Institute for Computer Science XVII
Robotics

Master's thesis

Robust Head Tracking for Underwater VR Astronaut Training

David Hilbert

February 2025

First reviewer:	Prof. Dr. Andreas Nüchter
Second reviewer:	Prof. Dr.-Ing. Sergio Montenegro
Advisor:	Dr. Michael Bleier

Abstract

Extra Vehicular Activities (EVA) are missions that take place outside a spacecraft or a space station, such as the International Space Station. These so-called spacewalks require a high level of training to ensure the necessary safety and precision needed for these missions. Previous approaches to simulating weightlessness in preparation for EVA include training in large pools such as NASA's Neutral Buoyancy Laboratory or ESA's Neutral Buoyancy Facility, in which the relevant modules of the ISS are recreated. However, this training method comes with a number of disadvantages, including high financial and personnel costs.

In contrast, the utilization of an underwater virtual reality headset, which can be used in any swimming pool of any size, offers a cost-effective and equally immersive alternative for astronaut training in a simulated zero-gravity environment.

The following work describes the development of an underwater VR headset for a full face diving-mask. Furthermore, the investigation focuses on an optical tracking solution that is extended and optimized by a sensor fusion with inertial sensors. The robustness of the system is then tested for latency and stability of the tracking.

Subsequently, a significant improvement in the overall robustness was observed. It has been demonstrated that the greatest progress is achieved for rotational movements, particularly due to the additional sensor data provided by the inertial sensor. In final tests at ESA's Neutral Buoyancy Facility, the system was tested and evaluated by qualified test persons in the field of underwater astronaut training and VR.

Zusammenfassung

Als Extra Vehicular Activities (EVA) werden jene Einsätze bezeichnet, die außerhalb eines Raumfahrzeuges oder einer Raumstation, wie der International Space Station, stattfinden. Diese sogenannten Spacewalks erfordern ein hohes Maß an Training, um die benötigte Sicherheit und Präzision dieser Missionen zu gewährleisten. Bisherige Ansätze zur Simulation der Schwerelosigkeit zur Vorbereitung von EVAs sind Trainings in großen Pools wie dem Neutral Buoyancy Laboratory der NASA oder der Neutral Buoyancy Facility der ESA, in denen die betreffenden Module der ISS nachgebildet werden. Diese Trainingsmethode weist jedoch eine Reihe von Nachteilen auf, darunter hohe finanzielle und personelle Aufwendungen.

Demgegenüber stellt der Einsatz einer wasserdichten Virtual-Reality-Brille, welche in jedem Schwimmbad beliebiger Größe verwendet werden kann, eine kostengünstige und ähnlich immersive Alternative dar, um sich auf die Schwerelosigkeit der Mission vorbereiten zu können.

In der folgenden Arbeit wird die Entwicklung eines Unterwasser VR Headsets für eine vollgesichts Tauchmaske (Full Face Mask) erläutert. Es wird eine Trackinglösung aus reinem optischen Tracking untersucht und durch eine Sensorfusion mit inertialen Sensoren erweitert und optimiert. Die Robustheit des Systems wird anschließend auf Latenz und Stabilität des Trackings getested.

Hierbei konnte insgesamt eine signifikante Verbesserung der Robustheit festgestellt werden. Insbesondere bei Rotationen wurden die größten Fortschritte durch die zusätzlichen inertialen Sensoren erzielt. In abschließenden Tests in der Neutral Buoyancy Facility der ESA wurde das System von qualifizierten Testpersonen im Bereich Underwater Astronaut Training und VR getestet und evaluiert.

Acknowledgement

I would like to take this opportunity to thank all those who have supported and motivated me during the preparation of this Master's thesis.

First and foremost, I would like to express my deepest gratitude to Prof. Dr. Andreas Nüchter for granting me the opportunity to write this thesis on such an engaging and exciting topic. Your guidance and support whenever needed have been invaluable throughout my academic journey. My appreciation also goes to my supervisor Dr. Michael Bleier for his invaluable assistance and expertise regarding the tracking of the VR headset. Your support, particularly with the optical tracking system, was essential to the quality of this thesis. I am also deeply grateful to Sven Jörissen, the master of 3D printing, whose knowledge and assistance with all hardware-related questions were indispensable. Your skill and dedication made a significant impact on the progress of this work. I would like to express my sincere gratitude to all the researcher of the Institute for Computer Science XVII (Robotics) at the University of Würzburg who contributed their time and assistance with the practical tests in various swimming pools. Your commitment and collaboration were invaluable in ensuring the smooth practical experiments of this research. I am especially thankful to Helene Klein for the excellent work during your internship, particularly in conducting benchmarking measurements for the overall latency of the system. Your dedication and contributions were instrumental in concluding this work's results.

Special thanks go to the European Space Agency (ESA) for providing me with the incredible opportunity to test the underwater VR headset at the European Astronaut Centre (EAC). This experience was a unique and unforgettable part of this journey. I am sincerely thankful to all the participants at the EAC who took the time to try out the headset and provided thoughtful feedback and ideas for improvement. Your contributions were critical for concluding the results of this work.

To my family, I owe a debt of gratitude for your unwavering support and understanding throughout this demanding process. Your encouragement and belief in me were a constant source of strength. Lastly, I would like to express my deepest appreciation to Anna, for your endless patience, support, motivation, and understanding through every challenge.

To everyone who played a part in making this thesis possible, I am forever grateful.

Contents

Chapter 1

Introduction

On November 2, 2000, NASA astronaut William Shepherd and Russian Aviation and Space Agency (now Roscosmos) cosmonauts Yuri Gidzenko and Sergei Krikalev arrived at the International Space Station (ISS) for Expedition 1 [27]. This event marks the beginning of a continuous human occupation of the ISS and humans living in space for over 24 years without interruption [15]. During this time, numerous experiments were carried out both inside and outside the ISS. In particular, the Extra Vehicular Activities (EVAs or spacewalks) are highly challenging tasks as they require a high degree of safety, precision and efficiency in their execution [6]. Those spacewalks demand full concentration and physical exertion for over six hours at a time [6].

To achieve this level of coordination, an extensive preparation and training is needed beforehand. While many of the tasks can be practiced in advance using replicas, the zero gravity environment also needs to be taken into account. To best prepare for this unfamiliar condition, astronaut training facilities offer large pools to simulate the weightlessness underwater. One example is the NASA Neutral Buoyancy Lab (NBL) located at the Sonny Carter Training Facility nearby the Johnson Space Center in Houston [14]. It is the largest indoor pool in the world, measuring 61.5 meters long, 31.1 meters wide and 12.2 meters deep while holding 23 million liters of water [6, 14]. Apart from the enormous size, the available space is limited and still not large enough to fit the complete International Space Station [6, 16]. Therefore, parts of the ISS required for specific tasks and trainings are contained in a different layout [6]. However, the operation of such large systems has some disadvantages, particularly in financial terms [8]. The funding required to maintain a pool of this size is significant and not every organization has the financial resources to construct or operate a laboratory of this size. Additionally, the pool requires a significant amount of maintenance. As an example, the water temperature is maintained within a narrow range of 28.9 to 30 degrees Celsius [8, 14]. Therefore, NASA is renting out the NBL to private companies to minimize the costs when the facility is not being in use [2, 8, 14]. Similarly to the financial resources needed to build or maintain such a pool, the creation of new modules for training purposes is associated with significant costs as well. In addition, as not every facility with a pool is capable of storing those modules, performing trainings with the modules is associated with high costs for transport and traveling. Added to this are the high personnel costs associated with the dives, as up to four safety divers are

required for each astronaut in training [8].

Therefore, exploring smaller and more cost-effective training solutions is beneficial. One such solution is virtual reality, which has become an integral part of astronaut training over the last decades [9]. The utilization of a virtual reality headset that can be used underwater integrates these two important practice methods of VR and neutral buoyancy training underwater. Simulations of the different modules and test objects provide valuable training experiences without the same financial burden. Developing a VR solution for underwater astronaut training results in a more comparable weightlessness situation in order to prepare for the actual zero-gravity experience in space. Concurrently, all visual references to the external environment are suspended, thereby increasing the sense of weightlessness and the immersive experience of the training. Another advantage of using a virtual reality headset is that large parts of the environment and modules can be modeled in software, which results in a reduction of the hardware actually required for testing. Furthermore, tests can be carried out in any swimming pool, increasing flexibility, reducing operational costs, and allowing for more frequent training sessions. [8]

In cooperation with the European Space Agency (ESA), these advantages of a hybrid system for astronaut training are researched by the Institute for Computer Science XVII (Robotics) at the University of Würzburg. The project was initiated to develop a watertight VR headset together with a tracking system for position and orientation determination to be used in any available pools, leveraging the benefits of a hybrid VR solution that requires a limited operational area. [7]

In addition to being watertight, a VR headset that is compatible with a full face diving mask like the Interspiro FFM [12] allows training sessions to be carried out over a long period of time while ensuring robust communication via the microphone integrated in the diving mask. However, to ensure additional immersive interactive training, it is advisable to place an object with reference points that are also visualized in the virtual world. An example of such a reference object is the cubic structure in the center of Figure 1.1, which is also rendered in the virtual environment. To ensure the effective use of the headset for astronaut training, it is essential to have six degrees of freedom (DOF) tracking capabilities. This allows for independent translational movements in a simulated zero-gravity environment.

One way to achieve consistent 6-DOF tracking of objects in space is to utilize an optical tracking system (OTS). In this context, cameras serve as the sensors, responsible for the estimation of object poses within the view. By placing different fiducial markers in the field of view of one or more cameras, the relative position to the camera is determined using a fixed arrangement of the markers, leading to an absolute position in space. Although many VR systems, such as the Meta Quest, utilize inside-out tracking, in which the cameras are mounted on the virtual reality headset, this work examines outside-in optical tracking, a technique in which cameras are mounted externally from the object under observation. The selection of this method was motivated by the benefits that the relocation of the hardware to the exterior provides a simplified structure with no limitations on the dimensions and capabilities of the components (such as a powerful computing system). With outside-in tracking, the position and orientation of the object are determined based on the location of fiducial markers on the object's surface. Nonetheless, optical tracking systems are not without drawbacks. For instance, when markers become obstructed, there is an immediate loss of pose information. Due to the increased data

Figure 1.1: Example of a person using the proposed VR headset underwater to simulate a space environment for training purposes in a neutral buoyancy environment. The virtual environment was subsequently added to the image as it is perceived by the person wearing the underwater VR headset.

that must be processed, optical tracking systems exhibit reduced processing speed, lower sampling rates, and increased delays in comparison to internal sensors, such as inertial measurement units (IMU). Therefore, the headset is equipped with additional IMU in this work to achieve low latency outside-in tracking.

The work at hand investigates a methodology for a robust six degrees of freedom head tracking utilizing an optical tracking system and inertial sensors to compensate for the previously named drawbacks of optical systems. Therefore, this thesis aims to develop a robust head tracking for an underwater VR headset that operates with lower latency and delay compared to an initial version of the VR headset utilizing only an optical tracking system. In the following, the work is first classified scientifically in relationship to the current state of the art in the domain of underwater VR and astronaut training. Furthermore, fundamental concepts employed in this work are explained in greater detail, with one such concept being the Kalman filter utilized for sensor fusion. After that, initial requirements for an underwater VR headset for astronaut training are defined. Subsequently, hardware and software solutions devised for the underwater VR headset are explained. Special attention is focused on the sensor fusion of the tracking, which is elaborated upon in detail. Finally, the developed concepts are put to the test in practice. The robustness of the tracking is assessed through a series of performance experiments. A comparison evaluation of the tracking robustness is performed between this system and the initial version based purely on optical tracking. Furthermore, a practical test at ESA's Neutral

Buoyancy Facility (NBF) is evaluated, and the experiences of ESA astronauts and VR experts for astronaut training are taken into account.

Chapter 2

Related Work

2.1 Virtual reality for EVA training and design

Garcia et al. [9] provided an overview of the utilization of virtual reality in astronaut training for the last decades. The utilization of virtual reality headsets in the preparation of astronauts for Hubble repair missions was the first step in the integration of VR training into the astronaut training schedule. This is particularly relevant for training involving the Simplified Aid for EVA Rescue (SAFER) jetpack system. For example, to train special rescue scenarios where physical contact with the International Space Station is lost. This training will take place both in the VR laboratory (VRL) and on the ISS to further supplement the training. In the VRL, the Charlotte Mass Handling Robot is integrated into the training program, which simulates the payload together with handrails in zero gravity, providing a physical representation additionally to the representation in the VR application.

Ennis et al. [5] discussed the development and evaluation of VR based On-Board Training (VR-OBT) for astronauts on the International Space Station ISS. A modified Oculus Quest headset was used as a standalone solution (requiring no external hardware), incorporating solutions to overcome the challenges of a microgravity environment due to the lack of acceleration. They demonstrated their approaches for increased usability and tracking with initial tests during parabolic flights. The VR-OBT is utilized for maintenance and "Remove & Replace" tasks for the ESA Life Support Rack payload.

Nillson et al. [17] demonstrated the application of virtual reality for design studies of lunar surface systems, using the European Large Logistics Lander (EL3) concept as an example. Through evaluations and interviews involving astronauts and space experts carrying out a predefined mission objective, qualitative feedback on the task and the design of the EL3 could be obtained. Therefore, the study highlights a further use of VR in addition to EVA astronaut training as a cost-effective tool for early prototyping, resulting in more ergonomic, safe, and efficient lunar surface systems.

2.2 Underwater VR for scuba experiences

Hatsushika et al. [11] developed an underwater VR headset for scuba training which is usable in easily accessible waters, such as pools. The development of confidence skills for dealing with unexpected and dangerous situations underwater is of great importance. As these situations are not easily replicated in open sea, they can be repeated as desired within an underwater virtual environment. The hardware implementation adapts a disassembled Oculus Rift DK2 into a 3D printed watertight housing. Alongside the VR technology, an additional inertial measurement unit is fitted in the housing to estimate the orientation of the headset. This configuration utilizes a data pipeline similar to that selected for this thesis, whereby sensor data is transmitted to a PC for orientation computation and rendering of the virtual environment in Unity. The video output is subsequently transmitted back to the headset. The watertight housing is designed to be used similar to traditional diving googles. However, the system is limited to three degrees of freedom tracking, which restricts capturing the headsets movements to only rotation without translational tracking.

2.3 Underwater VR for astronaut training

Sinnot et al. [8] described an underwater VR headset for a full-face scuba mask for simulation and training of the Simplified Aid for EVA Rescue (SAFER) jetpack system. The objective is to achieve zero gravity like conditions analogous to the neutral buoyancy training in the NASA's NBL, with comparable reasons to those of this project, including high costs and limited accessibility of large pools. In this system, a smartphone serves as the primary sensor, display, and processing unit, and is mounted inside the diving mask. Rotational movements are tracked using the smartphone's built-in inertial sensors, providing three degrees of freedom. Translational movements are controlled via an Xbox controller, restricted to two degrees of freedom, enabling forward and lateral motion only. Although this configuration is functional, it presents several limitations. The visual experience is less appealing, and the system is constrained by the smartphone's processing capabilities, resulting in a performance bottleneck. Additionally, the system lacks direct interaction with the environment and does not support true six degrees of freedom (6DOF) tracking, compared to the goal of this thesis.

2.4 Tracking and sensor fusion approaches

Tobergte et al. [25] propose a robust multi-sensor fusion approach for pose estimation in medical applications by integrating an optical tracking system and an inertial measurement unit. Among other things, the authors set requirements for the result of the sensor fusion, which were adopted as similar objectives for the outcome of the tracking of this thesis. The method employs an extended Kalman filter to tightly couple these sensor inputs to achieve improved latency compensation, reduced noise in orientation estimates, and robustness in tracking under challenging conditions.

Chapter 3

Fundamentals

3.1 Notation

The title of this thesis is referring to the robustness of tracking. As robustness is a quite general term, it will be clarified and distinguished for the following part of the work. *Delay* refers to both the latency of the system and individual time delays such as lags. *Robustness* on the other hand is evaluated primarily according to outliers and jumping of the tracking data.

In some cases, the terms *rotation* and *orientation* are used interchangeably. The current orientation of the viewing direction of the headset, or the rotation of the headset from its original position to the viewing direction of the headset is an example of this phenomenon.

The Robot Operating System 2 in the Iron Irwini version is used in this work. This will be referred to as *ROS* in the following, even if technically *ROS 2 - Iron Irwini* is meant.

3.2 Low-pass filter

As the name suggests, a low-pass filter is designed to permit only frequencies below a specified limit to pass through, thereby blocking high frequencies. A common application of the low-pass filter is the smoothing of inputs that are noisy at high frequencies. The low-pass filter is applied in various forms. As an electronic circuit, a first order low-pass filter is realized by utilizing a resistor and capacitor as seen in Figure 3.1. As no other source is used to filter the signal, this is also called a passive low-pass filter. [10]

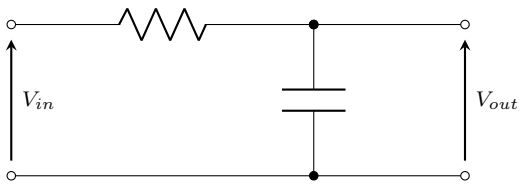

Figure 3.1: Electronic circuit of a (passive) RC low-pass filter. (Reproduced based on [10])

3.2.1 Moving-average filter

The low-pass filter can also be implemented as a digital filter, for example, to smooth sensor data and remove noise. One common type of digital low-pass filters is the discrete first-order infinite impulse response filter, also known as an *exponentially weighted moving-average filter* (EWMA), since its weighting of older values decreases exponentially. With the EWMA filter, the output y_t for time step t is calculated using a percentage weighting of the current input x_t and the previous output y_{t-1}. The weighting is determined by the smoothing factor α. [21]

$$y_t = \alpha x_t + (1 - \alpha) y_{t-1} \tag{3.1}$$

$$= y_{t-1} + \alpha(x_t - y_{t-1}) \tag{3.2}$$

The following on Algorithm 1 demonstrates an EWMA filter on a set of n input data x_t with $0 \leq t < n$.

