera–IMU systems: Theory and algorithms". In: *The International Journal of Robotics Research* 33.7 (2014), pp. 947–964.

[166] M. Li, H. Yu, X. Zheng, and A. I. Mourikis. "High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 409–416.

[167] Q. Li and D. Rus. "Global clock synchronization in sensor networks". In: *IEEE Transactions on computers* 55.2 (2006), pp. 214–226.

[168] X.-l. Li. "An optimizing method based on autonomous animats: fish-swarm algorithm". In: *Systems Engineering-Theory & Practice* 22.11 (2002), pp. 32–38.

[169] Y. Li, X. Niu, Q. Zhang, H. Zhang, and C. Shi. "An in situ hand calibration method using a pseudo-observation scheme for low-end inertial measurement units". In: *Measurement Science and Technology* 23.10 (2012), p. 105104.

[170] J. Liang, L. Zhang, L. Wang, Y. Dong, and T. Ueda. "Flip Chip Bonding of a Quartz MEMS-Based Vibrating Beam Accelerometer". In: *Sensors (Basel, Switzerland)* 15 (2015), pp. 22049–22059.

[171] C. Lindner, A. Rienow, and C. Jürgens. "Augmented Reality applications as digital experiments for education–An example in the earth-moon system". In: *Acta Astronautica* 161 (2019), pp. 66–74.

[172] D. Linnell. *The SAA handbook: a practical approach to IBM's systems application architecture*. Addison Wesley Publishing Company, 1990.

[173] J. S. Liu and R. Chen. "Sequential Monte Carlo methods for dynamic systems". In: *Journal of the American statistical association* 93.443 (1998), pp. 1032–1044.

[174] J. Lobo and J. Dias. "Relative pose calibration between visual and inertial sensors". In: *The International Journal of Robotics Research* 26.6 (2007), pp. 561–575.

[175] C. Lundquist. "Sensor fusion for automotive applications". PhD thesis. Linköping University Electronic Press, 2011.

[176] M. S. Lvov and H. V. Popova. "Simulation technologies of virtual reality usage in the training of future ship navigators". In: *Kryvyi Rih State Pedagogic University Publishing Center* (2019).

[177] R. Madan, S. K. Singh, and N. Jain. "Signal filtering using discrete wavelet transform". In: *International journal of recent trends in engineering* 2.3 (2009), p. 96.

[178] S. Madgwick. "An efficient orientation filter for inertial and inertial/magnetic sensor arrays". In: *Report x-io and University of Bristol (UK)* 25 (2010), pp. 113–118.

[179] S. Madgwick, A. Harrison, and R. Vaidyanathan. "Estimation of IMU and MARG orientation using a gradient descent algorithm". In: *2011 IEEE international conference on rehabilitation robotics*. IEEE. 2011, pp. 1–7.

[180] R. Mahony, T. Hamel, and J.-M. Pflimlin. "Nonlinear complementary filters on the special orthogonal group". In: *IEEE Transactions on automatic control* 53.5 (2008), pp. 1203–1218.

[181] E. Mair, M. Fleps, M. Suppa, and D. Burschka. "Spatio-temporal initialization for IMU to camera registration". In: *2011 IEEE International Conference on Robotics and Biomimetics*. IEEE. 2011, pp. 557–564.

[182] A. Makdissi, F. Vernotte, and E. De Clercq. "Stability variances: a filter approach". In: *IEEE transactions on ultrasonics, ferroelectrics, and frequency control* 57.5 (2010), pp. 1011–1028.

[183] S. G. Mallat. "A theory of multiresolution signal decomposition: the wavelet representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (1989), pp. 674–693.

[184] P. Matisko and V. Havlena. "Noise covariance estimation for Kalman filter tuning using Bayesian approach and Monte Carlo". In: *International Journal of Adaptive Control and Signal Processing* 27.11 (2013), pp. 957–973.

[185] C. Mehr and J. McFadden. "Certain Properties of Gaussian Processes and Their First-Passage Times". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 27.3 (1965), pp. 505–522.

[186] M. Melero, A. Hou, E. Cheng, A. Tayade, S. C. Lee, M. Unberath, and N. Navab. "Upbeat: augmented reality-guided dancing for prosthetic rehabilitation of upper limb amputees". In: *Journal of healthcare engineering* 2019 (2019).

[187] R. v. d. Merwe and E. A. Wan. "Sigma-point Kalman filters for integrated navigation". In: *Proceedings of the 60th annual meeting of the institute of navigation (2004)*. 2004, pp. 641–654.

[188] Microsoft. *Hololens*. 2020. URL: www.microsoft.com/hololens/.

[189] M. Mihelj. "Inverse kinematics of human arm based on multisensor data integration". In: *Journal of Intelligent and Robotic Systems* 47.2 (2006), pp. 139–153.

[190] P. Milgram and F. Kishino. "A taxonomy of mixed reality visual displays". In: *IEICE Transactions on Information and Systems* 77.12 (1994), pp. 1321–1329.

[191] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. "Augmented reality: a class of displays on the reality-virtuality continuum". In: *Telemanipulator and Telepresence Technologies* 2351.December 2013 (1995), pp. 282–292.

[192] D. Mills, J. Martin, J. Burbank, and W. Kasch. *Network time protocol version 4: Protocol and algorithms specification*. RFC 5905 (Proposed Standard), 2010.

[193] F. M. Mirzaei and S. I. Roumeliotis. "A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation". In: *IEEE transactions on robotics* 24.5 (2008), pp. 1143–1156.

[194] M. Mishra and M. Srivastava. "A View of Artificial Neural Network". In: *IEEE International Conference on Advances in Engineering and Technology Research (ICAETR - 2014)* (2014).

[195] A. H. Mohamed and K. P. Schwarz. "Adaptive Kalman filtering for INS/GPS". In: *Journal of geodesy* 73.4 (1999), pp. 193–203.

[196] S. Münzner, P. Schmidt, A. Reiss, M. Hanselmann, R. Stiefelhagen, and R. Dürichen. "CNN-based sensor fusion techniques for multimodal human activity recognition". In: *Proceedings - International Symposium on Wearable Computers, ISWC* Part F1305 (2017), pp. 158–165. DOI: 10.1145/3123021.3123046.

[197] NASA. *Hubble Space Telescope Servicing Mission 4 Gyroscopes*. URL: https://www.nasa.gov/mission_pages/hubble/servicing/SM4/main/Gyro_FS_HTML.html#:~:text=Gas%2Dbearing%20gyros%20are%20the,small%20movements%20of%20the%20telescope..

[198] S. Nassar, K.-P. Schwarz, and N. El-Sheimy. "INS and INS/GPS accuracy improvement using autoregressive (AR) modeling of INS sensor errors". In: *Proceedings of the 2004 national technical meeting of the institute of navigation*. 2004, pp. 936–944.

[199] A. Ng. "Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78. DOI: 10.1145/1015330.1015435.

[200] J. Nielsen. *Voice Interfaces: Assessing the Potential*. 2003. URL: https://www.nngroup.com/articles/voice-interfaces-assessing-the-potential/.

[201] J. Nikolic, M. Burri, I. Gilitschenski, J. Nieto, and R. Siegwart. "Non-parametric extrinsic and intrinsic calibration of visual-inertial sensor systems". In: *IEEE Sensors Journal* 16.13 (2016), pp. 5433–5443.

[202] X.-J. Niu, Z.-Y. Gao, R. Zhang, and Z.-Y. Chen. "Satellite TV Antenna Attitude Stabilization System Based on Micromachined Inertial Sensors". In: *J. Chin. Inertial Technol.* 5 (2002), p. 002.

[203] S. Nolle and G. Klinker. "Augmented reality as a comparison tool in automotive industry". In: *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2006, pp. 249–250.

[204] Novatel bulletin. *IMU Errors and Their Effects*. 2015.

[205] J. Oehm. "Development of an Input Device Based on a Microcontroller with an Inertial Measurement Unit to Interact in Three-Dimensional Space". Bachelor's Thesis (advisor Adnane Jadid). Technical University of Munich, 2020.

[206] H. M. Ólafsdóttir. "Virtual courtroom". PhD thesis. Reykjavıék University, 2019.

[207] C. Olah. *Understanding LSTM networks*. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[208] E. Olson. "A passive solution to the sensor synchronization problem". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 1059–1064.

[209] M. O. Onyesolu and F. U. Eze. "Understanding virtual reality technology: advances and applications". In: *Adv. Comput. Sci. Eng* (2011), pp. 53–70.

[210] A. Oppermann. *Artificial Intelligence vs. Machine Learning vs. Deep Learning*. 2019. URL: https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac.

[211] Oracle. *Types of Machine Learning*. URL: https://blogs.oracle.com/datascience/types-of-machine-learning-and-top-10-algorithms-everyone-should-know-v2.