Algorithm 1 Implementation of an EWMA filter

$y_0 \leftarrow x_0$
$t \leftarrow 1$
while $t < n$ **do**
  $y_t \leftarrow y_{t-1} + \alpha \cdot (x_t - y_{t-1})$
end while

Furthermore, y_t represents the filtered output value of the low-pass filter at step t. The smoothing factor α is an important part in the design of the filter, as it determines the extent to which high frequencies are filtered out. A very low smoothing factor leads to a slower response of the output to changes in the input values. [21]

Another type of moving average filter is the *simple moving average filter* (SMA). Here, the output y_t is determined by the average of the last k inputs. [21]

$$y_t = \frac{x_t + x_{t-1} + \cdots + x_{t-k+1}}{k} \tag{3.3}$$

3.3 Kalman filter

3.3.1 The origins of the Kalman filter

The Kalman filter is a set of mathematical functions that were published by R.E. Kalman in 1960 and have been continuously developed ever since. They provide by recursive application a computational efficient solution to the least-squares method. The Kalman filter is designed to estimate the state of a discrete-time system that is described by a linear difference equation. [29]

The discrete linear Kalman filter relies on the following two assumptions: [22]

1. The system is linear
2. The system model is affected by white Gaussian noise

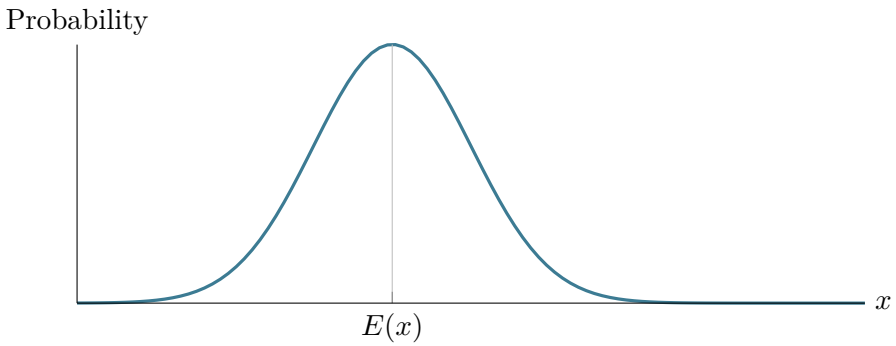

Figure 3.2: Probability Density Function (PDF) of random variable x with the expected value $E(x)$. (Reproduced based on [1])

There exist solutions for nonlinear systems like the Extended Kalman filter (EKF) or the Unscented Kalman filter (UKF) which are not utilized in this work. A Gaussian distribution, also known as the Normal distribution, is characterized by the fact that it is only defined by the first and second moments, mean (μ) and variance (σ^2), as seen in equations 3.4 and 3.5. In the following, x represents the random variable of our system.

$$\mu = E(x) \tag{3.4}$$

$$\sigma^2 = E((x-\mu)^2) \tag{3.5}$$

Gaussian distributions are described by the Probability Density Function. Therefore, a continuous random variable x has a Normal distribution defined by the first and second moment, if it has the following Probability Density Function as seen in Figure 3.2: [1, 22]

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.6}$$

Assumption 2 therefore states that a measurement is described with the Probability Density Function in order to be considered for the use of a Kalman filter. The random variable x_k represents the true system state to be estimated at a point in time k. Furthermore, this estimation is also expressed as the expected value at time k. [1]

$$E(x_k) = \hat{x}_k \tag{3.7}$$

$\hat{x}_k$ describes the state estimation, trying to minimize the error e_k corresponding to the real state x_k at time k. Therefore, error e_k is defined as

$$e_k = x_k - \hat{x}_k. \tag{3.8}$$

With the definitions of variance (equation 3.5) and expected value of a random variable (equation 3.7) and taking into account equations 3.7 and 3.8, once can derive the estimate error covariance P_k for the random variable x_k:

$$\begin{aligned} P_k &= E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \\ &= E[e_k e_k^T] \end{aligned} \tag{3.9}$$

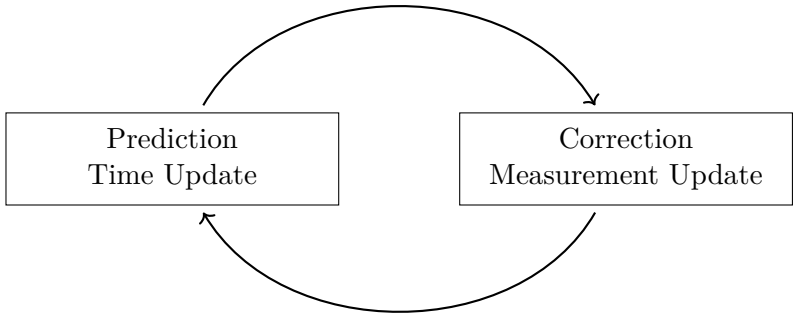

Figure 3.3: The Kalman filter algorithm. (Reproduced based on [29])

3.3.2 The Kalman filter algorithm

The Kalman filter is separated in two steps in a form of a feedback control loop as shown in Figure 3.3. The filter first predicts a state based on the physical model for the next time step and then corrects the previous prediction based on a new measurement at a later point in time. Therefore, all the equations are departed in either being used in the time update (prediction) or the measurement update (correction) steps of the filter. [29]

The most notably variable in those equations is the state estimation $\hat{\boldsymbol{x}}$. As this variable undergoes adjustments in both the prediction and the correction, a distinction is made below between the two states of this variable. The state prediction is expressed by $\hat{\boldsymbol{x}}_{k+1}^{-}$, also known as the *a priori* state estimation. In this case, the $k+1$ indicates that a prediction for the next time step is made. In contrast, the state correction is specified by $\hat{\boldsymbol{x}}_k$, also known as the *a posteriori* state estimation. The *a posteriori* state estimation $\hat{\boldsymbol{x}}_k$ corresponds to the final state estimation or expected value we want to derive for every time step k from equation 3.7. [29]

Similar to the distinction between the *a priori* state estimation $\hat{\boldsymbol{x}}_{k+1}^{-}$ and the *a posteriori* state estimation $\hat{\boldsymbol{x}}_k$, the *a posteriori* estimate error covariance from equation 3.9 has to be updated after every prediction step to the *a priori* estimate error covariance $\boldsymbol{P}_k^{-}$.

$$\boldsymbol{P}_k^{-} = E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{-})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^{-})^T] \tag{3.10}$$

As already mentioned in the beginning of Section 3.3.1, the subject of the Kalman filter is to estimate the state of a discrete-time system, represented by a linear difference equation at a time step k:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{w}_k \tag{3.11}$$

$\boldsymbol{w}_k$ is the process noise, which, following Assumption 2, is assumed to be independent, white and representing a Gaussian distribution. $\boldsymbol{u}_k$ describes a control input to the system. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ relate the system state $\boldsymbol{x}_k$ and input $\boldsymbol{u}_k$ to the state at the next time step $\boldsymbol{x}_{k+1}$. The measurement of the system is given by

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k. \tag{3.12}$$

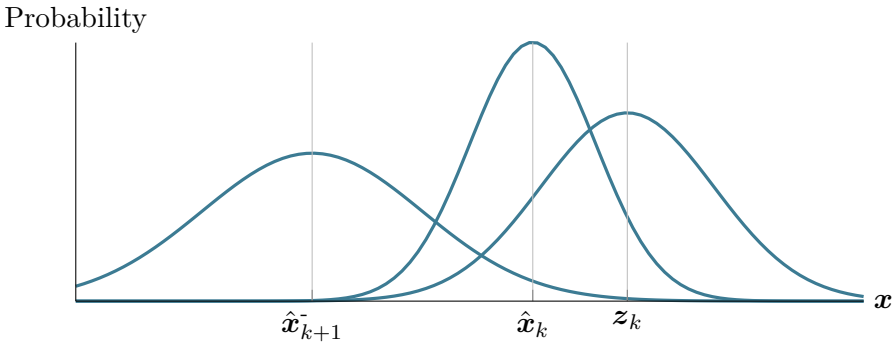

Figure 3.4: Probability density fusion of predicted estimate and measurement input in the correction step. (Reproduced based on [1])

Similar to the process noise, $\boldsymbol{v}_k$ represents the measurement noise, which is again assumed as independent, white and normally distributed. The matrix $\boldsymbol{H}_k$ relates the state $\boldsymbol{x}_k$ to the measurement $\boldsymbol{z}_k$. [29]

As already mentioned in the beginning of Section 3.3.2, the estimate of the system state gets updated during the correction step, based on the *a priori* state estimation $\hat{\boldsymbol{x}}_k$ from the prediction and the new measurement $\boldsymbol{z}_k$. This involves the combination of both measurement and state prediction, taking into account the uncertainty of both. In particular, their error covariance matrices $\boldsymbol{R}$ for the measurement and $\boldsymbol{P}^-_k$ for the state prediction. Since Assumption 2, both random variables follow a Gaussian distribution. By combining the Gaussian distributions of the predicted state and the measurement in the correction step, the resulting *a posteriori* state estimate $\hat{\boldsymbol{x}}^-_k$ also follows a Gaussian distribution with an improved uncertainty $\boldsymbol{P}_k$ (see Figure 3.4). [29]

Prediction equations

The prediction or time update step involves predicting the future state of the system and the uncertainty associated with this prediction. To predict the system state at the next time step, the Kalman filter uses the linear equation representing the system already described earlier with equation 3.11.

$$\hat{\boldsymbol{x}}^-_{k+1} = \boldsymbol{A}\hat{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{3.13}$$

Along with predicting the state, the error covariance matrix of the state estimate is updated, which represents the uncertainty of the predicted state estimate $\hat{\boldsymbol{x}}^-_k$.

$$\boldsymbol{P}^-_{k+1} = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^T + \boldsymbol{Q} \tag{3.14}$$

$\boldsymbol{Q}$ is the process noise covariance matrix that represents the uncertainty in the process model. [29]

Correction equations

The difference between the actual measurement and the predicted measurement is called **innovation** and therefore describes the new information coming to the system from the measurement

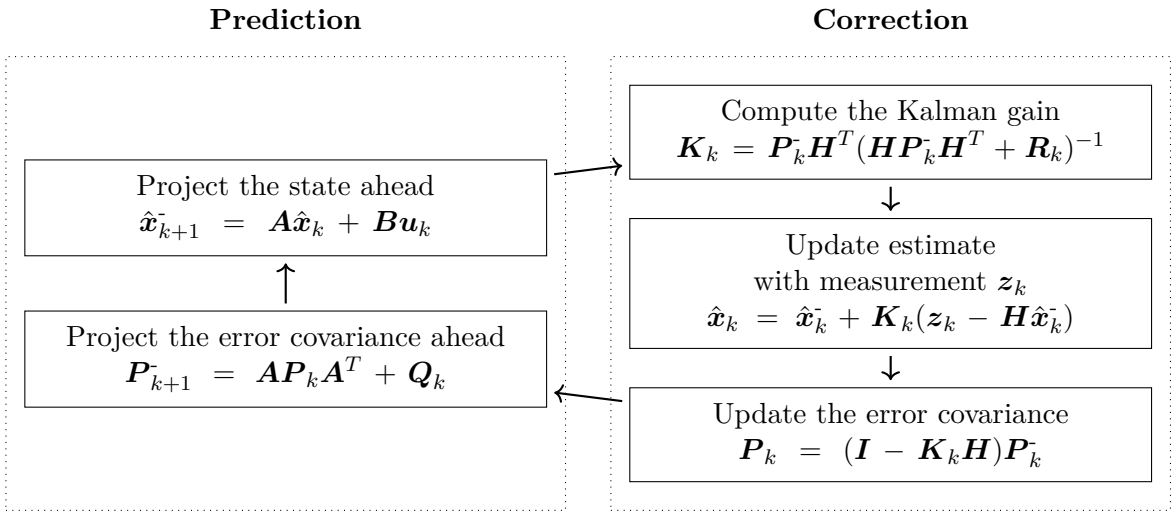

Figure 3.5: Detailed algorithm overview. (Reproduced based on [29])

input. The innovation $\boldsymbol{y}_k$ is expressed by the term

$$\boldsymbol{y}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^{-}. \tag{3.15}$$

Since there are covariance matrices for the measurement and state estimation, the error covariance matrix of the innovation is expressed as

$$\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_k^{-}\boldsymbol{H}^T + \boldsymbol{R}. \tag{3.16}$$

$\boldsymbol{R}$ is the measurement noise covariance matrix, describing the uncertainty in the measurement. [29]

The weight of the innovation is specified by the **Kalman Gain** $\boldsymbol{K}_k$. It determines how much the measurement should be weighted in contrast to the prediction.

$$\begin{aligned} \boldsymbol{K}_k &= \boldsymbol{P}_k^{-}\boldsymbol{H}^T\boldsymbol{S}^{-1} \\ \boldsymbol{K}_k &= \boldsymbol{P}_k^{-}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_k^{-}\boldsymbol{H}^T + \boldsymbol{R}_k)^{-1} \end{aligned} \tag{3.17}$$

The *a posteriori* state estimate $\hat{\boldsymbol{x}}_k$ is updated by adding the innovation $\boldsymbol{y}_k$, which is weighted by the Kalman Gain $\boldsymbol{K}_k$, to the predicted *a priori* state estimate $\hat{\boldsymbol{x}}_k^{-}$.

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^{-} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^{-}) \tag{3.18}$$

The state covariance $\boldsymbol{P}_k$ gets updated to represent the *a posteriori* error covariance of the updated state estimate $\hat{\boldsymbol{x}}_k$.

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^{-} \tag{3.19}$$

In the equation above, $\boldsymbol{I}$ describes the identity matrix. This reduces the uncertainty after combining it with the new measurement $\boldsymbol{z}_k$ as seen in Figure 3.4. [29]

Figure 3.5 summarizes the combined equations for the prediction and correction steps of the linear Kalman filter algorithm.

3.3.3 Adjusting the parameters

Recalling equation 3.17

$$\boldsymbol{K}_k = \frac{\boldsymbol{P}_k^- \boldsymbol{H}^T}{(\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R}_k)},$$

it becomes clear that different values of $\boldsymbol{R}_k$ and $\boldsymbol{P}_k^-$ have different effects on the Kalman Gain $\boldsymbol{K}_k$ and therefore on the significance of the measurement $\boldsymbol{z}_k$ in contrast to the predicted estimation $\hat{\boldsymbol{x}}_k^-$. In particular, if the measurement error covariance $\boldsymbol{R}_k$ is chosen to approach zero, the Kalman Gain $\boldsymbol{K}_k$ approaches $\boldsymbol{H}^{-1}$ and the innovation is weighted more heavily. In other words, the measurement and therefore the correction is trusted more than the estimate. [29]

$$\lim_{\boldsymbol{R}_k \to 0} \boldsymbol{K}_k = \boldsymbol{H}^{-1}$$

If the *a priori* estimate error covariance $\boldsymbol{P}_k^-$ approaches zero, the Kalman Gain $\boldsymbol{K}_k$ approaches zero as well and ultimately the innovation has less impact on the correction. Therefore, the estimation is trusted more than the correction or measurement. [29]

$$\lim_{\boldsymbol{P}_k^- \to 0} \boldsymbol{K}_k = 0$$

Chapter 4

Underwater VR and Tracking

4.1 Problem specification and requirements

The main objective of this work is to develop a VR headset with a robust head tracking in terms of low latency and lag for underwater astronaut training applications. As previously stated in the introduction, conventional astronaut training for extra vehicular activities is currently carried out in large pools containing replicas of modules needed for the task. Among the disadvantages named in the beginning is the need for large pools, associated with high service costs, as well as the high expenses needed for manufacturing and replacing the modules due to the still limited size of the pools. On the other hand, exploring a virtual reality solution to model any test environment directly in the VR headset results in a cost-effective and space-saving alternative. While these tests are still conducted underwater, the simulation of a weightless environment remains intact. Due to the modules or test objects being visualized directly in the glasses, there is no need for large physical testing modules, requiring much less needed space for the tests. As a result, the tests can be performed in any reasonably sized swimming pool. Another advantage is that changes to the displayed environment are quickly made, eliminating the need to create or modify a physical test environment. As mentioned in the introduction, this results in increased flexibility in conducting the tests, as well as reduced operating costs, allowing for more frequent training sessions. This leads to an improved preparation and enhanced safety for specific tasks that are associated with high risks such as Extra Vehicular Activities.

The challenges in the development of such headset are, on the one hand, to make it waterproof for use underwater and, on the other hand, to ensure reliable tracking without excessive lags and delays. To be used as a sufficient alternative to astronaut training, the trainee must operate underwater for an extended period of time. Since commercial headsets are not waterproof and cannot be used with a diving mask, a solution must be developed that allows for the combination of a diving mask and a waterproof VR construction. Motion sickness is an issue for many users of VR glasses, especially when the system is affected by large delays in the visualization of head movements. This effect is intensified when floating underwater, as it is easy to become disoriented without a view to the outside. In order to be used for training, the headset must be worn for extended periods of time without causing motion sickness.

While inside-out tracking has the advantage of all components being stored inside the head-

set, resulting in a simpler setup and greater portability, there are many reasons that favor outside-in tracking for a low latency head tracking for underwater astronaut training. Due to the limited space to accommodate all the components in a headset, there are clear drawbacks to the physical computing power that can be integrated. With outside-in tracking, there is no limit to the physical size of the processing unit, so it can be scaled up as needed to further improve results.

Since outside-in tracking has been chosen for this work, cameras need to be placed in fixed, known positions around the area to test. Those need to be watertight as well. In addition, the camera parameters must be adapted for underwater usage and optimized for the trade-off between high resolution and accurate tracking or low latency and lower accuracy. The recorded images then need to be processed at a central processing unit to retrieve the objectives relative position.

As already being mentioned, the utilization of an optical tracking system without any additional sensors has proven to be an unreliable solution [25]. Among the problems were the loss of positional information caused by the occlusion of markers in the line of sight from the cameras. Another drawback is the sampling rate of optical tracking systems, which is generally lower than that of inertial sensors like IMUs. This is due to the significant data processing required for image analysis. Therefore, the VR headset must be equipped with an inertial sensor to offset the limitations of an optical tracking system and enhance the overall tracking quality and stability. IMUs provide data at reliable intervals, as they are not dependent on external factors, such as good visibility. However, as the internal gyroscope and accelerometer measure only derivatives of orientation and position (i.e. angle rate and acceleration), position and orientation data can only be obtained by numerical integration of the measurements. Since the measurements are also affected by noise, this leads to drift in the calculated orientation and position. While most IMUs also contain a magnetometer to compensate for the orientation drift, there is no option for a standalone IMU to prevent position drifting. Therefore, an IMU is not a replacement for an optical tracking system. Instead, it serves as an inexpensive extension that is included directly in the VR headset to improve the overall head tracking. [25]

Furthermore, an appropriate approach for the sensor fusion between IMU and optical tracking is needed. The Kalman filter is a proven approach when it comes to sensor fusion and is used in many areas of real-time data acquisition. Examples of this include distance measurements, as used for self-driving cars with lidar and onboard radar, and position tracking using GNSS (Global Navigation Satellite System) and inertial sensors. [1]

Tobergte et al. [25] set different goals for a sensor fusion between IMU and OTS that also apply to this project. First, the sensor fusion needs to work autonomous, even in the case of loosing sight to the fiducial markers (by being blocked by body parts or getting out of view) for short periods of time. Position and orientation estimation should be free of significant delay (leading to symptoms like motion sickness). The sensor fusion of different inputs should reduce overall noise and lag. Finally, the resulting sensor fusion tracking data should always be at least as good as the underlying optical tracking system, or better, but never worse. [25]

As previously stated, one of the reasons for choosing outside-in optical tracking is that there is no limitation for the processing unit, which performs the position calculations as well as the visualization of the virtual environment on the VR headset. The Robotic Operating System (ROS) serves as the software foundation for the sensor data acquisition, fusion and tracking.

With the game engine Unity, the virtual environment for the astronaut training is modeled and visualized during the training. Additionally, the communication between the tracking (ROS) and visualization in the headset needs to operate at low latency and lag to fulfill the main objectives of this work.

4.2 Utilized software tools and libraries

A foundation of different software was needed for the sensor data acquisition and position tracking as well for creating the virtual environment displayed in the headset. The following section will describe the software that is running on the Linux workstation selected for this project.

4.2.1 Unity

The Unity game engine was chosen to create the virtual environment experienced through the headset. Unity is a game engine that allows creating 2D and 3D video game focused applications. With its visual editor, graphical environments can be designed easily and enhanced by importing own 3D files or contents of Unity's Assets Store. Unity supports cross-platform development and is therefore suitable to run on the Linux powered computer for the project. With many community forums and extensive documentation, Unity provides a good basis for creating the graphical user interface of the headset. [23]

In a quick summary, a Unity project consists of several game objects that can be placed in a 2D or 3D scene via the Unity editor and therefore form the virtual game environment. The game's mechanics are implemented through the integration of scripts into its game objects, enabling the manipulation of properties such as position, orientation, and color. These scripts can listen to different events, such as keyboard inputs. One important game object that is used in the majority of the Unity projects is the `player`. The `player` game object characterizes the central object within a scene, which is manipulated by the user. For example, by attaching a script that receives keyboard input signals and add or subtract those to the position value of the `player`, it can be moved within the scene.

4.2.2 ROS (2 - Iron Irwini)

The Robotic Operating System 2 (ROS 2) provides a good foundation for processing the optical and inertial sensor data and computing the poses. ROS is an open-source robotics middleware framework for building and developing robotic applications. Therefore, ROS comes with a set of software libraries, drivers, developer tools and algorithms making it a very suitable foundation for any automation and sensor focused software project. [20]

ROS 2 is the successor of ROS, which was started in 2007, and comes with a complete redesign of the framework. With a lot of improvements aimed at the shortcomings of the first generations, there are new changes in terms of architecture, general usability and security [13]. The ROS 2 version Iron Irwini was used for this project. When ROS is mentioned for the following parts of the paper, this refers to the used ROS 2 version Iron Irwini. The basic functionality of ROS communication is briefly described below, but a much more comprehensive insight is found in the ROS 2 documentation (see [20]).