[212] R. L. Page. "Brief history of flight simulation". In: *SimTecT 2000 Proceedings* (2000), pp. 11–17.

[213] F. Palmas, D. Labode, D. A. Plecher, and G. Klinker. "Comparison of a Gamified and Non-Gamified Virtual Reality Training Assembly Task". In: *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. 2019, pp. 1–8. DOI: 10.1109/VS-Games.2019.8864583.

[214] F. Palmas, J. Cichor, D. A. Plecher, and G. Klinker. "Acceptance and Effectiveness of a Virtual Reality Public Speaking Training". In: *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE. 2019, pp. 363–371.

[215] F. Palmas and G. Klinker. "Defining Extended Reality Training: A Long-Term Definition for All Industries". In: *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)*. IEEE. 2020, pp. 322–324.

[216] V. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo, and C. E. Campanella. "Gyroscope technology and applications: A review in the industrial perspective". In: *Sensors* 17.10 (2017), p. 2284.

[217] A. Patron-Perez, S. Lovegrove, and G. Sibley. "A spline-based trajectory representation for sensor fusion and rolling shutter cameras". In: *International Journal of Computer Vision* 113.3 (2015), pp. 208–219.

[218] M. Paulich, M. Schepers, N. Rudigkeit, and G. Bellusci. "Xsens MTw Awinda: Miniature wireless inertial-magnetic motion tracker for highly accurate 3D kinematic applications". In: *Xsens: Enschede, The Netherlands* (2018).

[219] K. Pearson, G. Yule, N. Blanchard, and A. Lee. "The Law of Ancestral Heredity". In: *Biometrica* 2.2 (1903), pp. 211–236.

[220] A. Pinz, M. Brandner, H. Ganster, A. Kusej, P. Lang, and M. Ribo. "Hybrid tracking for augmented reality". In: *Ögai Journal* 21.1 (2002), pp. 17–24.

[221] D. A. Plecher, M. Wandinger, and G. Klinker. "Mixed reality for cultural heritage". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 1618–1622.

[222] S. Poddar, V. Kumar, and A. Kumar. "A comprehensive overview of inertial sensor calibration techniques". In: *Journal of Dynamic Systems, Measurement, and Control* 139.1 (2017).

[223] J. Prins. "Process or Product Monitoring and Control". In: *NIST/SEMATECH e-Handbook of Statistical Methods*. 2003. DOI: https://doi.org/10.18434/M32189.

[224] D. Pustka, J. Willneff, O. Wenisch, P. Lukewille, K. Achatz, P. Keitler, and G. Klinker. "Determining the point of minimum error for 6DOF pose uncertainty representation". In: *9th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2010, Seoul, Korea, 13-16 October 2010*. 2010, pp. 37–45. DOI: 10.1109/ISMAR.2010.5643548.

[225] J. Radianti, T. A. Majchrzak, J. Fromm, and I. Wohlgenannt. "A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda". In: *Computers & Education* 147 (2020), p. 103778.

[226] H. M. A. Rahman. "Predicting Object Position and Orientation Using Recurrent Neural Network for Sensor Fusion of Multiple IMUs". Master's thesis. Technische Universität München, 2020.

[227] M. R. Rajamani and J. B. Rawlings. "Estimation of the disturbance structure from data using semidefinite programming and optimal weighting". In: *Automatica* 45.1 (2009), pp. 142–148.

[228] P. Ramachandran, B. Zoph, and Q. V. Le. "Swish: a self-gated activation function". In: *arXiv preprint arXiv:1710.05941* 7 (2017), p. 1.

[229] R. Ramalingam, G. Anitha, and J. Shanmugam. "Microelectromechnical Systems Inertial Measurement Unit Error Modelling and Error Analysis for Low-cost Strapdown Inertial Navigation System". In: *Defence Science Journal* 59.6 (2009), pp. 650–658. DOI: 10.14429/dsj.59.1571. URL: http://publications.drdo.gov.in/ojs/index.php/dsj/article/view/1571.

[230] K. Rana. *Pooling Layer - Short and Simple*. 2020. URL: https://medium.com/ai-in-plain-english/pooling-layer-beginner-to-intermediate-fa0dbdce80eb.

[231] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart. "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4304–4311.

[232] Reichelt. *Rollei Smartphone Tripod*. URL: https://www.reichelt.com/de/en/smartphone-tripod-traveler-rollei-22638-p212410.html.

[233] T. Reicher, A. Mac Williams, B. Brugge, and G. Klinker. "Results of a study on software architectures for augmented reality systems". In: *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.* IEEE. 2003, pp. 274–275.