As previously stated, ROS is a middleware that utilizes a strongly-typed, anonymous publish/subscribe mechanism. Based on that communication protocol, it allows for message passing between various processes within a ROS system and also between several ROS entities in the same network. A central component of any ROS system is represented by the ROS graph. It refers to the network of nodes in ROS and their connections among each other that enable the communication within the system. A node represents an entity within the ROS 2 graph, that can communicate with other nodes within the same process, across different processes or even on different machines. Furthermore, a node should typically represent the unit of one computation and therefore multiple nodes should be separated according to the nature of their logical calculations. As mentioned for the ROS graph, nodes can publish data to named topics to share data or subscribe to named topics to receive data. [20]

ROS topics are one of the interface styles used for communication. They are often used for continuous data streams, such as data coming from different sensors or the state of a robot. Nodes acting as publishers or subscribers connect to each other through a ROS topic, which acts as a common identifier to help the entities locate each other. A topic is not limited to a specific number of publishers or subscribers. Therefore, a particular topic may have zero or more publishers and zero or more subscribers. While a ROS topic is a common interface through which the various publisher and subscriber entities can communicate, the underlying instance that is exchanged is called a ROS message and is defined in ".msg" files. A message consists of one or more data fields and/or constants, that are specified by common data types like "int32" and "string". Therefore, a ROS message describing the position of an object in a three-dimensional space would consist of three fields with the field types "float32" and field names x, y and z. Finally, since ROS nodes represent different entities that work and communicate with each other, it would be a tedious process to start each node individually in order to get the desired ROS system running. This procedure is automated by the ROS launch system, which is designed to automate the execution of multiple nodes within a single command. A major advantage of the launch system is that the state of the launched nodes can be monitored and feedback and changes in the state of the nodes are reported. [20]

The ROS bag tool provides a convenient way of recording and playing back published topics on a system. This functionality enables the recording and storage of either all or selected topics in a designated ".bag" file. Subsequently, this file can be replayed, thereby publishing the recorded topics. Therefore, the recorded testing scenario is repeatedly simulated for further development. [20]

4.2.3 AprilTag - Fiducial marker tracking

Fiducial markers, or visual fiducial tags, are two-dimensional artificial landmarks designed to be easily recognized and distinguished from one another. They play a key factor in experimental settings where the objective is to create controlled experiments with simplified visual features. This is in contrast to scenarios where the focus is on identifying naturally occurring features. [18]

These are commonly used in the field of augmented reality and robotic systems. For the latter, they can be used to achieve ground-truth estimation for robot trajectories and to close control loops. An example of this is a potential application of SLAM (Simultaneous Localization

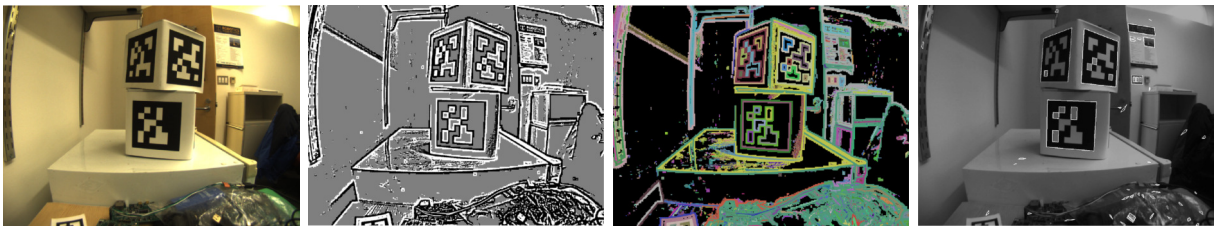

Figure 4.1: Processing steps (1) to (4) in the AprilTag algorithm to obtain the relative Pose. (Source [28])

and Mapping) systems. [18, 28]

Fiducial Tags

Examples of fiducial tag systems include ARToolKit, ARTag, and AprilTag, with the latter being utilized for the optical tracking system in this project. ARToolKit represents on of the first fiducial systems and consists of a library for augmented reality visual tag tracking. The fiducial tags comprise a black square tag with an embedded arbitrary image pattern inside, and they enable full 6-DOF pose estimation when matched against a database of known tags. ARTag replaced the arbitrary image by a 2D binary barcode pattern that allowed the correction of bit errors in the detection. AprilTag [18] adopted the binary barcode pattern from ARTag with improved localization accuracy, and provided an open source library of their detection algorithm. Therefore, AprilTag offers a good foundation to be used for the VR Headset tracking in this work, as seen in Figure 4.4. [28]

AprilTag (2) detector

The original AprilTag algorithm relied on computing an image gradient to extract the features by filtering for local maxima and clustering them into components. However, with AprilTag 2 [28], the detection algorithm has been updated to fundamentally improve the computational time.

Instead of computing the gradient of the pixels, the AprilTag 2 detector employs an adaptive threshold approach to differentiate the features provided by the fiducial tags. Other than finding a threshold for the entire image, it is divided into regions that assign each pixel a black or white value depending on their local thresholds. Regions of pixels with insufficient contrast are colored in gray and excluded from further computations to improve the overall performance, as illustrated in Figure 4.1 (2). The next step in the process is to connect the adjacent black and white pixels to segments of components. The boundaries of components are calculated by clustering pixels that are adjacent to the same black and white components, as outlined in step (3). Finally, quads are fit to each cluster of border pixels by identifying borders and lines, as seen on Figure 4.1 (4). Bad quad fits, as well as quads where no fiducial tag payload or barcode is recognized, are discarded. After identifying the edges of the fiducial tags of known size, their relative position to the camera can be determined. [28]

Figure 4.2: Interspiro Divator full-face diving mask. (Source [12])

4.3 Methods

4.3.1 Hardware description

The following section will describe the technical structure of the headset designed for this project and the test setup with cameras.

The watertight VR headset

To support astronaut training underwater for an extended period of time, a full-face diving mask serves as the basis for the headset. This has the advantage that the second stage of the regulator (through which the diver breathes) is attached directly to the mask instead of a separate mouthpiece and therefore does not interfere with speech [19]. In this case, an Interspiro Diving Mask is used, as seen in Figure 4.2, which also offers pressure equalizing pads for the use of earphones and microphones. Flawless communication between the trainees and the team is essential for safe and effective training.

The electrical components, such as displays and sensors, are then bonded to the exterior of the mask within a waterproof housing machined from aluminum. Early versions of the headset have been enclosed in a 3D printed case from PLA. The displays are two 2.9-inch MIPI (Mobile Industry Processor Interface Alliance) displays with a resolution of 1440×1440 pixels and a refresh rate of 120 Hz. These are connected together via a DisplayPort (DP) driver PCB. This board is powered by USB and therefore connected to a computer outside the headset by a DisplayPort and USB connection. The utilization of a 10-meter active optical DP cable for the connection to the computing unit is intended to minimize latency during the transmission of visuals to the headset. A set of lenses with a focal length of 50 millimeter is placed in front of the displays. This configuration serves to focus images for the user and expand the field of view. In early versions and prototypes of the headset (see Figure 4.3), the lenses were positioned directly ahead of the displays, allowing for small adjustments by attaching them to a rail.

Figure 4.3: First headset prototype with movable lenses directly in front of the displays.

However, for the final version of the headset, the lenses are placed inside the diving mask to minimize the distance between the eyes and the lenses. Since an inertial sensor is used in addition to optical tracking, this must also be integrated into the headset. For this purpose, the BNO085 IMU breakout board from Adafruit was selected, which is equipped with the BNO085 inertial measurement unit from CEVA Technologies, Inc. The sensor employs the same hardware as the Bosch Sensortech BNO055, but with an optimized firmware specifically designed for AR/VR applications. An Arm Cortex M0 processor is integrated in the sensor, which, when utilized in conjunction with the firmware from CEVA Hillcrest Laboratories, provides a sensor fusion of the accelerometer, gyroscope and magnetometer. The BNO085 is connected to the Adafruit QtPy RP2040 microcontroller via an I2C connection. The QtPy RP2040 is connected over a USB UART serial connection to the workstation, thereby providing the sensor data for subsequent processing. Consequently, the microcontroller is responsible for the initialization of the IMU, the access of the data and the transmission to the tracking system. As previously referenced, the DisplayPort cable and power connections, in conjunction with the USB cable for the IMU breakout board, are routed through a watertight tube that extends from the water to the computer and the external setup. [3, 4]

Furthermore, fiducial markers are attached to the exterior of the headset, enabling its precise localization within the room by the outside-in optical tracking system. A total of four markers are attached to the headset. Two markers to the front side of the headset and one is secured on each side, as seen in Figure 4.4.

Underwater cameras and external setup

Four cameras are used for the outside-in optical tracking. For this purpose, the FLIR BFS-PGE-23S3C-C Ethernet cameras are utilized within a waterproof housing, as illustrated in Figure 4.5. The cameras are connected to an Ethernet switch that transmits the data via a 10 GBit connection to the computer. All four cameras are positioned around the test area to ensure comprehensive visual coverage. To simplify the alignment during both the installation and the

Figure 4.4: Trying out the headset mounted on diving mask with AprilTag fiducial markers attached to sides. The virtual environment rendered on the headset's displays is mirrored to a monitor in the background.

transport, the cameras are mounted in pairs on a fixed structure consisting of 40 mm × 40 mm aluminum profile bars. Each of the structure is about 2.2 m long and 0.25 m by 0.25 m wide and high. This results for each camera being separated from the other by approximately 1.8 m along the structure and elevated about 0.3 m above the ground. Finally, the cameras are tilted inwards (30°) and upwards (15°) for an improved field of view of the test area (see Figure 4.6).

In addition to the camera setups, a central object is required that the user can interact with both physically and in the virtual representation. Therefore, it serves both as a kind of reference point that forms a bridge between the virtual training environment and the reality. The aim is to achieve an even more immersive training effect. For this purpose, a cubic structure measuring approximately 1 meter in length, width, and 0.42 meters in height is placed in the center of all four cameras. In addition, a 30 mm diameter aluminum rod is positioned on top of the structure. This serves as a guiding rail, similar to those found within and outside the International Space Station. It helps the trainee to physically connect to the simulated virtual environment, thereby enhancing the overall immersion experience. A colored marker is placed on top of each of the four corners, helping for navigating and positioning the cubic structure. These markers serve to identify each of the cameras in the software, as they are named after the respective marker, which they are facing. A three-dimensional rendering of the central structure, as well as the overall external setup of the cameras, can be seen in Figure 4.6.

Figure 4.5: Left: FLIR camera in underwater housing. Right: Camera and Lens in housing fixture.

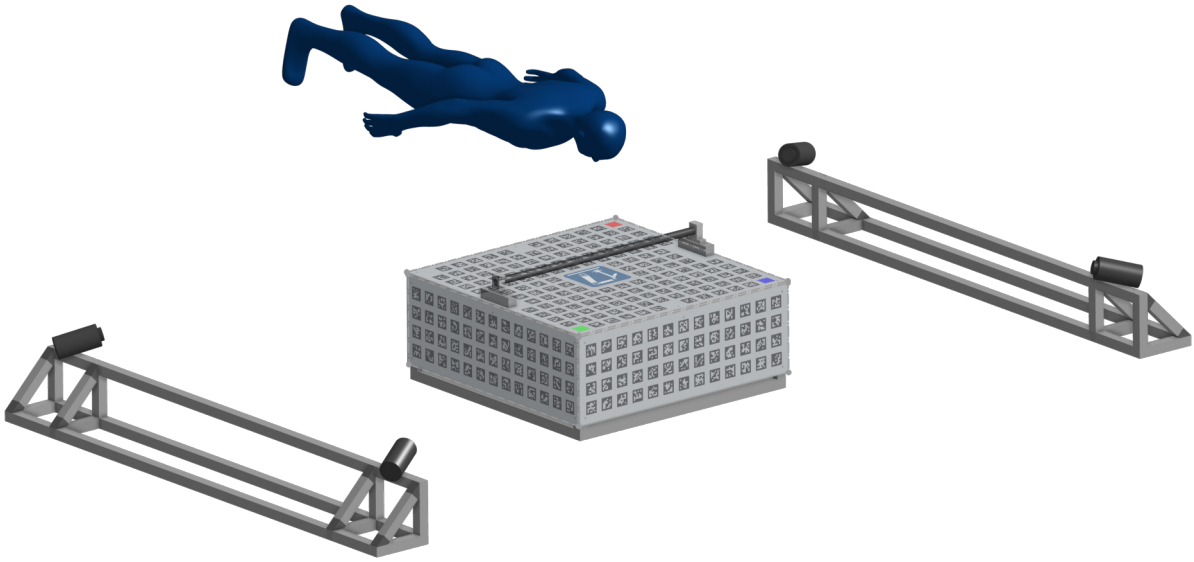

Figure 4.6: Render of the external setup underwater. A person is floating above the cubic structure with the cameras attached bars to both sides.

As previously stated, the project's core component is the processing unit, where all necessary software and calculations are executed. This includes the reception of optical and inertial sensor data from all four cameras and the integrated IMU to calculate the current position and orientation. It also comprises the computation and transmission of the graphics to the headset via the DisplayPort connection. In this instance, the processing unit is a high-performance VR-ready desktop PC (CPU: 13th Gen Intel(R) Core(TM) i9-13900KS, GPU: NVIDIA GeForce RTX 4090, RAM: 64GByte) running a desktop Linux distribution (Ubuntu 20.04).

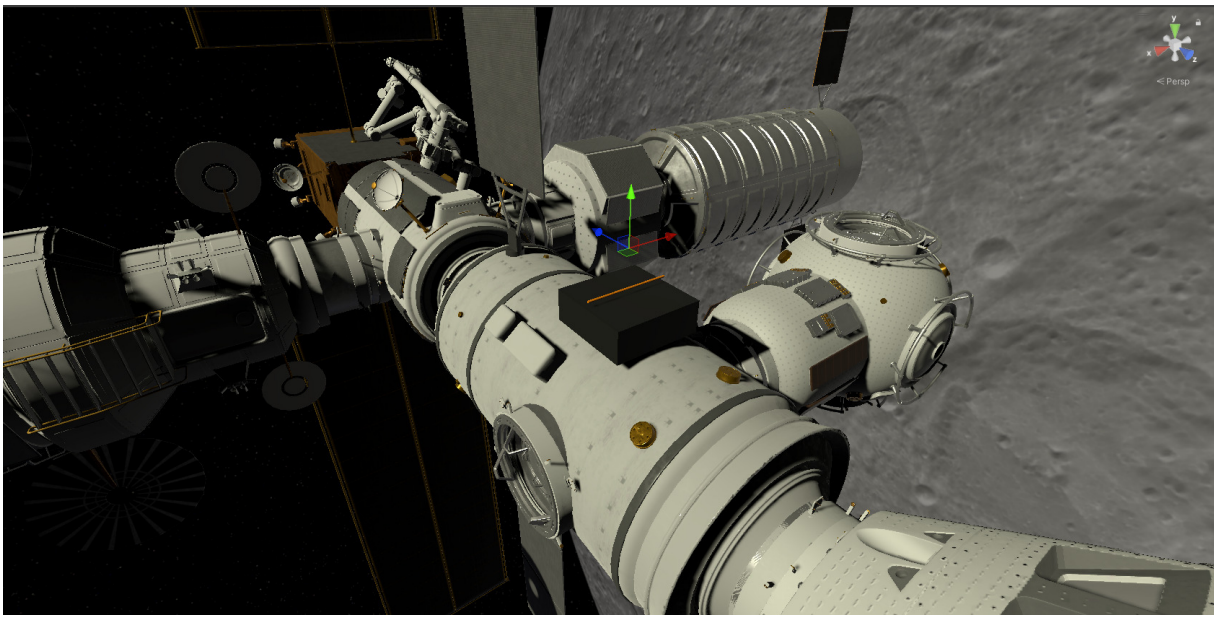

Figure 4.7: 3D virtual environment modeled in Unity. On top of the Lunar Gateway model, the platform (black) and guiding rail (orange) are recreated from the hardware setup.

4.3.2 Training environment in Unity

As already mentioned in Section 4.2, the game and graphics engine Unity was chosen to create the 3D virtual environment. One of the goals defined by ESA was to have the Lunar Gateway Space Station as the central objective in the modeled game environment. As there already exist numerous 3D models on the Internet under open licenses, one of these models was selected to be utilized for this project. The chosen model[1] provides Unity Universal Render Pipeline (URP) files to embed it directly into Unity. In addition, the cubic platform and guiding rail mentioned in the Hardware description (Section 4.3.1) were recreated as game objects in Unity and placed on top of the Gateway model. The exact dimensions and positions of the objects in relation to each other have been preserved in Unity, so they match the dimensions of the real objects.

In Figure 4.7 the 3D model of the Lunar Gateway and the game objects of the platform (black) with guiding rail (orange) can be seen, as well as the world coordinate frame in the top right corner. The player's coordinate frame is visualized in the center of the figure above the platform. A characteristic of Unity is that the coordinate frames are left-handed. This can be seen with both player and world coordinate frames in Figure 4.7, where the red axis represents x, green y and blue z. The z axis is pointing towards the robotic manipulator of the space station and represents the player's line of sight. The player's view is shown in Figure 4.8, which illustrates the dual screen view as displayed inside the VR headset. This was achieved by

[1] Created by Andreas Engevold and published under free license:
`https://www.artstation.com/marketplace/p/5j9Jv/gateway?utm_source=artstation&utm_medium=referral&utm_campaign=homepage&utm_term=marketplace`
(No changes were made to the 3D model)

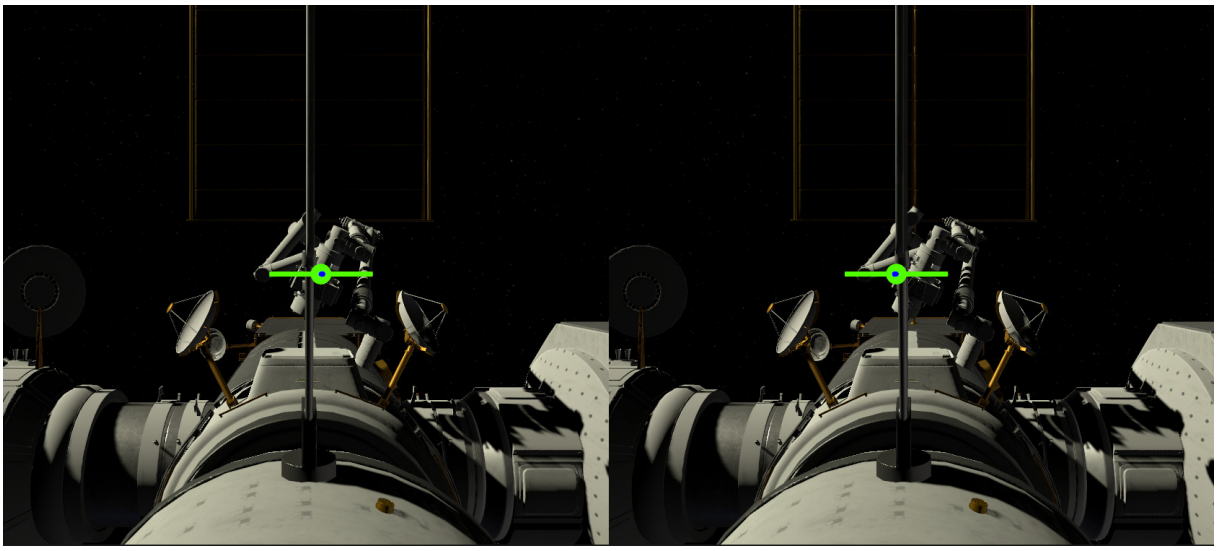

Figure 4.8: The player view as seen in the dual display setup from the VR headset. An artificial horizon is located in the center for the initial calibration step.

placing two virtual cameras as part of the Unity game object `player`. The `player` game object contains all the information needed to orientate the attached cameras in the scene and therefore to represent the position of the head in the virtual world. The virtual cameras are both oriented in the z-direction and are located at the `player`'s head height with a distance of 6 cm between them. Therefore, the two cameras represent the two different views of the human eyes with a natural offset to each other. On closer comparison of the two views in Figure 4.8, this offset can easily be determined (especially when looking at the edges of each view). When wearing the glasses, this offset and difference in camera views results in a natural three-dimensional sense of depth.

In addition, Figure 4.8 shows an artificial horizon (green) in the center of each view. This serves as a further tool for orientation and is required for the initial calibration step of the headset. The lines to the sides visualize the difference in roll, while the blue dot in the center of the artificial horizon represents the difference in pitch and yaw. If there exists a deviation in the pitch or yaw angle, the blue dot will no longer appear centered, but will instead appear as an extended line pointing in the direction needed for calibrating the headset. This calibration step is required to match the inertial sensor coordinate frame (in this case the IMU) to the coordinate frame provided by the optical tracking system.

4.3.3 Software environment in ROS

ROS 2 Iron (introduced in Section 4.2) is chosen as the software framework foundation of the project to process the various sensor and camera inputs and to determine the head pose. Different ROS nodes are utilized for the sensor data acquisition, the sensor fusion and the transmission of the pose to Unity. In the following, a brief overview of the utilized ROS nodes and topics is presented, which are explained further in the following sections.

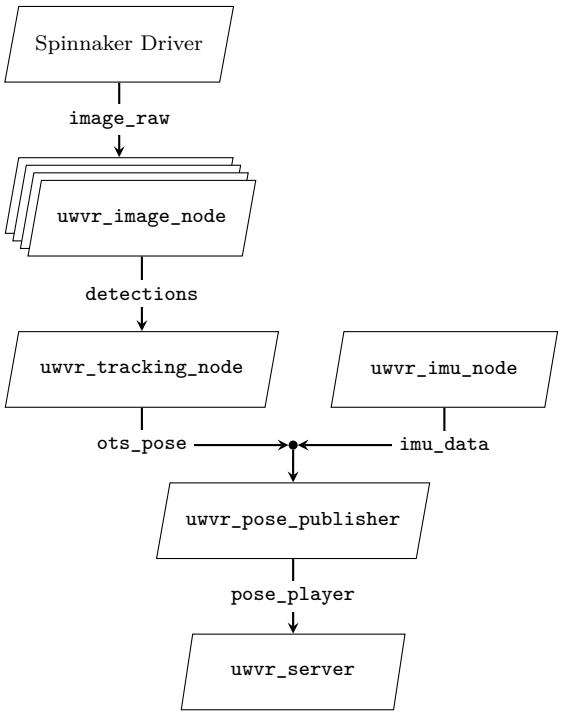

Figure 4.9: Overview of the ROS nodes utilized for the overall tracking.

The Spinnaker Camera Driver for the FLIR BFS-PGE-23S3C-C cameras is embedded as a ROS node in the project. The launch file `cameras.launch.py` starts the driver of the cameras, thereby leading the four cameras to begin publishing their raw image under the ROS topic `image_raw`. These topics are subscribed by four ROS nodes `uwvr_image_node` for every camera, which themselves publish the topic `detections` containing the positions of the AprilTags found in the images. Furthermore, the four `detections` topics are subscribed by the ROS node `uwvr_tracking_node`, which calculates position and orientation from the received data. This pose is published under the `ots_pose` topic. In parallel, the ROS node `uwvr_imu_node` receives serial data from the inertial measurement unit and publishes the acceleration and orientation from the IMU under the `imu_data` topic Both `ots_pose` and `imu_data` topics are subscribed by the `uwvr_pose_publisher` ROS node, that integrates the sensor fusion and calculates the final pose from IMU and OTS data. The final pose is then being published as the ROS message `PosRot.msg` under the `pose_player` ROS topic for a further transmission to Unity. A `PosRot.msg` consists of eight data fields that contain information about the headsets pose (position and orientation) and calibration status. Data fields one to three represent the position (x, y, z) of the headset as 32 bit floating point numbers, while the next four values describe the orientation or angle of the head pose as a quaternion (x, y, z, w). The last data field is a boolean value that holds true, if the inertial and optical tracking has been calibrated. Finally, `pose_player` is subscribed by the `uwvr_server` node, which transmits the data to Unity.

The launch file `uwvr_tracking.xml` starts all nodes related to the tracking system. This includes the nodes for the optical tracking (`uwvr_image_node` and `uwvr_tracking_node`), as

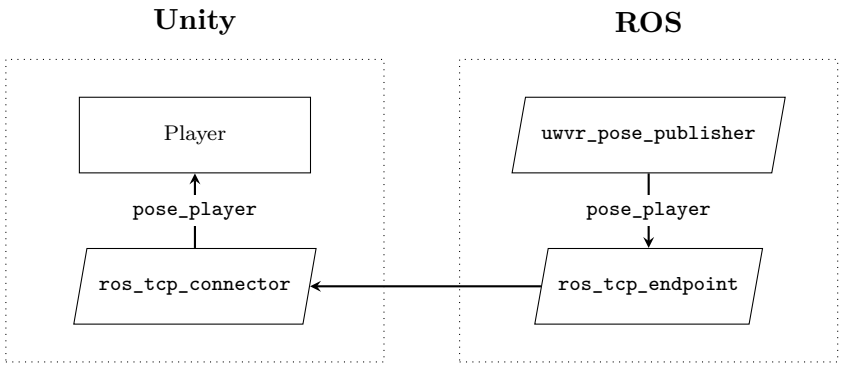

Figure 4.10: Visualization of the communication between ROS-TCP-Endpoint and ROS-TCP-Connector. (Reproduced based on [30])

well as the inertial tracking systems (`uwvr_imu_node`), the publishing node for the sensor fusion (`uwvr_pose_publisher`) and the nodes for transferring the pose to the Unity application (`uwvr_server`).

4.3.4 Connection between ROS and Unity

For transmitting the pose data of the headset to the `player` game object in the Unity scene, two approaches were tried out that both are based on TCP as the transmission protocol.

Unity Robotics Hub

The Unity Robotics Hub is a GitHub repository for robotic simulation in Unity, containing tools to visualize ROS based applications. One of the components of the Robotic Hub is the establishment of a connection between ROS and Unity. This connection is necessary to graphically display processes in ROS, as described in [26]. [30]

ROS-TCP-Endpoint. The Endpoint[2] is the server script that runs as a node on the ROS system. The endpoint is started by the launch file `endpoint.py` that specifies the IP address and TCP port. An TCP endpoint node subscribes to all ROS topics on a system. The basic functionality of the ROS-TCP-Endpoint is that all ROS topics of the ROS system are getting serialized and made available via the defined IP address via TCP. [30]

ROS-TCP-Connector. The Connector[3] is the client script running on the Unity side. After all ROS topics have been serialized by the endpoint over a specified IP address, the ROS-TCP-Connector is set up to subscribe or publish to those topics.

Worth mentioning is that serializing all ROS topics of a system and transmitting them over TCP to enhance the publisher/subscriber communication with Unity also comes along with drawbacks in terms of transmission speed and latency. Furthermore, this functionality is a bit

[2] `https://github.com/Unity-Technologies/ROS-TCP-Endpoint`
[3] `https://github.com/Unity-Technologies/ROS-TCP-Connector`

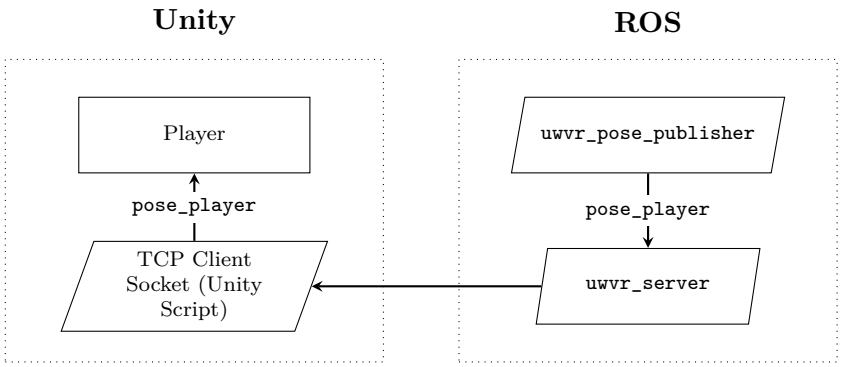

Figure 4.11: Visualization of the lightweight TCP communication solution between ROS and Unity. (Reproduced based on [30])

over-proportionate for the project, as only a one-way transfer of eight data fields from ROS to Unity is required. Therefore, a more lightweight TCP transfer solution from ROS to Unity was developed.

Lightweight ROS Unity TCP connection

In the following, a lightweight solution for a one-way transfer of the `PosRot.msg` data from ROS to Unity is presented.

The overall workflow of the data is visualized in Figure 4.11 and works similar to the Unity Robotics Hub, but without serializing every available ROS topics over TCP. The TCP Server Socket is a ROS node (`uwvr_server`) that, as the name already suggests, hosts a TCP socket for a client to connect to. The socket is opened on the localhost (`127.0.0.1`) since the client is expected to run on the same computer. It is initialized as a ROS subscriber to the `PosRot.msg` message under the `pose_player` ROS topic. This topic is published by the tracking system, which is represented by the `uwvr_pose_publisher` ROS node. Upon the publication of each new pose the TCP server node is responsible for receiving the pose and calibration information. Subsequently, the server node disseminates this information to the connected client via TCP.

In this particular instance, the TCP client is initiated through a Unity script, which is attached to the Unity game object `player`. It extracts the pose information and adjusts the position and orientation of the player accordingly. The calibration information is used to determine whether the artificial horizon should be displayed.

4.3.5 Optical tracking

The Spinnaker Camera Driver for the FLIR BFS-PGE-23S3C-C cameras is embedded as a ROS node in the project. The initiation of the four cameras is executed through the utilization of a designated launch file `cameras.launch.py`, which comprises the designated specifications and parameters for each camera. These parameters include, but are not limited to, exposure time, image resolution and compression options. Therefore, upon the start of the launch file, the four cameras are initialized with their launch parameters and begin publishing a timestamped raw

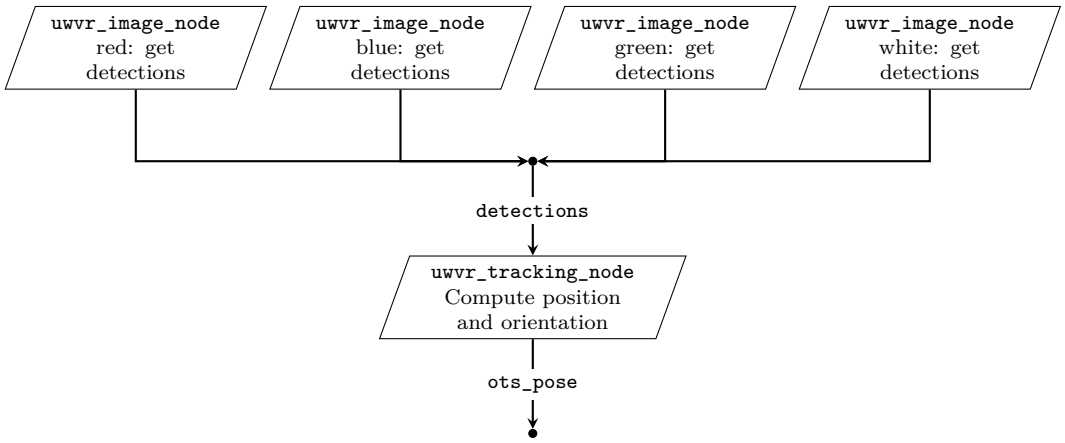

Figure 4.12: Visualization of the ROS nodes utilized for the optical tracking system.

image under the ROS topic `image_raw`.

This topic is subscribed to by the `uwvr_image_node` ROS node, which is started for each of the four cameras, as seen in Figure 4.12. As mentioned in Section 4.3.1, each camera is identified by a colored marker positioned at each of the four corners of the cubic structure, thereby yielding the following identification codes: red, blue, green and white. Each of the `uwvr_image_node` ROS nodes employs the AprilTag library to identify the various fiducial tags present in the raw images. These detections are concluded in a ROS message `Detections.msg` as a list of x and y values identifying the tags position in the image, along with the original timestamp from the `image_raw` topic. Subsequently, this message is published under the topic `detections` of their respective camera for further processing and pose computation in `uwvr_tracking_node`. The detections are evaluated in the ROS node `uwvr_tracking_node`, and the positions of the cubic structure and the headset in relation to the camera are determined using the AprilTag library. The tracking node updates the message whenever a new position and rotation is computed under the topic `ots_pose`.

The calculated pose is oriented to the world coordinate system of optical tracking, which will be referred to as the OTS frame or OTS coordinate system in the following. This coordinate system is located in the center of the cubic structure. The OTS frame is a right-hand coordinate system. If one stands behind the cubic structure as visualized in Figure 4.6, the y-axis points forwards, the z-axis shows upwards and the x-axis points to the right.

Improvements for faster image detection and pose computation

In contrast to the initial version of the optical tracking, modifications are implemented to achieve higher data rates and reduced delay. With the initial optical tracking (referred to as *OTS old* in the following), the image data was not yet processed in the respective `uwvr_image_node` in different threads, but together in the `uwvr_tracking_node`. This involved waiting for all four cameras to transmit new image data and subsequently using the combined image data to calculate the position and orientation of the headset.

By making the first processing step of the camera data in the image node independent of

each other for the new version of optical tracking (referred to as *OTS new*), faster tracking times are achieved. Furthermore, the tracking node has been modified to calculate a position and orientation already when the AprilTag data of only one camera is received. While this may result in less accurate results, significant performance improvements are expected.

4.3.6 Inertial tracking

As already mentioned in the Hardware description 4.3.1, the CEVA Hillcrest Laboratories firmware SH-2 optimized for AR/VR applications is running on the BNO085 inertial measurement unit [3]. This firmware offers different data outputs that result from internal sensor fusion and calibration algorithms for AR/VR purposes. To subscribe to those different data types, the QtPy RP2040 microcontroller is connected to the IMU breakout board. It sets up the firmware running on the IMU and receives the desired data. Overall, the following data outputs are utilized from the BNO085 sensors internal processing:

- AR/VR Stabilized Game Rotation Vector
- Linear Acceleration

Given the optimization of the sensor for AR/VR applications, an important output of the device is the provision of a reliable estimation of the headset orientation that incorporates the inertial measurement unit. There are different rotation vector outputs provided by the SH-2 firmware (see the reference manual [3]) with different optimizations. Format of the data is a unit quaternion which indicates the difference in the rotation vector to the orientation during the initialization.

The *AR/VR Stabilized Game Rotation Vector* delivers a self-correcting position estimation utilizing the onboard accelerometer and gyroscope measurements. Accumulated errors in the integration of the gyroscope measurements are corrected by estimations from roll/pitch angles delivered by the accelerometer data. The major difference to the *AR/VR Stabilized Game Rotation Vector* is that the magnetic field estimation from the magnetometer is not taken into account. This has the advantage of ensuring the stability of the rotation vector in environments where the Earth's magnetic field may be influenced by external magnetic fields. Despite the fact that the frame of the headset is constructed from aluminum, a non-magnetic material, the integration of cables and breakout boards within the device can potentially induce distortions in the magnetic field. A key feature of the *AR/VR Stabilized Game Rotation Vector* output is to minimize the occurrence of sudden jumps in the angles. For an AR/VR application, sudden jumps in the orientation would severely restrict the user experience and cause symptoms such as motion sickness. According to the BNO085 data sheet [4], the velocity of the system is taken into account in a process referred to as "AR/VR stabilization" to prevent these jumps in the estimation. [3, 4]

Another data output utilized for this project is the *Linear Acceleration*. This output results from the accelerometer data which has been improved by calibration. Furthermore, the gravitational acceleration is calculated and subsequently subtracted from the acceleration vector. This results in a linear acceleration vector of the sensor. The format of the output is a three-dimensional vector with the linear acceleration in x, y, z direction of the sensors coordinate frame in $^{m}/_{s^2}$. [4]

The different data outputs are enabled via the I2C connection from the microcontroller through feature requests. For every request, the ID of the report, as well as the report interval for the output, has to be initialized. Whenever new sensor events are available, those are received by the QtPy microcontroller and serialized for further transmission, corresponding to the data output of either rotation vectors or linear accelerations. Due to the onboard filtering of the rotation vector related data of the SH-2 firmware, the outputs received on the microcontroller result in stable values without noticeable jumps or wrong values. Therefore, the rotation vector data is forwarded directly to the PC without further processing on the microcontroller.

On the other side, the linear acceleration outputs are characterized by deviations and larger jumps, even in the absence of external forces on a uniform surface. Since the accelerometer is the only sensor in the IMU that could potentially be used to assist position determination, this data cannot be improved by sensor fusion of different IMU sensors themselves. However, in order to smooth this data to prevent sudden and false jumps in the data, a digital low-pass filter for the accelerometer measurements is implemented directly on the microcontroller.

Low-pass filter for accelerometer measurements

As previously stated, the low-pass filter implemented on the QtPy microcontroller serves the function of smoothing the overall data and eliminating the higher frequencies that appear as noise in the dataset. The low-pass filter is implemented as shown in Algorithm 1 in Section 3.2. As this is a first-order infinite-impulse-response filter, it is identical to an exponential weighted moving average (EWMA) filter as mentioned in Section 3.2 [21]. Recalling equation 3.2, for each incoming value of the accelerometer x_t, a new iteration of the low-pass filter is computed to determine the filtered value y_t:

$$y_t = y_{t-1} + \alpha \cdot (x_t - y_{t-1})$$

This process is repeated for every axis. It was determined that a smoothing factor $\alpha = 0.1$ results in an adequate improvement of the outcome data.

Another common method for smoothing high-noise datasets is to compute the average over the last k values (equation 3.3). This simple moving average (SMA) filter eliminates rapid changes in the data, thereby preserving lower frequencies. However, it should be noted that this approach may be less efficient in terms of computing power and memory. Specifically, implementing a queue for the last k values and computing the average for each step can impose additional overhead.

As illustrated in Figure 4.13, both methods are compared to the raw acceleration data. As previously mentioned, a first order low-pass filter with a smoothing factor of $\alpha = 0.1$ was employed. In contrast, the simple average filter was implemented with a queue size of $k = 35$. The graph indicates that both methods yield satisfactory results. However, the slightly higher queue size of the simple average filter leads to a stronger smoothing effect, but also to a greater inertia of the system and therefore a higher delay compared to the raw data. The low-pass filter, while yielding analogous outcomes, was selected due to its more effective and resource saving design to ensure optimal performance of the system.

Subsequently, this filtered data is transmitted to the PC via the USB serial connection.

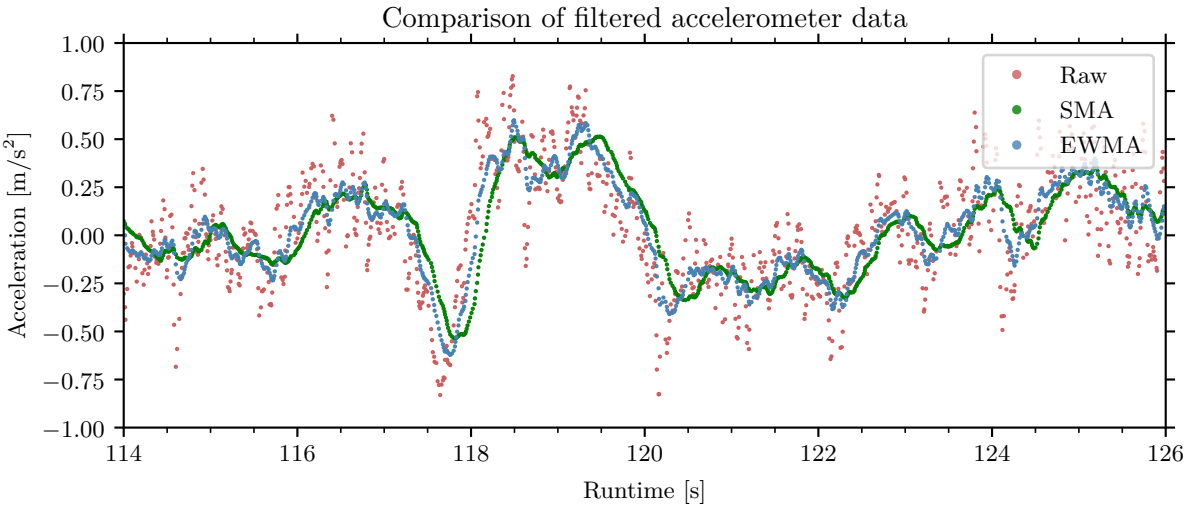

Figure 4.13: Raw acceleration in x-direction compared to EWMA and SMA filtered values.

Serialization and transmission to PC

As already mentioned, whenever new sensor data are available, they get serialized and transmitted to the PC via the UART serial protocol. When the sensor supplies new data, this is either forwarded raw (rotation vector) or low-pass filtered directly on the microcontroller (acceleration), depending on the data type. In the latter case, the filter's acceleration output is transmitted after the filtering. The baudrate of the UART connection is set to the maximum of 921600, which stands for 921600 bits transferred per second, to ensure a low data transmission time and therefore a low latency. The data is then sent in the following format: `[A/R] [data separated by spaces]`. In explanation, the first character is either an `A` (for acceleration) or an `R` (for rotation). This simple format allows for a fast serialization and deserialization on the receiving end, while the start character prevents errors with out-of-order messages. On the receiving end, one can readily filter for the first character and subsequently evaluate the message. In the event that a message is incomplete, it is declared invalid and therefore discarded.

Deserialization and ROS publisher

The deserialization process is executed by the ROS node `uwvr_imu_node`. Initialized with a sampling rate of 200 Hz, the node operates in a state of readiness, connected to the UART port and tasked with the deserialization of incoming IMU data. After this, the data is appended to the ROS message `ImuData.msg` containing the acceleration and rotation vectors, as well as a timestamp and an identifier for the changed part of the message. This message is then published under the `imu_data` topic for utilization in the following sensor fusion phase.

4.3.7 Sensor fusion

Since the IMU data is transferred to the PC and the OTS system has calculated the pose in a ROS node utilizing the AprilTag library for object tracking, the final determination of the player pose is now carried out based on the IMU and OTS data. The sensor fusion is implemented in the ROS node `uwvr_pose_publisher`, which then publishes the final pose of the headset in the ROS message `PosRot.msg` under the `pose_player` ROS topic. As previously stated in Section 4.3.4, this topic is then subscribed by a ROS node, responsible for forwarding the data to Unity via a TCP connection. Subsequently, the pose of the headset is displayed accordingly in the virtual Unity environment.

In this context, the sensor fusion refers to the integration and combination of all elements of the tracking system in a manner that ensures optimal functionality and robustness, to apply with the initial design specifications. Therefore, the primary objective remains to benefit from the faster publishing rates of the IMU data and enhance the tracking reliability by compensating for possible failures associated with the optical tracking system. However, prior to their integration within the sensor fusion process, it is necessary to transform the IMU and OTS data into a shared coordinate system. Ideally, this should correspond to the system used in Unity, as the transmitted data can then be used directly in Unity without further transformation needed.

Transforming the coordinate frames

As already mentioned in Section 4.3.2, the coordinate frame employed in Unity is left-handed, which is in direct contrast to the right-handed coordinate frame utilized by the OTS. The optical tracking system outputs a 3D vector containing the translation and a quaternion that represents the orientation of the headset in the right-handed OTS coordinate frame.