[234] E. L. Renk, W. Collins, M. Rizzo, F. Lee, and D. S. Bernstein. "Optimization-based calibration of a triaxial accelerometer-magnetometer". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 1957–1962.

[235] J. P. Rolland, L. Davis, Y. Baillot, et al. "A survey of tracking technology for virtual environments". In: *Fundamentals of wearable computers and augmented reality* 1.1 (2001), pp. 67–112.

[236] T. Rose, C. S. Nam, and K. B. Chen. "Immersion of virtual reality for rehabilitation-Review". In: *Applied ergonomics* 69 (2018), pp. 153–161.

[237] R. J. Rossi. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

[238] F. Rousseau, P. Hellier, and C. Barillot. "A novel temporal calibration method for 3-D ultrasound". In: *IEEE transactions on medical imaging* 25.8 (2006), pp. 1108–1112.

[239] J. Rutman. "Characterization of phase and frequency instabilities in precision frequency sources: Fifteen years of progress". In: *Proceedings of the IEEE* 66.9 (1978), pp. 1048–1075.

[240] K. Sabarinathan, N. Kanagasabapathy, V. A. Kumar, P. Rishikesh, R. Priyadharshan, and A. Abirami. "Machine Maintenance Using Augmented Reality". In: *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*. IEEE. 2018, pp. 613–618.

[241] K. Saktaweekulkit and T. Maneewarn. "Motion classification using IMU for human-robot interaction". In: *ICCAS 2010*. IEEE. 2010, pp. 2295–2299.

[242] N. J. Sanket. *Madgwick Filter*. URL: https://nitinjsanket.github.io/tutorials/attitudeest/madgwick.html.

[243] G. Schall, D. Wagner, G. Reitmayr, E. Taichmann, M. Wieser, D. Schmalstieg, and B. Hofmann-Wellenhof. "Global pose estimation using multi-sensor fusion for outdoor augmented reality". In: *2009 8th ieee international symposium on mixed and augmented reality*. IEEE. 2009, pp. 153–162.

[244] M. Schuster and K. K. Paliwal. "Bidirectional Recurrent Neural Networks". In: *Freescale Semiconductor* 55 (1997), pp. 2673–2681.

[245] Search Enterprise AI. *Convolutional Neural Network*. URL: https://searchenterpriseai.techtarget.com/definition/convolutional-neural-network.

[246] A. Serrano, V. Sitzmann, J. Ruiz-Borau, G. Wetzstein, D. Gutierrez, and B. Masia. "Movie editing and cognitive event segmentation in virtual reality video". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–12.

[247] M. Al-Shabi. "Sigma-Point Filters in Robotic Applications". In: *Scientific Research Publishing* (2015).

[248] N. El-Sheimy, H. Hou, and X. Niu. "Analysis and Modeling of Inertial Sensors Using Allan Variance". In: *Instrumentation and Measurement, IEEE Transactions on* 57 (Feb. 2008), pp. 140–149. DOI: 10.1109/TIM.2007.908635.

[249] E.-H. Shin and N. El-Sheimy. "A new calibration method for strapdown inertial navigation systems". In: *Z. vermess* 127 (2002), pp. 1–10.

[250] A. Shintemirov, T. Taunyazov, B. Omarali, A. Nurbayeva, A. Kim, A. Bukeyev, and M. Rubagotti. "An Open-Source 7-DOF Wireless Human Arm Motion-Tracking System for Use in Robotics Research". In: *Sensors* 20.11 (2020), p. 3082.

[251] I. Skog and P. Händel. "Calibration of a MEMS inertial measurement unit". In: *XVII IMEKO World Congress*. 2006, pp. 1–6.

[252] G. Škulj, R. Vrabič, P. Podržaj, et al. "A Wearable IMU System for Flexible Teleoperation of a Collaborative Industrial Robot". In: *Sensors* 21.17 (2021), p. 5871.

[253] J. M. Slater and J. S. Acterman. *Gas bearing gyroscope*. US Patent 3,048,043. 1962.

[254] R. J. Solomonoff. "An inductive inference machine". In: *IRE Convention Record, Section on Information Theory*. Vol. 2. 1957, pp. 56–62.

[255] S. R. Sorko and M. Brunnhofer. "Potentials of augmented reality in training". In: *Procedia Manufacturing* 31 (2019), pp. 85–90.

[256] M. Speicher, B. D. Hall, and M. Nebeling. "What is Mixed Reality?" In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–15. DOI: 10.1145/3290605.3300767.