To ensure that the determined rotation and translation matches the subsequent coordinate system of the Unity environment, it must be transformed using a rotation matrix. Although this process can be combined into a single three-dimensional spatial transformation matrix, it was calculated separately for position and rotation in this work, since all matrices and vectors already existed in a 3×3 (or respective 3×1) form:

$$\boldsymbol{R}_{player} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \qquad \boldsymbol{P}_{player} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

A first-rotation matrix is employed in the sensor fusion algorithm as $\boldsymbol{R}_{OFFSET\_WORLD}$. The purpose is to rotate the final computed orientation of the player into the desired direction in the virtual environment. As illustrated in Figure 4.8, the player's orientation has been rotated by 180 degrees around the y-axis in the Unity frame, resulting in the following rotation matrix:

$$\boldsymbol{R}_{OFFSET\_WORLD} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.1}$$

In the following, adequate rotation matrices for the OTS and IMU sensors are presented, to transform their respective outputs into the Unity coordinate frame. The position vector and

rotation matrix in the OTS frame are rotated into the Unity frame by utilizing the following rotation matrix:

$$\boldsymbol{R}_{OTS} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.2}$$

The same procedure may be employed to transform the axes of the IMU frame into the coordinate frame of the virtual environment. The following transformation matrix applies when the headset is oriented in the y-axis of the OTS frame to transform it into the Unity frame:

$$\boldsymbol{R}_{IMU} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

It is important to note that the IMU coordinate frame is linked to the orientation of the headset in space. This is in contrast to the coordinate frame of optical tracking system, which is determined by the actual orientation of the platform. Since this platform is installed not to be movable, the OTS frame is fixed as well as the Unity frame. Consequently, the difference in the orientation of the headset to the platform must also be determined, before the IMU frame can be transformed into the Unity frame. Therefore, the objective is to rotate the IMU coordinate system to the aforementioned orientation in the y-axis of the OTS frame, followed by a transformation using $\boldsymbol{R}_{IMU}$. When working with quaternions representing rotations, this rotation into the y-axis is achieved by multiplying with the quaternion representing the difference of the coordinate frames:

$$Q_{OTS} = Q_{IMU} \cdot Q_{diff} \tag{4.4}$$

Consequently, the rotation Q_{diff} must be determined as part of an initial calibration step. This enables the utilization of the other data from the IMU, such as acceleration, in conjunction with the positional data from the optical tracking system.

Calibration of IMU and OTS rotation vector

The objective of the calibration of the IMU is to compute the difference in the sensor rotation vector from the IMU to looking in straight y-direction in the OTS coordinate frame. Therefore, the IMU output is calibrated to a fixed frame for its data, independent of changes in its mounting position or orientation. Since the IMU output coordinate frame is not fixed in the environment and resets itself for every new power cycle, it may need to be calibrated accordingly at the beginning of every operation after it received power. Following from equation 4.4, Q_{diff} is calculated from solely knowing Q_{OTS} and Q_{IMU}, which results in the following equations (when representing rotations with quaternions):

$$Q_{diff} = Q_{OTS} \cdot \text{inv}(Q_{IMU}) \tag{4.5}$$

$$Q_{diff} = Q_{OTS} \cdot \text{conj}(Q_{IMU})/\text{abs}(Q_{IMU}) \tag{4.6}$$

In order for Q_{diff} to indicate the difference of the IMU coordinate frame to the y-axis in the OTS frame, it is optimal to calculate it when the headset is precisely pointing in the y-direction

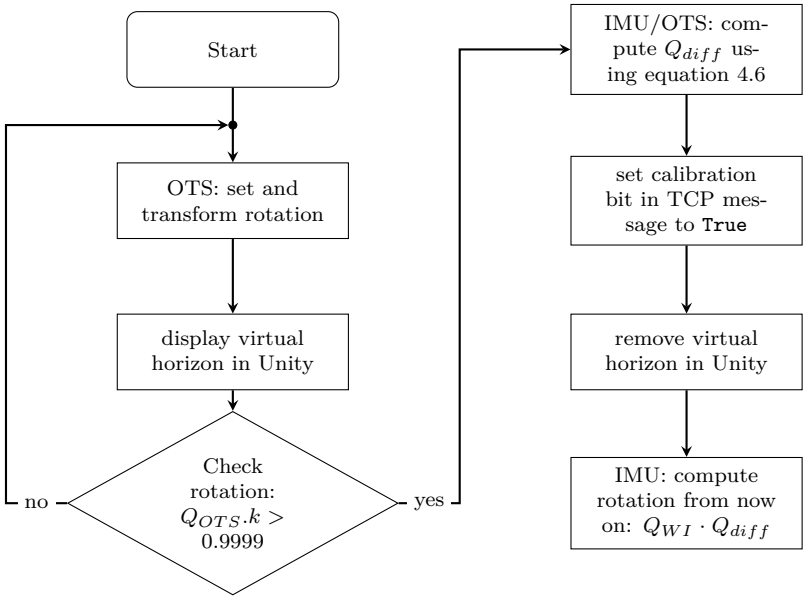

Figure 4.14: Overview of the calibration step to compute the difference in orientation of the IMU and OTS coordinate frames.

within the OTS frame. Accordingly, the objective of the algorithm is to initially calculate the orientation and position exclusively through the optical tracking system until the headset is oriented directly in the y-direction. By displaying a virtual compass in the headset as seen in Figure 4.8, the difference to the desired orientation of the headset is indicated until the user has achieved to bring the headset into position.

Therefore, an appropriate indicator is needed for determining when the alignment of the headset has been achieved with sufficient precision. As Q_{OTS} also indicates the difference from the alignment in the y-direction, the parameter k of Q_{OTS} is utilized as a sufficient indicator, as it applies $k = 1$ for a perfect alignment. It was chosen that an alignment with $k > 0.9999$ is sufficiently accurate.

Once the appropriate orientation is achieved, Q_{diff} is calculated following equation 4.6 and the virtual compass is disabled in the virtual environment by setting the calibration value to `False` in the TCP communication between ROS and Unity. From this point onwards, the orientation of the system is determined by utilizing the orientation measurement of the IMU, which can then be employed in the subsequent sensor fusion process. The complete calibration algorithm to calculate Q_{diff} is presented in Figure 4.14.

Sensor fusion with Kalman filter

The principal component of the sensor fusion is the utilization of a Kalman filter for the combination of IMU and OTS tracking data. In order to achieve the objective of a robust head tracking with minimal delay and lag, following decisions were made regarding the selection of data to be utilized for the Kalman filter.

- IMU rotation vector output for orientation determination
- OTS and IMU Kalman filter sensor fusion for position determination

Orientation determination The *AR/VR Stabilized Game Rotation Vector* output was found to return highly accurate and stable orientation values of the BNO085 IMU due to the internal sensor fusion for AR/VR applications provided by the CEVA Hillcrest SH-2 firmware [3]. As a result of the calibration algorithm presented in Section 4.3.7, the output data is independent of the sensors mounting position in the headset or its initial orientation. Additionally, it is synchronized to the coordinate frame the optical tracking. It was therefore determined that in order to achieve the aforementioned objectives in terms of robustness, it would be sufficient to rely solely on the output of the IMU in regard to the tracking of rotations, after the completion of the calibration phase. As illustrated in Figure 4.14, this process is presented in the subsequent computation of the IMU following a successful calibration of the rotation data. This approach allows the tracking system to benefit from the sensor fusion already performed on the IMU, as well as the higher output rates in comparison to the OTS. Furthermore, it avoids potential distractions from the optical tracking system, which could otherwise result in lags or error-prone data.

Position determination In order to determine the position, it is not sufficient to rely solely on the IMU *Linear Acceleration* output. This would result in significant drifting due to the two-time integration of the acceleration. Therefore, the OTS data is essential for providing the absolute position of the headset without being influenced by drift. However, there are also some limitations to consider. For instance, incorrect data may be generated when the optical tags are covered or not identified correctly, and the update rates may be slower, potentially leading to sudden changes and jumps in position. Subsequently, these data outputs containing information about the translational movement and position of the headset are integrated into a Kalman filter to compensate the inherent limitations of both systems, thereby resulting in a more robust overall position tracking.

Since using the Kalman filter only for the translational tracking of the headsets position, a linear Kalman filter is utilized for this purpose. In consequence, the corresponding equations of motion are as follows:

$$p(t) = p_0 + v\Delta t + \frac{1}{2}a\Delta t^2 \tag{4.7}$$

$$v(t) = v_0 + a\Delta t \tag{4.8}$$

$$a(t) = a_0 \tag{4.9}$$

The system state is therefore chosen to consist of position p_k and velocity v_k at a time k.

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} \tag{4.10}$$

The headset position data returned by the optical tracking system is chosen for the measurement $\boldsymbol{z}_k$, while the acceleration $\boldsymbol{a}_k$ is added to the system as an additional input u_k. Recalling

equation 3.13 for the *a priori* state prediction

$$\hat{\boldsymbol{x}}_{k+1}^{-} = \boldsymbol{A}\hat{\boldsymbol{x}}_k + \boldsymbol{B}u_k,$$

$\boldsymbol{A}$ and $\boldsymbol{B}$ are determined to match the corresponding system state. For clarity, the system state was first identified for the one dimensional case, with acceleration a_k representing the system input u_k.

$$\hat{\boldsymbol{x}}_{k+1}^{-} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} a_k \tag{4.11}$$

This can then be calculated for the three-dimensional case as well. Furthermore, the system state is selected in a manner analogous to that employed in the one-dimensional case, whereby the three dimensions are expressed successively after another.

$$\hat{\boldsymbol{x}} = \begin{bmatrix} p_x \\ v_x \\ p_y \\ v_y \\ p_z \\ v_z \end{bmatrix} \tag{4.12}$$

Moreover, matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are expressed analogous to the single one-dimensional cases written in a diagonal form:

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & \Delta t \end{bmatrix}$$

Recalling equation 3.15 for the innovation

$$\boldsymbol{y}_k = \boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^{-},$$

the matrix $\boldsymbol{H}$ represents the conversion from the system state to the measurement $\boldsymbol{z}_k$, containing the position of the headset as determined by the OTS.

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.13}$$

The final step is to identify the initial parameters for the covariance matrices $\boldsymbol{P}_0$, $\boldsymbol{R}$ and $\boldsymbol{Q}$. A common practice for the state error covariance matrix $\boldsymbol{P}_0$ is to initially select larger

numbers for the diagonal entries. This signifies a higher degree of uncertainty regarding the initial system state, which is subsequently reduced in the following correctional phases as the system state uncertainty is diminished. Accordingly, the initial selection of $\boldsymbol{P}_0$ was made as follows:

$$\boldsymbol{P}_0 = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{4.14}$$

As already mentioned in Section 3.3.3, the measurement error covariance $\boldsymbol{R}$ influences the degree to which the correction is trusted, while the process noise covariance matrix $\boldsymbol{Q}$ influences the extent to which the estimation is trusted. There exist rules of thumb for determining initial values, such as selecting the variances of the sensors at rest. However, for achieving better results for the sensor fusion, better working values of $\boldsymbol{R}$ and $\boldsymbol{Q}$ are obtained through a process of trial and error. In this particular instance, the values for $\boldsymbol{R}$ and $\boldsymbol{Q}$ were initially defined using the variances and subsequently refined until a satisfactory result was obtained. Consequently, the measurement error covariance $\boldsymbol{R}$ comprises the diagonal entries of the initial variance of the OTS and afterward the better working value of 0.001.

$$\boldsymbol{R} = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \tag{4.15}$$

The process noise covariance matrix $\boldsymbol{Q}$ was determined by utilizing the Python library *filterpy* with the function `Q_discrete_white_noise`. Initially, the variance of the IMU was used as the input, and subsequently, custom values were employed. Thus, the value 1 was ultimately taken as the input of the function, resulting in the following process noise covariance matrix:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}^* & \mathbf{0}_{2,2} & \mathbf{0}_{2,2} \\ \mathbf{0}_{2,2} & \boldsymbol{Q}^* & \mathbf{0}_{2,2} \\ \mathbf{0}_{2,2} & \mathbf{0}_{2,2} & \boldsymbol{Q}^* \end{bmatrix} \tag{4.16}$$

$\boldsymbol{Q}^*$ is defined as following:

$$\boldsymbol{Q}^* = \begin{bmatrix} 9.80121907e-09 & 2.45030477e-06 \\ 2.45030477e-06 & 6.12576192e-04 \end{bmatrix} \tag{4.17}$$

In this instance, the values of $\boldsymbol{Q}$ were selected lower than the entries of $\boldsymbol{R}$, thereby yielding a stronger weighting of estimation and input (of the inertial measurement unit).

Preventing drift

As previously stated, utilizing the accelerometer data exclusively for position tracking results in a drift of position due to the double integration of the data. It is possible that this case may still occur in instances where the OTS fails to deliver new data. This is particularly relevant in scenarios where visual contact from the cameras to the tags is obstructed, or the headset is operated at the edge of the visible tracking area. In such circumstances, the calculated position of the headset may differ from the real position, resulting in a negative impact on the user experience. Given that wrong feedback of this kind could potentially lead to dangerous situations in an underwater environment, it is mandatory to reduce the degree of motion possible when the optical tracking is unavailable for an extended duration. In the event of the OTS data being unavailable for a duration of 10 Kalman filter cycles, the system's velocity is reduced by 10% for each subsequent cycle. This results either in a reduction of the velocity until a halt of the system is achieved or until the OTS returns data again.

Complete sensor fusion algorithm

Figure 4.15 illustrates the aforementioned steps, offering a high-level overview of the entire head tracking and pose determination process. The initial input block signifies the callback functions that are utilized for subscribing to the sensor data. The sensor data are encapsulated in ROS messages under the topics `ots_pose` and `imu_data`. These topics are published by the respective ROS nodes `uwvr_tracking_node` (Section 4.3.5) and `uwvr_imu_node` (Section 4.3.6).

Subsequently, the calibration steps outlined in Section 4.3.7 are executed until the headset is centered and the IMU rotation is calibrated to the OTS rotation. The pre-filtered IMU rotation data is then employed for the purpose of orientation tracking. Furthermore, the position is computed by the Kalman filter sensor algorithm that was previously mentioned.

In the final step, the calculated pose is published under the ROS topic `pose_player`, resulting in a transmission from the ROS node `uwvr_server` via the TCP connection to Unity for the visualization in the headset.

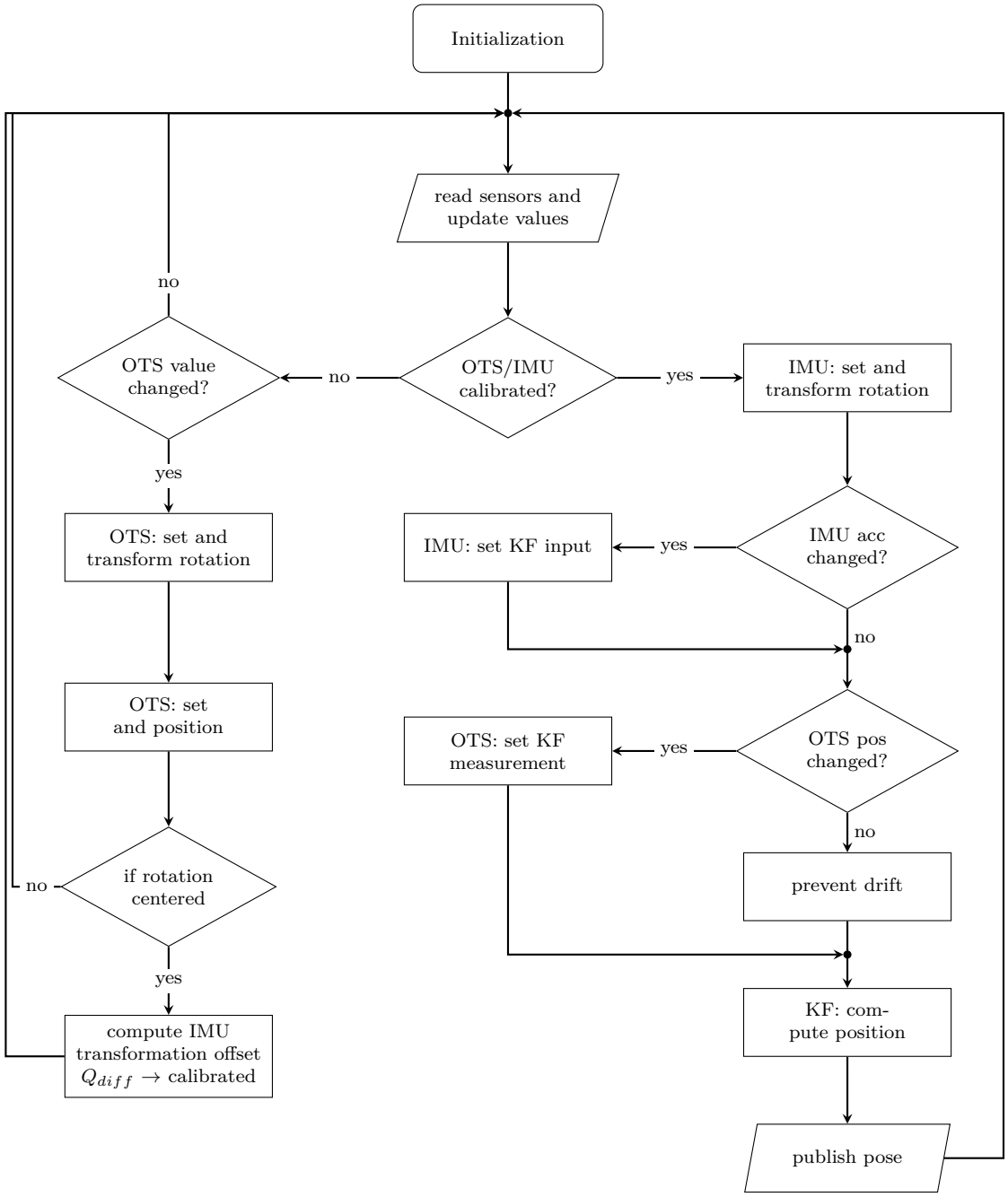

Figure 4.15: High-level overview of the sensor fusion algorithm for the determination of the final headset pose implemented in the `uwvr_pose_publisher` ROS node.

Chapter 5

Evaluation

In the subsequent chapter of the thesis, an evaluation of the underwater VR headset and subsequent tracking approaches is conducted. This evaluation is based on performance measurement tests and a practical field-test at ESA's Neutral Buoyancy Facility (NBF) at the European Astronaut Centre (EAC) in Cologne. Finally, the results obtained from both experimental and practical tests are discussed.

5.1 Robustness testing

The objective of the robustness evaluation tests is primarily to assess the performance and general functionality of the underwater VR headset, which design and development was presented in the preceding Chapter 4. The evaluation process was primarily divided into two categories:

- Delay
- Robustness

In the following section, a detailed description of the configurations and execution of the aforementioned test categories is provided.

5.1.1 Test setup description

As previously stated, the experiments are classified into two categories, which consequently also define the experiments that are to be executed. The experiments include the measurement and comparison of the delay through the system and an evaluation of the tracking robustness. The structure and implementation of these tests are described below.

Measuring time delays through the system

The objective of this experiment is to calculate the time difference from the original recording of the sensor data to the final display of the pose in Unity. Once this time difference has been determined, it corresponds to the delay between the user's input, in this case a head movement and the actual display in the virtual environment in the VR headset. There exist

various solutions for determination of this difference. One option, which is utilized for this experiment, is to obtain a timestamp of the sensor data at the earliest possible point, ideally at the point of data collection. This timestamp is then tracked through the system in conjunction with the sensor data. Given that the optical tracking system and the inertial measurement unit provide two distinct sensor inputs to the system, these inputs result in different methodologies for computing and forwarding their respective timestamps.

For the optical tracking system, the Spinnaker driver for the FLIR cameras creates a timestamp once an image is taken. This stamp is attached to the raw image data and published under the `image_raw` topic, as mentioned in Section 4.3.5. Additionally, this time is incorporated into the ROS messages published by `uwvr_image_node` and subsequently `uwvr_tracking_node`, as visualized in Figure 4.12. Ultimately, the timestamp is received in the `uwvr_pose_publisher` node and subsequently transmitted to Unity, along with the final pose, through the TCP server connection described in Section 4.3.4.

However, it has to be acknowledged that the timestamp solely signifies the moment in time at which the image was captured. Consequently, any delays occurring prior to that moment must be taken into consideration as well. The most significant of these delays may be the exposure time, which is configured to 10 milliseconds.

In the case of the inertial measurement unit, the timestamp is appended to the `imu_data` ROS topic upon receipt of the serial message from the microcontroller and subsequently transmitted to Unity along with the OTS stamp. This also initiates an error in the delay measurement, namely the period at which the data is recorded on the IMU until it is transmitted and received via UART in the `uwvr_imu_node` ROS node. The two errors of the two systems and possible estimates of these values are referred to further in the upcoming results.

Furthermore, the timestamps are then compared with the real time at the state of the visualization, in this case in the Unity application, resulting in a time difference of the whole system. Once the delay has been calculated, it is printed out in a Unity log message. Subsequently, the log will be saved and analyzed, resulting in a dataset of delays over the runtime of the program. This dataset is further visualized in graphs.

Therefore, the comparison will focus on the different tracking methods such as the OTS and IMU systems. Another objective of the comparison will be the evaluation of the difference in transmission time between the different TCP connection applications between ROS and Unity.

Measuring data robustness

The second category of experiments is focused on the evaluation of the tracking robustness. Furthermore, the objective is to compare the original optical tracking system to the subsequent modifications to the tracking system, such as the sensor fusion with an additional inertial measurement unit. A potential approach for comparing these tracking systems involves executing the exact same motion sequence with the various tracking versions. Therefore, a comparison of the position and rotation movements are conducted. A graphical representation of the individual axes is a suitable approach for this purpose.

Nevertheless, reproducing the precise motion sequence on multiple occasions poses a considerable challenge. In this work, the challenge was addressed by leveraging the ROS bag functionality as mentioned in Section 4.2.2. By recording a motion sequence with a ROS bag containing

the four camera data, it is possible to replay the bag file multiple times, each time selecting a different version of the tracking. When the `--clock 200` option is selected during playback of the bag file, the recorded system time is replayed at a frequency of 200 Hz. Additionally, when the `use_system_time` parameter is set true for each designated node, the recorded system time is simulated as the current system time for all selected nodes.

Upon the playback of recorded camera data, the execution of various tracking methods occurs in real time, leveraging a shared data foundation and, consequently, a consistent movement trajectory. The calculated position and orientation data is stored in a ".csv" file for a subsequent display and comparison in graphs.

5.1.2 Test results

In the following section, the results of the experiments are presented and evaluated.

Measuring time delays through the system

As previously indicated in the setup description of this experiment (Section 5.1.1), the delay times are measured through the system and subsequently compared against the different versions. The experiments were carried out with the originally used version of the data transfer to Unity, the ROS-TCP-Connector, as well as with the newly implemented TCP server.

ROS-TCP-Connector At first, the delays were measured with the ROS-TCP-Connector selected as the transmission method between ROS and Unity. The time differences were plotted over a period of time, resulting in the following graph, as illustrated in Figure 5.1.

Figure 5.1 displays a 10 s section of the total running time of about 85 s for this delay measurement. The delay times of both the inertial measurement unit (red) and the new version of the optical tracking system (blue) utilizing the ROS-TCP-Connector are illustrated in the graph, while their corresponding mean and standard deviation are represented in Table 5.1.

Most notably is a clear distinction in the delay times between IMU and OTS new of about 54.31 ms. In addition, both measurements reveal the occurrence of periodic peaks in the delay time at an interval of about one second. The spikes reach up to a maximum 46 ms difference to the average delay time, as seen at the OTS measurements at a runtime of 39.5 s. Similar differences are obtained at the IMU measurements, with a peak of 35 ms deviation to the average at a runtime of 38.5 s. This additional delay, which has been previously cited of being up to 46 ms higher than the average delay time, results in the absence of new data during this interval. Therefore, these spikes result during the operation of the headset in a lag of the tracking performance, occurring every second.

	IMU	OTS new
Mean μ	13.92 ms	68.23 ms
Stdev σ	6.18 ms	9.26 ms

Table 5.1: Mean μ and standard deviation σ of the delay measurements utilizing the ROS-TCP-Connector according to Figure 5.1.

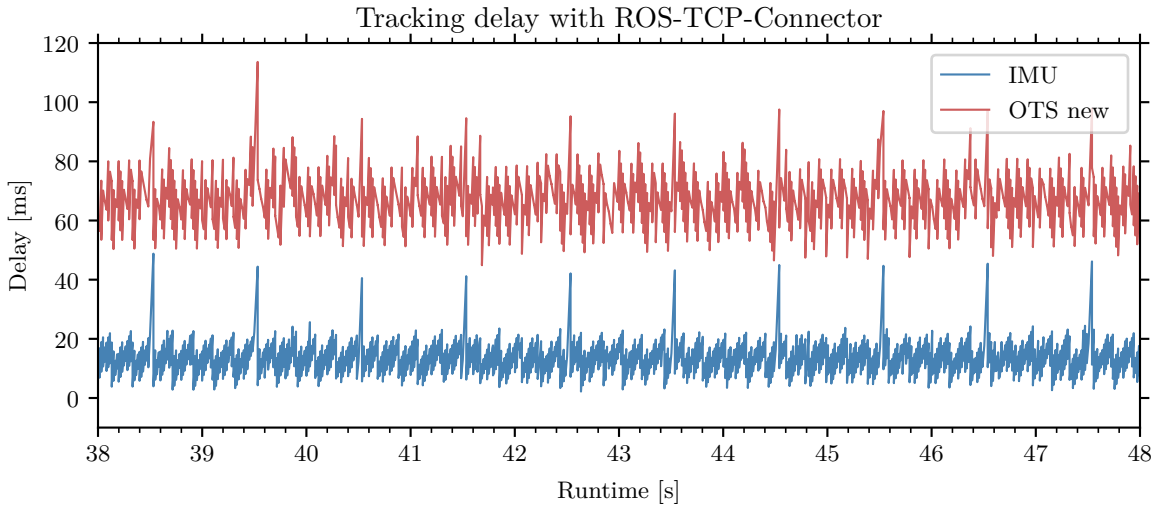

Figure 5.1: The overall delay of the camera tracking system in contrast to the IMU utilizing the ROS-TCP-Connector. The delay corresponds to the time from when the data is first received to when it is finally processed in Unity.

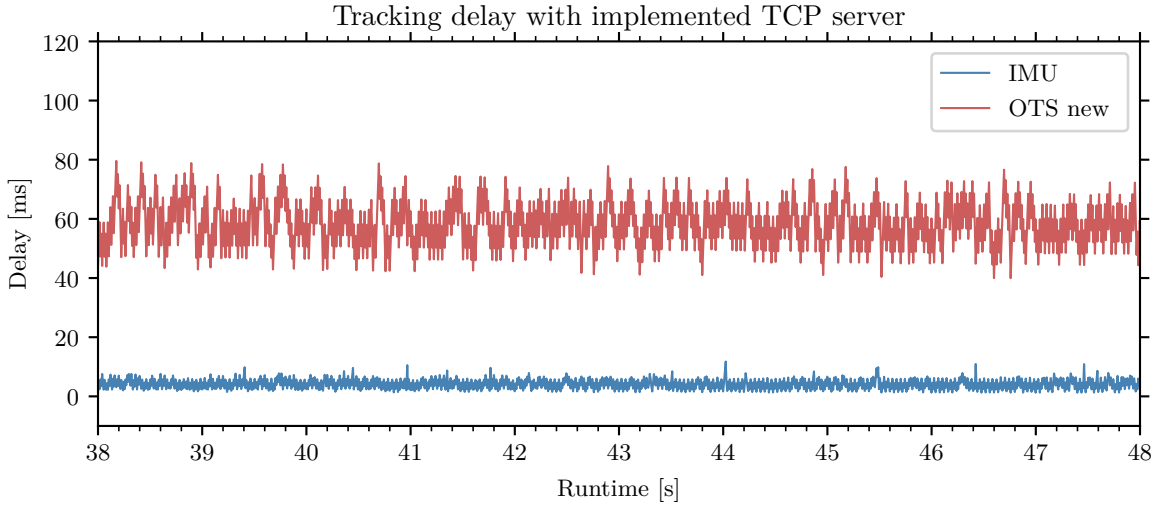

Figure 5.2: The overall delay of the camera tracking system in contrast to the IMU utilizing an TCP server. The delay corresponds to the time from when the data is first received to when it is finally processed in Unity.

TCP Server When utilizing the TCP server method of transmitting the timestamps to Unity, a general decline in the delay times is observable, as seen in Figure 5.2.

As illustrated in Table 5.2, the overall decline is also evident in the mean times of the measurements. The mean IMU delay reduced about 9.58 ms similarly to the OTS delay of about 9.53 ms. Consequently, the implementation of the new TCP server has led to a reduction

	IMU	OTS new
Mean μ	4.34 ms	58.70 ms
Stdev σ	1.66 ms	8.96 ms

Table 5.2: Mean μ and standard deviation σ of the delay measurements utilizing the TCP server node according to Figure 5.2.

in delay by approximately 9.5 milliseconds. The discrepancy in delay time between the IMU and the OTS is approximately 54.36 milliseconds, which is nearly equivalent to the result obtained from the ROS-TCP-Connector. This congruence can be attributed to the equal reduction in delay of 9.5 ms observed for both the IMU and the OTS values.

Moreover, the implementation of the new TCP server has effectively eliminated the periodic peaks that previously led to additional lag. This also becomes clear when comparing the standard deviations of both versions, which have now been significantly reduced. The standard deviation of the IMU has decreased by 4.52 ms while the OTS σ decreased by 0.30 ms.

Consideration of the measurement error In the following, the measurement errors previously addressed in Section 5.1.1 are evaluated and taken into account.

In case of the OTS delay measurement, the timestamp is generated and attached to the raw image data as it becomes available. The time error e_{ots} is not present in the current measured delays d_{ots}^{-}. Therefore, the true delay d_{ots} is defined as follows:

$$d_{ots} = d_{ots}^{-} + e_{ots}$$

Given that the exposure time of the cameras is set to 10 milliseconds, this can be regarded as a lower bound for the delay measurement error. Consequently, the following estimation for the true delay, d_{ots}, can be derived:

$$e_{ots} \geq 10\text{ms}$$
$$d_{ots} \geq d_{ots}^{-} + 10\text{ms}$$

For the IMU measurement on the other hand, the timestamp is generated, once the serial data is received from the microcontroller. Therefore, the time error e_{imu} represents the time difference of receiving the raw data on the microcontroller, processing it (see the Low-pass filter implementation in Section 4.3.6) and transmitting it over a serial connection to the workstation. The true delay d_{imu} for the IMU is defined analogous to the OTS measurements:

$$d_{imu} = d_{imu}^{-} + e_{imu}$$

In an attempt to find an upper limit for the measurement error, the time required for processing and data transfer via serial and back is measured. Consequently, a timestamp is generated upon the collection of IMU data and transmitted to the workstation. Upon receipt, the timestamp is immediately transmitted back, and the delay is subsequently measured. The measured difference in time is about 10.1 milliseconds. Given that the return step is generally not required, the error

is likely to be significantly lower. However, the 10 milliseconds represent an upper limit for the error, and further estimation of the true delay is possible by considering this upper limit.

$$e_{imu} < 10\text{ms}$$
$$d_{imu} < d^{-}_{imu} + 10\text{ms}$$

As a result, when computing the difference between the OTS and IMU delay times, those measurement errors can be neglected, since the calculated difference therefore represents a lower bound.

$$d_{diff} = d_{ots} - d_{imu} > d^{-}_{ots} - d^{-}_{imu}$$

In explanation, the improvements can be higher, but never less than the difference $d^{-}_{ots} - d^{-}_{imu}$.

Measuring data robustness

In the following section, the data from the various tracking approaches is presented graphically to illustrate disparities and improvements in terms of data robustness. A movement sequence of approximately 185 seconds was recorded. This sequence includes all types of translation and rotation movements. Sections from the recording are presented in the following graphs. Furthermore, the comparison is segmented into position and rotation data.

Position The initial comparison will be of the positional data, including the Kalman filter sensor fusion and the new and old versions of the optical tracking system. It is important to note that the Kalman filter sensor fusion uses the new OTS tracking as the basis for the inputs. To ensure consistency, the subsequent graphics are constrained to the x-axis of the headset's position.

Figure 5.3 illustrates the enhancements made to the optical tracking system. While the old OTS exhibits a higher number of outliers and deviations, the new OTS's data points are more coherent and smoothed, showcasing an increased stability and contiguity of the optical tracking system. It is also noticeable that the frequency of data points from the old optical tracking version is significantly lower, particularly around the 157-second mark, as one or more cameras were no longer be able to track the tags. Consequently, substantial lags and delays are experienced at this point, which has a considerable impact on the user experience.

In contrast, the sensor fusion tracking based on the Kalman filter delivers the highest and most consistent data rate, as it estimates new data points in the prediction step, if further OTS data is missing. This calculation also takes the acceleration data into account, to be able to react to changes at short notice. Additionally, the sensor fusion calculations are characterized by the smoothness of the trajectory, since minor inaccuracies in the optical tracking are corrected.

The smoothing effect of the sensor fusion is even more evident in Figure 5.4. In contrast to Figure 5.3, the new version of the optical tracking also exhibits larger deviations and incorrect values in this case. Furthermore, it is observed that both versions encounter challenges in accurately tracking the correct position of the underwater VR headset. This difficulty is likely attributable to the restriction of visual contact with the fiducial tags. It is evident that the sensor fusion, thus being influenced by outliers, is leading to a dampening of these deviations. This results in a smoother trajectory, while at the same time maintaining continuity. Good

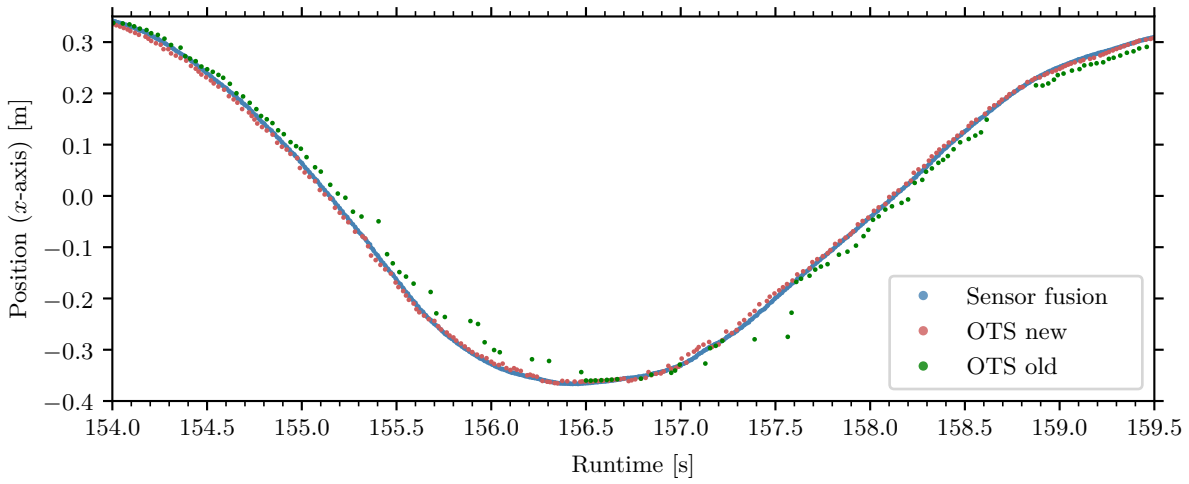

Figure 5.3: The headset's position in the x-axis of the various tracking systems is visualized over a 5.5-second period. The sensor fusion with Kalman filter is compared to the old and new version of the optical tracking system.

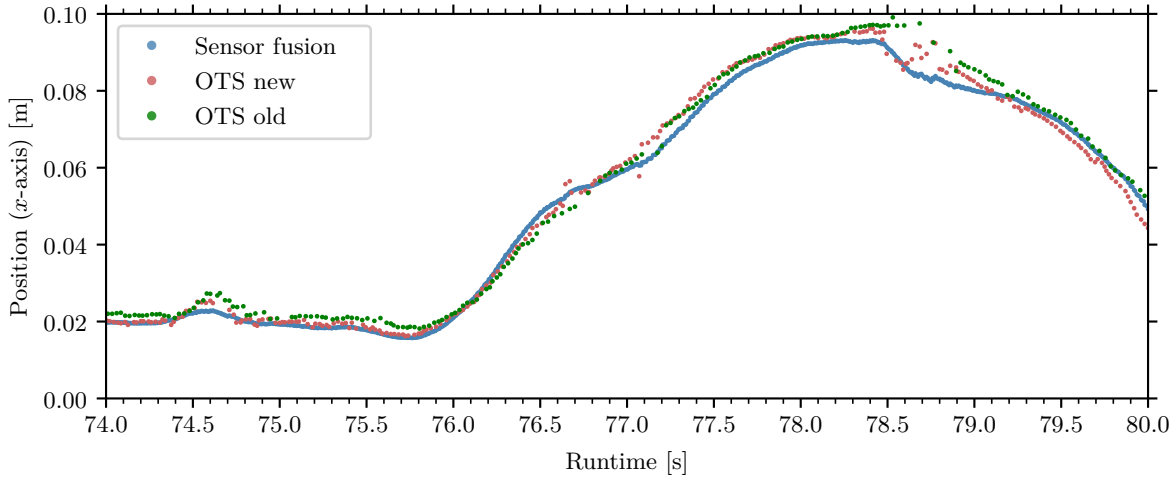

Figure 5.4: The headset's position in the x-axis of the various tracking systems is visualized over a six-second period. The sensor fusion with Kalman filter is compared to unstable outputs characterized by jumps from the old and new version of the optical tracking system.

examples of these effects are observable around the time marks of 74.5 and 77.0 seconds. An extreme case of these deviations and incorrect values is also observed around 78.5 seconds in Figure 5.4. It is evident that the sensor fusion was influenced by the abrupt decline in values from the optical tracking system, resulting in a decrease as well. This decrease is most likely incorrect, as new values continue to be detected at the actual previous position of the trajectory in between. However, instead of leading to a jumping behavior, the calculated position of the

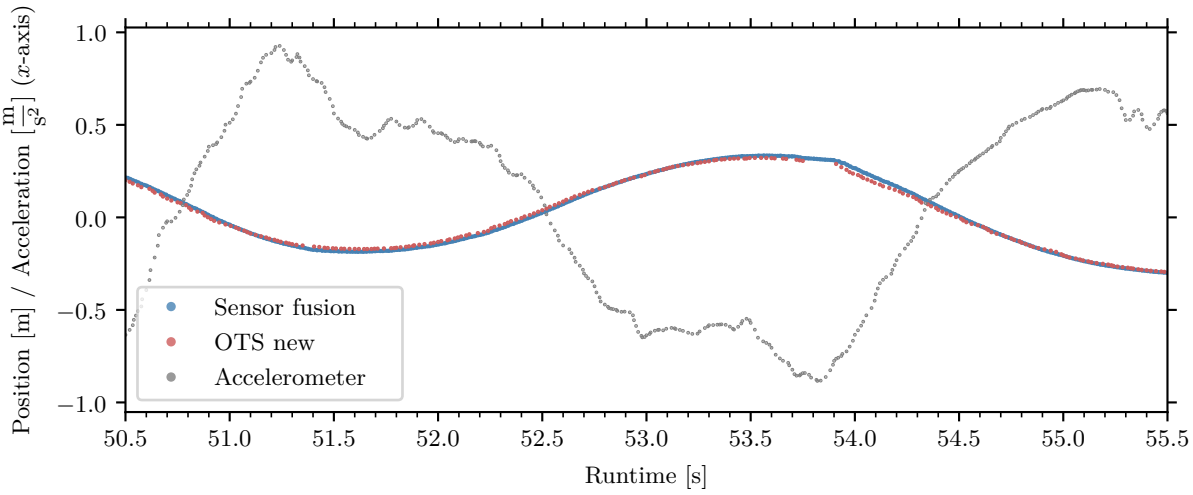

Figure 5.5: The headset's acceleration output from the IMU in the x-axis is visualized over a five-second period together with the position tracking of sensor fusion and OTS new.

sensor fusion flattens out and approaches the true trajectory again.

In addition to the new version of the OTS and the sensor fusion, Figure 5.5 also illustrates the accelerometer data, which is taken into account in the Kalman filter when estimating the state. The accelerometer data is utilized after applying the low-pass filter, which is observed by the continuous data points and reduced high frequencies. As demonstrated in previous graphs, in the absence of new data from optical tracking, the sensor fusion process persists in calculating the position based on the estimate of the prediction step in the Kalman filter and the accelerometer data. This estimation is also apparent at 53.8 seconds, as no new OTS data is currently available for about 100 milliseconds. If only relying on the optical system, the tracking would have stopped at this point, whereas the Kalman filter sensor fusion effectively closed this gap in the data.

In order to estimate the time differences from the positional tracking as well, Figure 5.6 displays two examples to estimate the delay between the systems. Given that the Kalman filter for sensor fusion utilizes new version of the OTS as the system input, the computed position is significantly influenced by the new OTS data. Consequently, there is negligible time difference between the two systems. For measuring the overall time improvements for the position tracking, a more effective approach is to compare the new OTS version with the previous iteration. Example 1 shows a small peak in both the OTS data, from which the time difference is determined. The smoothing ability of the sensor fusion is also emphasized here again, as there is no deviation in the filtered data. Comparing both peaks marked by a red circle in Example 1, the difference yields $7.866 - 7.800 = 0.066$ seconds. In Example 2, the time difference between the two marked data points is $54.841 - 54.763 = 0.078$ seconds. However, as the optical tracking system data is affected by measurement inaccuracies (especially OTS old in Example 2), this calculated time difference should only be used as a further estimate of the delay for comparison purposes.