[257] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[258] R. Steffan, U. Schull, and T. Kuhlen. "Integration of virtual reality based assembly simulation into CAD/CAM environments". In: *IECON'98. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 98CH36200)*. Vol. 4. IEEE. 1998, pp. 2535–2537.

[259] S. Stein. "Frequency and Time – Their Measurement and Characterization". In: *Precision Frequency Control*. Academic, New York, 1985. Chap. 12, pp. 191–232.

[260] G. Strang and T. Nguyen. *Wavelets and filter banks*. SIAM, 1996.

[261] I. E. Sutherland. "A Head-mounted Three Dimensional Display". In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, 1968, pp. 757–764. DOI: 10.1145/1476589.1476686.

[262] P. Swerling. *First-order error propagation in a stagewise smoothing procedure for satellite observations*. RAND, 1959.

[263] Z. F. Syed, P. Aggarwal, C. Goodall, X. Niu, and N. El-Sheimy. "A new multi-position calibration method for MEMS inertial navigation systems". In: *Measurement science and technology* 18.7 (2007), p. 1897.

[264] W. Tao, Z.-H. Lai, M. C. Leu, and Z. Yin. "Manufacturing assembly simulations in virtual and augmented reality". In: *Augmented, Virtual, and Mixed Reality Applications in Advanced Manufacturing* (2019).

[265] D. Tedaldi, A. Pretto, and E. Menegatti. "A robust and easy to implement method for IMU calibration without external equipments". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014, pp. 3042–3049. DOI: 10.1109/ICRA.2014.6907297.

[266] H. G. Teklemariam, V. Kakati, and A. K. Das. "Application of VR Technology in Design Education". In: *DS 78: Proceedings of the 16th International conference on Engineering and Product Design Education (E&PDE14), Design Education and Human Technology Relations, University of Twente, The Netherlands, 04-05.09. 2014*. 2014.

[267] TensorFlow. *TensorBoard: TensorFlow's visualization toolkit*. URL: https://www.tensorflow.org/tensorboard.

[268] R. Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[269] D. Titterton, J. L. Weston, and J. Weston. *Strapdown inertial navigation technology*. Vol. 17. IET, 2004.

[270] F. Tungadi and L. Kleeman. "Time synchronisation and calibration of odometry and range sensors for high-speed mobile robot mapping". In: *Proc. Australasian Conference on Robotics and Automation*. 2008.

[271] S. Tzima, G. Styliaras, and A. Bassounas. "Augmented reality applications in education: Teachers point of view". In: *Education Sciences* 9.2 (2019), p. 99.

[272] J. K. Uhlmann. "Dynamic map building and localization: New theoretical foundations". PhD thesis. University of Oxford, 1995.

[273] F. Vázquez. "Deep learning made easy with deep cognition". In: *Retrieved April* 19 (2017), p. 2019. URL: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351.

[274] J. J. Vidal. "Toward direct brain-computer communication". In: *Annual review of Biophysics and Bioengineering* 2.1 (1973), pp. 157–180.

[275] J. J. Vidal. "Real-time detection of brain events in EEG". In: *Proceedings of the IEEE* 65.5 (1977), pp. 633–641.

[276] G. S. Von Itzstein, M. Billinghurst, R. T. Smith, and B. H. Thomas. *Augmented Reality Entertainment: Taking Gaming Out of the Box*. 2019.

[277] R. M. Voyles, J. D. Morrow, and P. K. Khosla. "The shape from motion approach to rapid and precise force/torque sensor calibration". In: *J. Dyn. Sys., Meas., Control* (1997).

[278] VRgineers. *XTAL*. URL: https://vrgineers.com/xtal/.

[279] J. Wahlström and I. Skog. "Fifteen Years of Progress at Zero Velocity: A Review". In: *IEEE Sensors Journal* (2020).

[280] P. L. Walter. "The history of the accelerometer". In: *Sound and vibration* 31.3 (1997), pp. 16–23.

[281] J. Wang, Q. Chang, G. Xiao, N. Wang, and S. Li. "Data driven production modeling and simulation of complex automobile general assembly plant". In: *Computers in industry* 62.7 (2011), pp. 765–775.

[282] L. Wang, C. Zhang, S. Gao, T. Wang, T. Lin, and X. Li. "Application of Fast Dynamic Allan Variance for the Characterization of FOGs-Based Measurement While Drilling". In: *Sensors* 16.12 (2016).

[283] E. W. Weisstein. *CRC concise encyclopedia of mathematics*. CRC press, 2002.