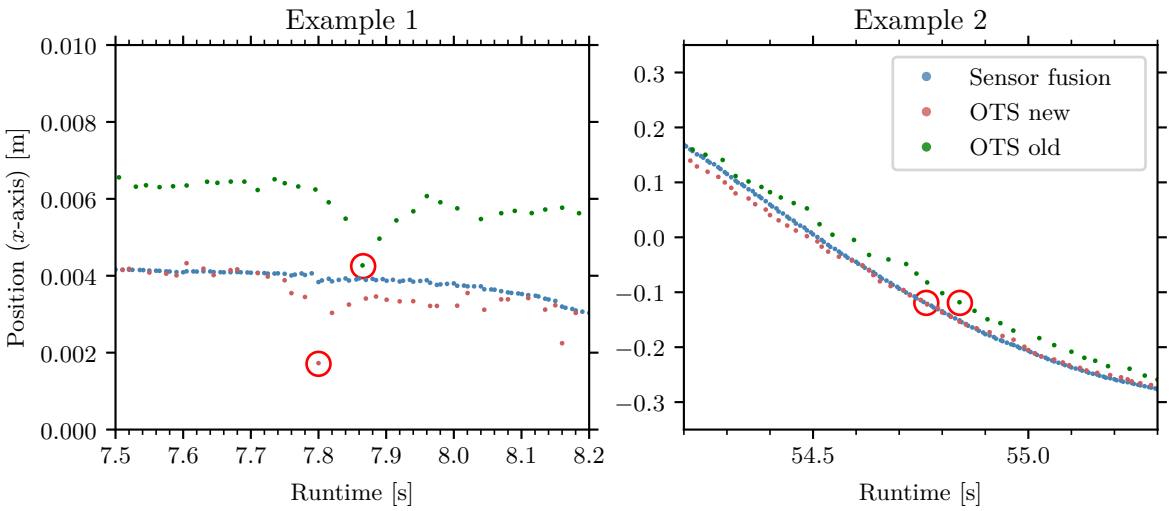

Figure 5.6: Illustration of two examples of x-axis position tracking to determine the time differences between OTS new and OTS old. The data points marked with a red circle, which represent matching features in the tracking, are compared in each case.

Rotation In the subsequent section, a comparison of the rotation data is made between the IMU and the new and old versions of the optical tracking system. As mentioned in Section 4.3.7, the rotation vector returned by the IMU is directly used for the rotation tracking, after a transformation to the OTS coordinate frame in the calibration step. Therefore, the rotation tracking in the sensor fusion algorithm is entirely distinct from the optical tracking system. This is illustrated by the following examples. To ensure consistency and clarity, the subsequent graphics are constrained to the quaternions y-parameter of the headset's orientation. The y-parameter was selected because a good continuous trajectory of the data points was recognizable here. In contrast, the x-parameter led to the data mostly clustered at points -1 and 1 (based on the previous rotation of 180 degrees), which resulted in graphs less suitable for continuous visualization of a trajectory.

As illustrated in Figure 5.7, the rotation tracking y-parameter of the aforementioned tracking methods is plotted in a trajectory over a span of 10.5 seconds. Similar to the prior position comparisons, the data from the optical tracking system (version old and new) contains outliers and erroneous values. In contrast, the rotation tracking of the IMU provides continuous smooth values. Furthermore, the higher data rate of the IMU values is noticeable, which leads to more accurate tracking with less delay and lag between the individual data points. A deviation of the individual trajectories in both axes is also apparent. The time difference is further elaborated upon in Figure 5.8. The disparity in the y-parameter between the old and new version of the OTS can be attributed to the recalibration of the cameras as part of the optical tracking improvements. Nevertheless, a discrepancy persists between the IMU rotation and the new OTS data as well, despite the transformation of the IMU rotation vector to the OTS orientation during the preceding calibration step.

In Figure 5.8, the aforementioned time difference between the system is evaluated from the

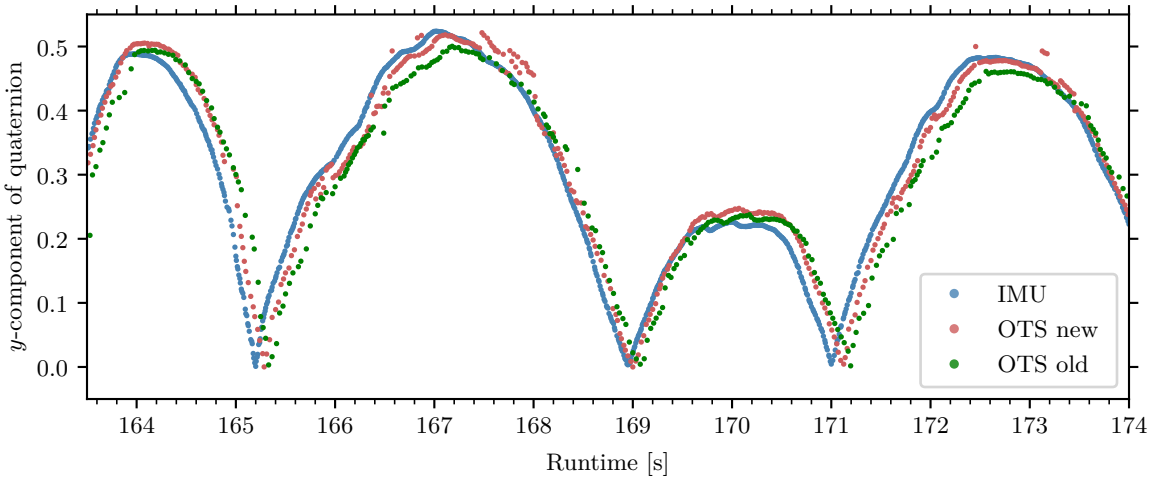

Figure 5.7: The y-component of the headset's rotation quaternion is visualized over a 5.5-second period. The sensor fusion with Kalman filter is compared to the old and new version of the optical tracking system.

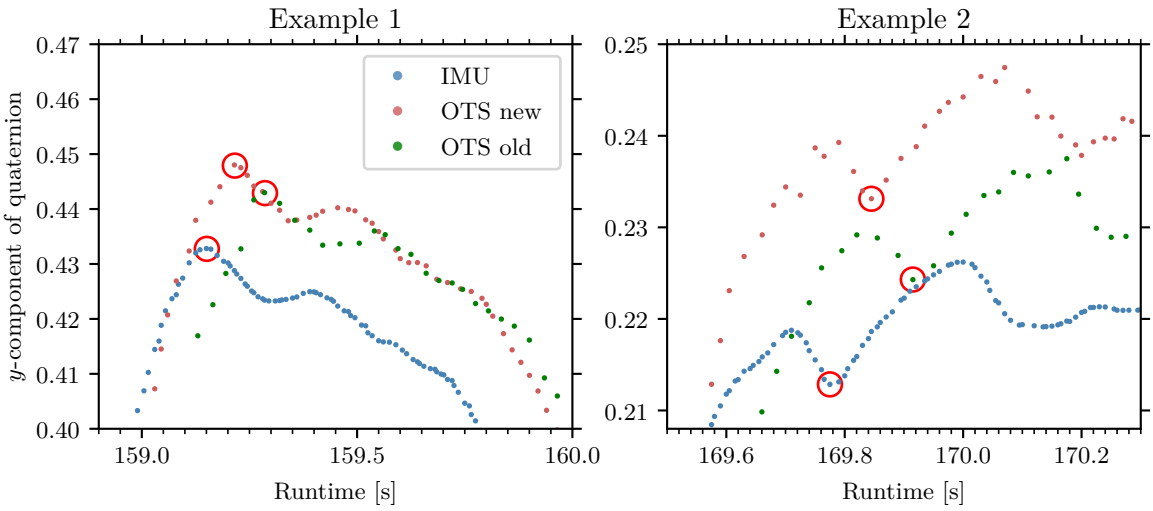

Figure 5.8: Illustration of two examples of the y-component of the rotation quaternion to determine the time differences between IMU, OTS new and OTS old. The data points marked with a red circle, which represent matching features in the tracking, are compared in each case.

single trajectories. The red circles denote the points that are being compared. The corresponding times are displayed in Table 5.3. For both examples illustrated in Figure 5.8, recurring points were selected among all tracking methods to ensure optimal comparability between them. It should be noted that a direct comparison of the time-based data, using the same y-parameter, is not feasible due to the previously mentioned shift in both axes. As illustrated in Example 1 of Table 5.3, the output from the IMU is 65 milliseconds faster than that of the new OTS and 135

	IMU	OTS new	OTS old
Example 1	159.151 s	159.216 s	159.286 s
Example 2	169.775 s	169.845 s	169.915 s

Table 5.3: Timestamps of the points marked in Figure 5.8.

Time difference between tracking methods in ms				
Tracking methods	Position		Rotation	
	Example 1	Example 2	Example 1	Example 2
OTS old - OTS new	66	78	70	70
OTS new - IMU	-	-	65	70
OTS old - IMU	-	-	135	140

Table 5.4: Summary of the overall latency/delay improvements of different tracking solutions.

milliseconds faster than that of the old OTS. In Example 2, the IMU is 70 ms faster than the new OTS and 140 milliseconds faster than the old OTS. Therefore, both examples align with a discrepancy of 70 milliseconds between the new OTS and the previous version of the optical tracking system.

Table 5.4 summarizes the time differences measured thus far for the different tracking solutions utilized in the robustness tests of translation and rotation. It is evident that the time differences between OTS old and OTS new for the translation and rotation correspond to each other with a mean difference of 71 ms. Time differences between sensor fusion position and OTS were not recorded, as there are no significant time differences between sensor fusion and OTS new, as already mentioned in the evaluation of the graph in Figure 5.6.

As noted at the beginning of the evaluation of Figure 5.3, the various tracking solutions have led to a significant increase in the frequency at which new data are published. At low frequencies, larger gaps become more noticeable, which can be identified through increased latency or disruptions in the tracking. In order to compare the different tracking systems, the time differences between the individual data points and each previous data point are summarized in Table 5.5. Particularly notable are the high standard deviations observed in both OTS

Time difference to last data point in ms			
Tracking method	Mean μ	Stdev σ	Max
OTS old (pos/rot)	32.16	16.42	260.08
OTS new (pos/rot)	21.65	8.32	160.04
IMU (rot)	9.37	3.68	31.57
Sensor Fusion (pos)	8.00	2.46	13.20

Table 5.5: Summary of the time differences between the single data points.

tracking versions, along with the significant maximum time period during which no new data was available. Since only comparable locations with current available data for all tracking systems have been chosen to obtain the true system dependent latency in Table 5.4, these delays between data points must be regarded to as an additional measurement error.

5.2 Field-testing at ESA's Neutral Buoyancy Facility

As mentioned in the introduction, this project originated from a collaboration with the European Space Agency. This collaboration and the further enhancements detailed in this thesis consequently provided the opportunity to visit the European Astronaut Center in Cologne and test the underwater VR headset within the pool of the Neutral Buoyancy Facility. This visit resulted in an occasion to have the VR headset assessed by a group of test persons, including the ESA astronauts Matthias Maurer and Marco Sieber, along with VR experts in the domain of astronaut training purposes and an astronaut training instructor (referred to as test subjects in the following test evaluation).

The cameras and cubic structure are mounted on a height-adjustable platform in the NBF, enabling a system testing in the dry prior to deployment. The system is then lowered exactly to the desired water depth of 1.5 meters, at which depth the tests are conducted, as seen in Figure 5.9.

5.2.1 Test procedure description

To ensure a consistent experience for all test subjects, a sequence of movements and tasks was defined in advance. This sequence is discussed with the participants during a pre-test briefing and also communicated via the headset during the dive.

The initial task of the test dive is to grip the handrail as a reference point and to become familiar with the virtual environment. Subsequently, the calibration between the IMU and the OTS needs to be performed, as outlined in Section 4.3.7. Therefore, under the guidance of the virtual horizon displayed in the headset, the headset must be centered, as previously visualized in Figure 4.8. The subsequent step in this process is to take a closer look around and to become more familiar with the surroundings. One such example is the observation of the moon. The goal is to identify any differences between the tracking robustness before and after the calibration step. Given that the tracking is based purely on the optical system prior to calibration, comparisons are made with the performance improvements of the sensor fusion. Subsequently, movements such as translation in the x, y, z direction and rotations in roll, pitch, and yaw are carried out separately in order to gain further impressions of the tracking. The next task requires moving beyond the left and right edges of the cubic platform to be able to observe the lateral surfaces. The ESA logo is randomly distributed on one of both sides for each participant. The task at hand is to locate the logo and report its position via the headset communication to the diving instructor. Subsequently, the subjects are instructed to orient their heads above the four colored markers positioned at the corners of the platform, while continuing to hold onto the handrail. The sequence of the markers is communicated via the headset by the diving instructor. Finally, each participant is given a period of time for unrestrained movement

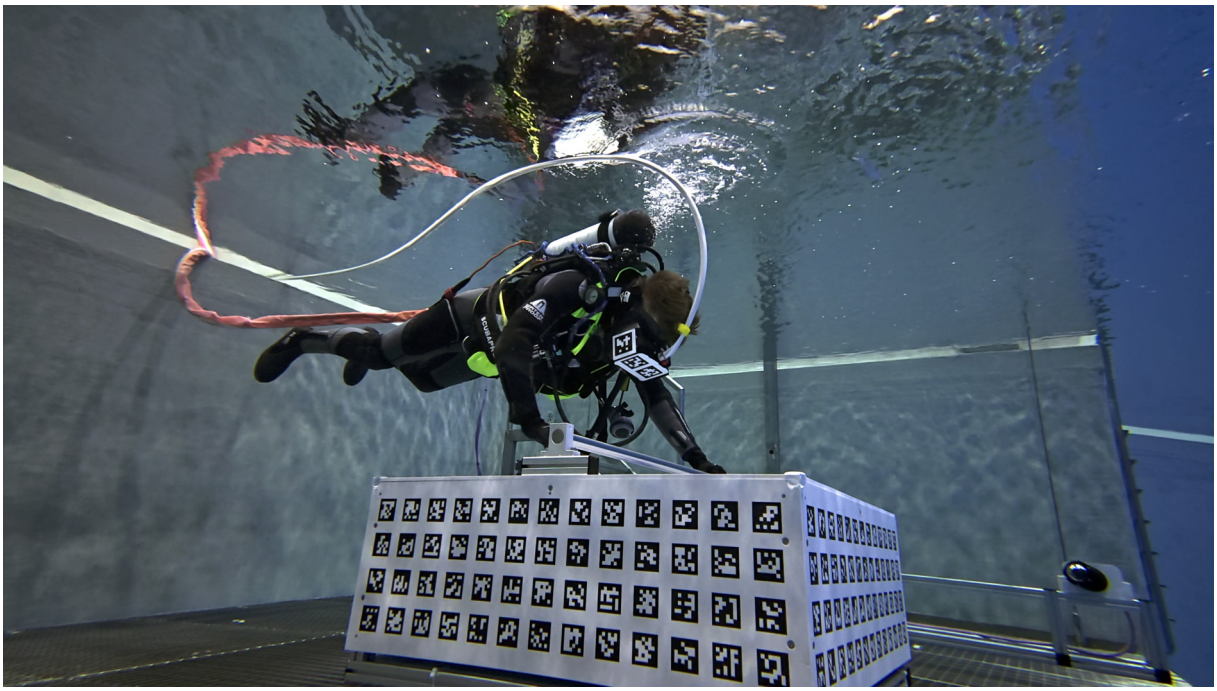

Figure 5.9: Testing the underwater VR headset at the Neutral Buoyancy Facility at the EAC.

within reach of the cubic platform, thereby further experiencing the simulation of floating in the virtual environment in zero gravity.

After carrying out the procedure, the participants are interviewed and asked to share their impressions and experiences during the test of the underwater VR headset.

5.2.2 Test results at the NBF

The following section presents a summary of the group's impressions and evaluation of the test. For more clarity, these have been divided into subsections representing different topics. The first subsections in particular relate to the tracking experience of using the underwater VR headset.

Calibration and tracking

As this was one of the initial tasks assigned to the test subjects, the first topic to be evaluated is the calibration and the differences in tracking compared to the pre-calibration state.

All test subjects agreed that the calibration process for the tracking was uncomplicated and quickly completed due to the explanation in the briefing beforehand. The primary differences after the successful calibration that were observed included a significantly higher frame rate and fewer jumps in the pose determination. However, the participants still occasionally noticed smaller jumps in the pose after the calibration. It was once mentioned that these were noticeable when objects were viewed in proximity. As soon as objects were viewed from a distance, no jumps have been observed. One of the test subjects also mentioned experiencing jumps in the world

when having the arms close to the head, which led to an occlusion of the AprilTags on the headset. Overall, all participants agreed that the tracking got improved after the calibration step.

Perception of delay

The delay is a critical factor in the overall tracking of a VR headset. As one of the main points of this work was to improve the head tracking, the test subjects were explicitly asked about their impressions of the tracking delay.

The participants' perceptions of the delay varied. Three out of the five individuals did not observe any disturbing delays, particularly with the slower movements of the head due to the inertia of the water. Two test subjects reported a noticeable delay when focusing on objects in the near environment, such as the cubic structure or handrail. One participant noted that this delay was particularly present during translational head movements. However, both subjects agreed that there was minimal to no perceptible delay when focusing on objects in the far environment.

One person of the group had already tested the old version of OTS tracking before as part of the collaboration between ESA and the University of Würzburg and therefore could compare it to the current version. It was mentioned, that the delay has improved significantly compared to the old version, which had a perceived delay of hundreds of milliseconds.

Navigation and tasks

This section will cover the impressions about the general navigation and orientation underwater in the virtual environment, as well as completing the aforementioned tasks.

The general feedback regarding the orientation and navigation within the virtual environment was that there were no problems overall. Therefore, it was not difficult to perform rotation and translation movements. However, the majority of the test subjects agreed that a change in the design of the handrail would significantly improve the overall navigation underwater. Some of the ideas included the addition of a second handrail to increase the mobility for simple movements, as well as adding vertical end pieces to the handrails.

The tasks performed underwater have been reported as relatively straightforward and not challenging, and are more comparable to system checks than to actual tasks. From the perspective of astronaut training, it has been noted that the tasks were not comparable to those performed during astronaut training. This is due to the fact that operating different types of tools plays a significant role in Extra Vehicular Activity (EVA) procedures. It was also noted that communication through the headset played an important role during the tasks, as otherwise visual contact with the outside was not given.

Motion sickness

Motion sickness must be not an issue for the headset to be used for training, which extends over longer periods of time. For this reason, the test persons were explicitly asked how they felt when wearing the headset.

During the tests, none of the test subjects experienced signs of motion sickness. Two of the subjects had no signs of any discomfort. Otherwise, it was reported that it felt unfamiliar at first not being able to see the own hands. It was also mentioned on one occasion that the previously stated problems with focusing on close objects felt slightly uncomfortable, which could possibly become a problem for longer testing periods. Finally, it was noted that the higher weight of the mask did result in a slight discomfort.

Immersion

Since the headset is mounted onto the diving mask, there is a greater distance between the lenses and the display, resulting in a smaller field of view. Therefore, this section will summarize how this affected the immersion of the virtual environment and how the simulation of zero gravity was experienced.

Overall, all participants reported a full immersion into the virtual environment. The visible edge area of the diving mask was mentioned briefly, but it was ignored after some time as the test subjects became more attracted to the virtual world. It was also noted that it was challenging to find the right focus initially. The buoyancy of the mask caused a slight shift, resulting in a change in focus compared to the first wearing of the headset outside the water. Once the correct focus underwater was determined, the field of view was described as being both adequate and not limiting.

With regard to the simulation of a zero gravity environment, it was noted that both the inertia of the headset in the water and the overall water resistance were experienced. Otherwise, the floating experience was reported to be good and realistic, provided that the diving suit and jacket were correctly buoyant.

Given Matthias Maurer's prior experience with zero gravity during his stay on the International Space Station, it was particularly fascinating to ask for a comparison between true weightlessness and of the underwater VR simulation. Regarding his experience, the effect of underwater VR is quite similar. However, he reported that in actual weightlessness, one is never stable and constantly floating around. In contrast, in the underwater VR simulation, it was easy to maintain a stable floating position with hand movements due to the water environment.

Conclusion and improvement suggestions

Finally, the participants were asked to share their impressions of underwater VR for astronaut training and any ideas for improvements.

Overall, all participants agreed that the underwater VR system would be an important addition to existing astronaut training. However, it was also noted that the system in its current state is not a substitute for a VR training session in the dry or a training dive underwater. Moreover, this would be a good addition to the existing training. It would provide an immersive experience of a zero-gravity environment for initial familiarization. After becoming used to the weightlessness experienced in the first underwater training sessions, following VR training sessions can be performed in a dry environment.