[284] E. W. Weisstein. *Convolution*. URL: https://mathworld.wolfram.com/Convolution.html.

[285] G. Welch and E. Foxlin. "Motion tracking: No silver bullet, but a respectable arsenal". In: *IEEE Computer graphics and Applications* 22.6 (2002), pp. 24–38.

[286] P. Whittle. *Hypothesis testing in time series analysis*. Vol. 4. Almqvist & Wiksells boktr., 1951.

[287] W. Winn. "A conceptual basis for educational applications of virtual reality". In: *Technical Publication R-93-9, Human Interface Technology Laboratory of the Washington Technology Center, Seattle: University of Washington* (1993).

[288] C. Woodford. *Accelerometers*. 2020. URL: https://www.explainthatstuff.com/accelerometers.html.

[289] Q. Xia, M. Rao, Y. Ying, and X. Shen. "Adaptive fading Kalman filter with an application". In: *Automatica* 30.8 (1994), pp. 1333–1338.

[290] Y. Xu, X. Zhu, and Y. Su. "A novel network calibration method for inertial measurement units". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 229.7 (2015), pp. 1336–1348.

[291] P. Yang, W. Wu, M. Moniri, and C. C. Chibelushi. "A Sensor-based SLAM Algorithm for Camera Tracking in Virtual Studio". In: *International Journal of Automation and Computing* 5.2 (2008), p. 152. URL: http://www.ijac.net/EN/abstract/article_992.shtml.

[292] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer. "Depth-Gated Recurrent Neural Networks". In: *Neural Computation* 12 (2015), pp. 2451–2471.

[293] S. You, U. Neumann, and R. Azuma. "Hybrid Inertial and Vision Tracking for Augmented Reality Registration". In: *IEEE Virtual Reality 1999 Conference, VR'99, Houston, Texas, USA, March 13-17, 1999, Proceedings*. 1999, p. 260. DOI: 10.1109/VR.1999.756960.

[294] G. Yule. "On the Theory of Correlation". In: *Journal of the Royal Statistical Society* 60.4 (1897), pp. 812–854.

[295] D. Zachariah and M. Jansson. "Joint calibration of an inertial measurement unit and coordinate transformation parameters using a monocular camera". In: *2010 International Conference on Indoor Positioning and Indoor Navigation*. IEEE. 2010, pp. 1–7.

[296] G. Zachmann. "Real-time and exact collision detection for interactive virtual prototyping". In: *Proc. of the 1997 ASME Design Engineering Technical Conferences*. 1997, pp. 14–17.

[297] G. Zachmann. "Virtual reality in assembly simulation-collision detection, simulation algorithms, and interaction techniques". PhD thesis. Zachmann, Gabriel, 2000.

[298] N. Zaitseva, T. Pogarskaia, O. Minevich, J. Shinder, and E. Bonhomme. *Simulation of aircraft assembly via ASRP software*. Tech. rep. SAE Technical Paper, 2019.

[299] M. Zaman and J. Illingworth. "Interval-based time synchronisation of sensor data in a mobile robot". In: *Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004.* IEEE. 2004, pp. 463–468.

[300] Z. Zeng, S. Liu, and L. Wang. "UWB/IMU integration approach with NLOS identification and mitigation". In: *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2018, pp. 1–6.

[301] H. Zhang, B. Lennox, P. R. Goulding, and Y. Wang. "Adaptive Information Sharing Factors in Federated Kalman Filtering". In: *IFAC Proceedings Volumes* 35.1 (2002), pp. 79–84.

[302] R. Zhang, H. Yang, F. Höflinger, and L. M. Reindl. "Adaptive zero velocity update based on velocity classification for pedestrian tracking". In: *IEEE Sensors journal* 17.7 (2017), pp. 2137–2145.

[303] Y. Zhao. *GPS/IMU integrated system for land vehicle navigation*. Licentiate Thesis, 2011. URL: http://kth.diva-portal.org/smash/record.jsf?searchId=1%5C&.

[304] F. Zhou, H. Been-Lirn Duh, and M. Billinghurst. "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR". In: *7th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR 2008, Cambridge, UK, 15-18th September 2008*. 2008, pp. 193–202. DOI: 10.1109/ISMAR.2008.4637362.

[305] R. Zhu and Z. Zhou. "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package". In: *IEEE Transactions on Neural systems and rehabilitation engineering* 12.2 (2004), pp. 295–302.

اَلحَمدُ لِلّهِ رَبِّ العَالَمِين