However, a few improvements should be considered, since the system is not mature enough in some aspects, as mentioned by a few participants. One frequently suggested improvement

would be the addition of hand tracking and optionally more body parts, as this would increase the general navigation underwater. In addition to integrating various trackable tools used for EVA's, this would result in particular use cases for astronaut training. However, this requires that the perception of the objects must match exactly, which is also related to the field of view of the glasses. It was also recommended, to increase the field of view in future versions to mitigate potential negative training outcomes due to inaccurate representations. Furthermore, an improvement in the mobility and comfort of the headset was also mentioned.

Finally, an enlargement of the tracking area was suggested to allow larger translational movements. It was also proposed that the arrangement of the cameras or tags should be changed in order to not occlude the tags by certain arm positions, resulting in an overall enhanced robustness.

5.3 Discussion

The results of the experiments and the practical test in the NBF are compared and discussed below. In this context, a distinction is made between delay, defined as the latency of underwater VR tracking and the robustness of the tracking. Delay refers to both the latency of the system and individual time delays such as lags while the robustness of the system is evaluated primarily according to outliers and jumping of the tracking data.

5.3.1 Delay

In summary, the improvements in TCP connection, OTS and sensor fusion with IMU have resulted in a significant reduction in latency. The new version of the TCP server has led to a reduction of around 10 ms in the overall latency. The periodic peaks from the ROS-TCP-Connector, which led to sudden further delays of up to 50 ms, have also been eliminated. The lag and higher delay experienced are most likely related to the serialization of all ROS messages within the system by the ROS-TCP-Connector. However, in the particular scenario of achieving a robust tracking system and given that only a single message was required to transmit the pose to Unity, the introduction of the new TCP server is an effective solution.

A difference in latency between the IMU and OTS new tracking systems of about 54 ms was measured until the data arrived in Unity. Although, as already derived in Section 5.1.2, this improvement represents a lower bound for the actual time difference between IMU and OTS new, as both latency measurements contain unknown errors, which can only be estimated. The time difference between the various tracking systems is also evident in the robustness tests from Section 5.1.2. As summarized in Table 5.4, a time difference of about 68 ms has been extracted between OTS new and IMU from the rotation graphs. This is in the range of 54 ms, taking into account that this represents a lower bound. This difference of 14 ms also confirms the assumption that the measurement error $e_{ots} \geq 10$ ms is a lower bound and the measurement error $e_{imu} < 10$ ms an upper bound. In reality, these errors are therefore higher and lower respectively, with an approximate difference of the aforementioned 14 ms. Consequently, the subsequent measurements are estimated with a correction of the error by the mean value of 7 milliseconds.

However, it should be noted that reading from the graph is an imprecise method and only two data points per position/rotation test were compared. As the time difference between the measurement methods remains constant over the different graphs (as seen in Figure 5.7), the results provide a sufficient impression of the overall improved system performance. Looking at the difference between IMU and OTS old from this point of view, the improvement in tracking latency is approximately 138 ms according to Table 5.4. Since only the IMU data is used to determine rotation due to the initial calibration, an **overall rotation tracking latency improvement of 138 ms** is assumed. The actual latency is more difficult to estimate. With the previous assumption of a measurement error $e_{imu} < 10$ and the results from Section 5.1.2, this is assumed to be less than 14 ms due to the upper bound error e_{imu} (while estimated to be slightly lower in the range of 7 ms due to the previously determined difference between e_{ots} and e_{imu}).

The improvements for position determination are likely to be smaller than for the rotation, as these are dependent on the OTS new tracking data. This correlation is also apparent when observing the various position graphs in Section 5.1.2. This is also explained from a technical point of view, as the Kalman filter in this case is primarily intended to provide stability and robustness, but is still dependent on the OTS new data as the anchor point of the true position, and therefore results in no or only very small time improvements. Therefore, the time difference achieved by changes in the optical tracking system yields an approach to obtain a value for the time improvements in the positional tracking. Following Table 5.4, the mean time difference between OTS old and OTS new is about 71 ms. After all, an **overall position tracking latency improvement of 71 ms** is assumed. With the previous assumption of a measurement error $e_{ots} \geq 10$ and the results from Section 5.1.2, the overall position delay is therefore at least 69 ms (while estimated to be slightly higher in the range of 76 ms due to the previously determined difference between e_{ots} and e_{imu}).

Therefore, an estimated delay for the OTS old is calculated by adding the time improvements to the estimated tracking method delays. The OTS new delay, when added to the difference between the OTS old and new tracking, results in an approximate value of $71 + 69 = 140$ milliseconds as a lower bound. On the other hand, the IMU delay together with the difference between IMU and OTS old yields an estimated value of $138 + 14 = 152$ milliseconds as an upper bound. The 12 millisecond difference corresponds to the estimated 14 millisecond disparity between e_{ots} and e_{imu}. Therefore, the mean value between the two delay times is used as an estimate for the OTS old latency, resulting in 146 ms.

However, it should be noted that this is purely a technical delay in the determination of position and rotation up to the Unity application. To determine the time difference, reference points were selected only where all tracking solutions have available data points, as seen in Figure 5.8. Yet this is not the case for every reference point, especially with the lower frequencies of optical tracking methods. This can often lead to a situation where the optical tracking does not provide any new data or only provides it with a delay. Therefore, Table 5.5 summarizes the time differences between the single data points of each tracking system. These time differences until a new data point is available are then perceived as an additional delay or even lag. With OTS old in particular, this results in a significant addition to the existing latency with a mean interval of 32 ms and values up to a maximum of 260 ms between the single data points.

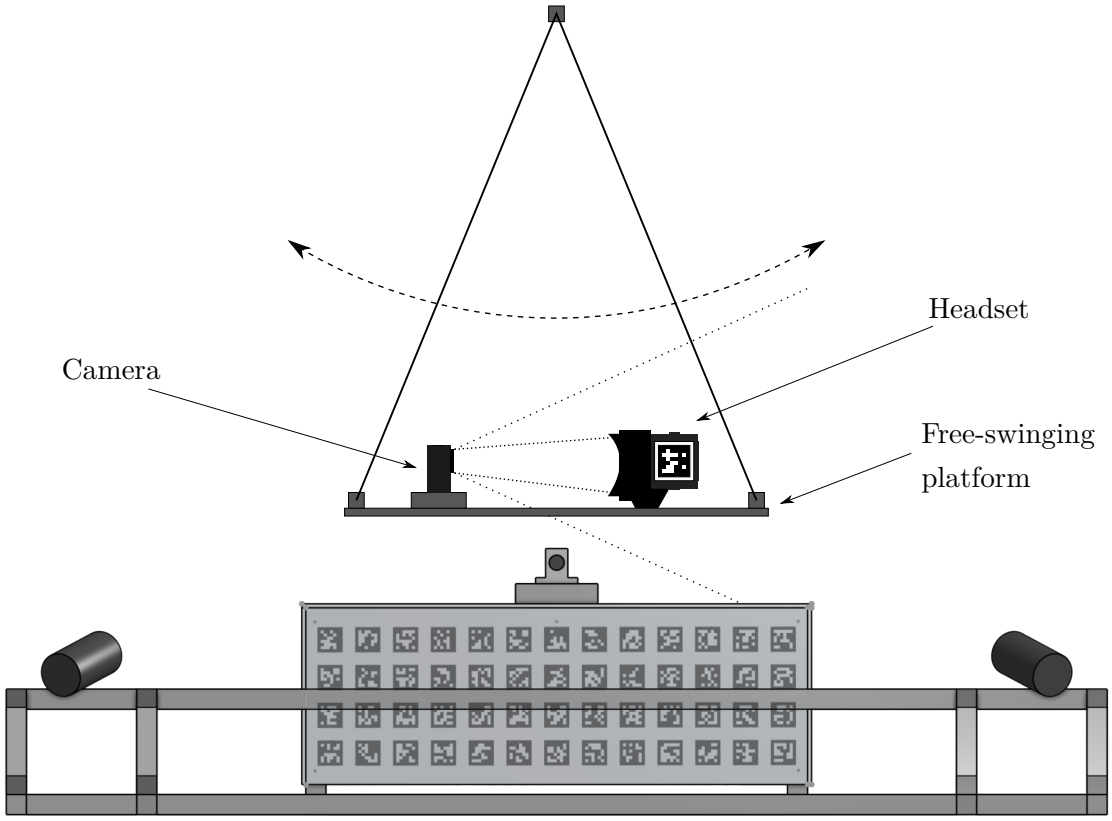

Figure 5.10: Drawing of the experimental setup to determine the end-to-end latency using a free-swinging platform. A camera and the underwater VR headset are mounted on a moving platform in order to determine the delay in the representation on the displays and in reality through subsequent frame comparison. The drawing may differ from the actual scale.

Comparison to initial visual experiments These differences become particularly clear when comparing the results obtained so far with latency measurements carried out by Helene Klein during an internship at the Institute for Computer Science XVII (Robotics) at the University of Würzburg. As part of those measurements, the end-to-end tracking delay was determined by analyzing the frames of 240 frames per second (fps) video data between an initial motion and subsequent visualization on the displays inside the headset. Those experiments are referred to as *visual experiments* in the following work. To obtain these test data, a smartphone capable of 240 fps video capture and the underwater VR headset were mounted on a free-swinging platform, as illustrated in Figure 5.10. This platform was attached to the ceiling of the room using ropes. Therefore, this attachment enabled oscillating motions, comprising both translational and rotational movements, which are observable in the video footage captured from both the displays and the surrounding environment. Following the frame-level comparison of the footage, the latency between the final visualization of the tracking pose can be determined with an accuracy of about $1/240 \approx 4$ milliseconds. Figure 5.11 shows the experimental setup of the free-swinging platform that was utilized to record the subsequent time measurements. As these are measure-

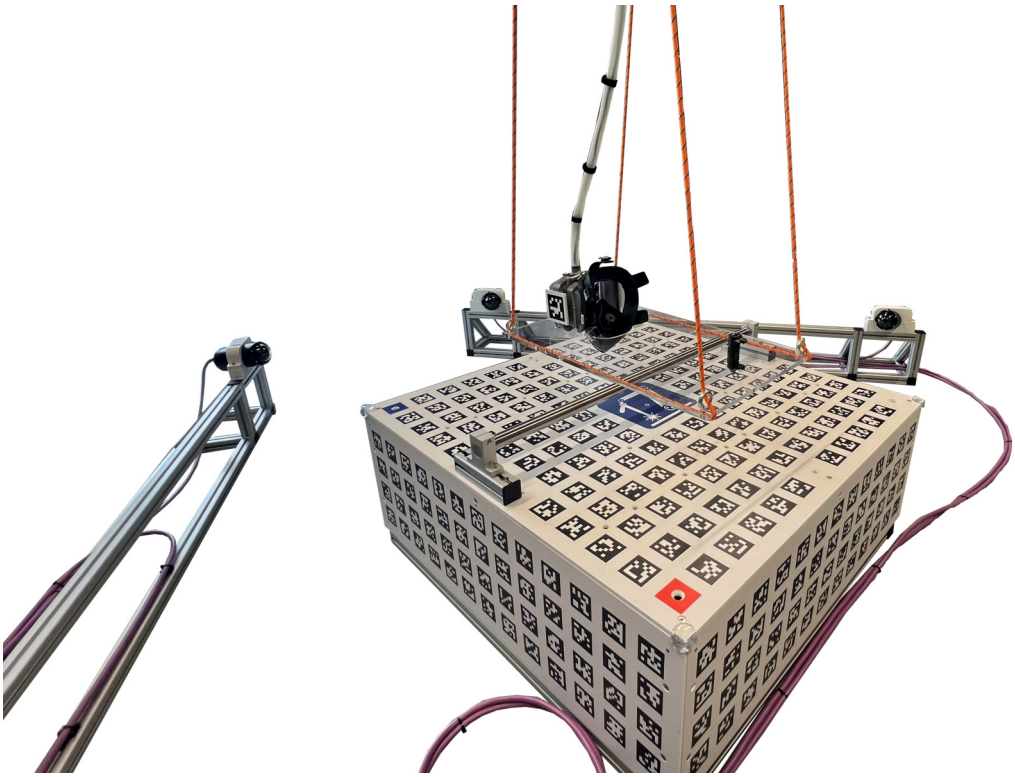

Figure 5.11: Picture of the free-swinging platform experimental setup to determine the end-to-end latency. A camera mount for a smartphone capable of 240 fps recording and the underwater VR headset are fixed on a moving platform in order to determine the visual delay. The camera bars were positioned at an angle to mitigate the obstruction caused by the ropes mounted to the ceiling.

ments from an early version, they can only be used to compare the initial delay. These were measured with the initial version of optical tracking and for the most time with the old version of the ROS-TCP server. However, they are a great example for illustrating the differences to the technical tracking delay due to influencing factors such as the low data rate, which the user experiences as well, but are difficult to measure otherwise. As an example, the latency measured for OTS old with the ROS-TCP-Connector was 268.0 ms, which is made up of measurements of the oscillation in translation 350.9 ms and rotation 161.3 ms. Both results are the average of several individual readings from the visual experiments, with high deviations in the single values. The significant differences here also highlight the high variance of the measurements and tracking, as the rotation and translation of OTS old were determined for this software version and should therefore be similar. The experiments exceed the estimated technical delay for the OTS old of 146 ms. These differences are probably resulting from the different data rates of the tracking systems (Table 5.5) as well as possible error sources in measurements and evaluation. This conclusion is further supported by additional results from the tests conducted by H. Klein. In explanation, an earlier version of the Kalman filter sensor fusion with OTS old as the positional input was also tested for latency. The results indicate a significant reduction in the variance of the measured times. Comparing this to the time differences between individual

data points of the sensor fusion in Table 5.5, the significantly lower standard deviation is also noticeable here.

This leads to the following summary: The stability in the time differences of the data rates has significantly improved the perceived lags and the overall perceived delay experienced by the user. These factors were not taken into account when calculating the true technical system latency. Another potential issue is the missing time delay between rendering and display in the technical latency estimation, though this should not result in a significant overhead. The initial delay experienced by the user with the original tracking version was likely substantially higher than the estimated 146 milliseconds. It was more likely in the range of approximately 268 milliseconds, as observed in the visual experiments.

This high delay is also confirmed by the results of the on-field tests in ESA's Neutral Buoyancy Facility at the EAC. One of the participants (VR expert in astronaut training) had already been able to test the system in its original version due to the collaboration with the university. The statements made here were that there was an initial delay of hundreds of milliseconds. Compared to that, the delay improved significantly in its final tracking version. The other participants also confirmed that there was no or only minimal delay perceived. It was noted that slight delays were only experienced during translations. This finding aligns with the previous results, which indicated that while the delay in rotations is minimal at approximately 7 milliseconds, while the delay of the OTS can be noticeable at approximately 76 milliseconds in certain scenarios.

5.3.2 Robustness

Large outliers and sudden deviations have been identified as contributing factors to motion sickness and a suboptimal user experience, leading to the technology no being suitable for use in astronaut training [8, 24]. Therefore, the following discussion will focus on the overall improvements in the robustness of the tracking data with regard to outliers and jumps. These improvements are based on the results of the experiments and evaluations.

For the positional tracking, the new version of the optical tracking system already improved the tracking robustness, as evidenced by a substantial reduction in outliers and jumps (Figure 5.3). Nevertheless, as seen in Figure 5.4, the system is still prone to errors like inaccuracies and outliers, probably due to occlusion of the cameras or false detections of the AprilTags. However, it was possible to further enhance the tracking robustness against these errors by employing a Kalman filter as a sensor fusion with additional IMU accelerometer data (Figure 5.5). A clear and immediate consequence of the sensor fusion is the smoothing of the data. This results in slight deviations being flattened out as seen in Figure 5.3. As illustrated in Figure 5.4, even substantial jumps and errors are mitigated. However, with larger errors, the computed trajectory by the sensor fusion gets influence by the inaccuracies. Moreover, minor errors, deviations, or inaccuracies are corrected (Figure 5.6). Additionally, minor gaps and stops are successfully closed or estimated, thereby preventing any occurrences of lag or jump (see Figure 5.5). This is a consequence of the prediction in the sensor fusion, which continues to estimate the subsequent state if no new input is available, which results in a substantial increase in the data rate. Overall, the integration of sensor fusion enhances the overall convenience and ease of use of the system.

Regarding the rotation tracking, it should be noted that this relies exclusively on the IMU's

rotational output after the calibration step. Consequently, its robustness has been significantly enhanced due to becoming independent of the OTS new data. Therefore, no visible inaccuracies or outliers occur anymore, as well as gaps or stops in tracking, as seen in Figure 5.7. However, it is important to note that the IMU data shows a constant shift in the data compared to the OTS outputs. The minor shift persists due to not accounting for the time offset between the IMU and the OTS tracking during the calibration step. Consequently, the two states of orientation are not precisely aligned after calibration because the faster IMU was calibrated to the rotation of the OTS that occurred slightly earlier. Therefore, this shift in time results in the observed offset. However, this offset is not perceptible in the practical use, as it was only identified during the evaluation of the graphs. Finally, as previously stated, the increased frequency of the system tracking data enhances its robustness as well, leading to a reduction in the frequency of jumps.

The results from the practical tests in the EAC are consistent with the results of the experiments. After successful calibration, the increase in the frame rate and the reduction in jumps and incorrect values were the most mentioned improvements by the participants. However, it was observed that sometimes nearby objects jumped slightly, while objects in the far field remained stable. The shifts in the tracking primarily occurred only in translational directions. These phenomena are due to the position tracking, which calculates the translations of the user relative to the world, resulting in a perceived shift in the virtual environment. This can also be validated by the statement that distant objects have not jumped, as rotation is a more significant factor for objects at a distance. In summary, the consensus from the tests is that navigation in translation and rotation was very straightforward.

Chapter 6

Conclusion and Future Work

As part of this work, it was shown that the sensation of weightlessness in space can be achieved by combining a VR headset within an underwater neutral buoyancy environment. The combination of inertial and optical sensors has been demonstrated as a viable approach for six-degree-of-freedom tracking underwater. A latency of 146 milliseconds was identified as the pure technical delay of the initial version of the study. However, substantial irregularities in the data rates resulted in an actual perceived delay of 268 ms on average. The Kalman filter sensor fusion effectively eliminated these irregularities and significantly increased the data rate. The integration of an additional inertial sensor led to a substantial reduction in the delay, with a resulting latency of approximately 7 ms for rotations and approximately 76 ms for translations. Additionally, the sensor fusion visibly enhanced the smoothness of the overall data, leading to a reduction of overall noise as well as outliers and jumps in the underlying optical tracking system data. Consequently, the changes presented have significantly improved the tracking capabilities compared to the initial version of the study. In the final testing phase at the European Astronaut Centre, the opinions and impressions of experts in this field have been surveyed.

In the problem specification of this thesis in Section 4.1, requirements regarding the sensor fusion for pose tracking of the headset have been defined based on [25]. One of them is that the sensor fusion must work autonomously even if visual contact to the fiducial markers is lost for a short period of time. This is fulfilled for short periods, as shown in the previous experiments and the discussion in Section 5.3. Another requirement was that position and orientation estimation should be free of significant delay, as this can lead to symptoms like motion sickness. This has been achieved, however with temporal differences in the delay between position and rotation tracking, with the latter being significantly lower due to relying only on the inertial measurement unit sensor data. L. Terenzi and P. Zaal demonstrated in [24] that the thresholds for rotations were considerably lower to experience symptoms of motion sickness. This observation was also reflected in the results obtained by the participants, who showed no indications of discomfort or motion sickness. Another specification was defined for the sensor fusion with the objective of reducing overall noise and lag. This was achieved by reducing jumps and overall noise and significantly increasing the overall data rate. A final requirement was that the resulting sensor fusion tracking data must consistently outperform the underlying optical tracking system, or at the very least, match its performance in terms of robustness. As outlined in Section 5.3, the

results of the experiments demonstrate that the requirements have been fulfilled, particularly with regard to overall delay, stability and consistency.

Based on the observations and the participant feedback, underwater VR can be integrated into existing training courses for astronauts. For example, the number of dives in large facilities such as NASA's NBL or ESA's NBF may be reduced and conducted more frequently in VR. Additionally, after initial familiarization with a zero gravity environment, further tests could be carried out in dry conditions. However, to ensure the system's full maturity, the following additions and improvements are recommended.

In future research, the suggestions from the participants at the field-test at the NBF as well as experiences gained through this work can be investigated. One of the most notable enhancements is the shift from outside-in tracking to inside-out tracking, where the cameras are positioned inside the headset. In this scenario, fiducial tags could be distributed in the environment, which would then be detected by the headset, resulting in the determination of the headset's pose. Concurrently, body parts such as the hands can also be detected if they are within the field of vision, which has been a frequently requested improvement by the test participants. This method would greatly enhance the headset's ease of use and improve its portability, eliminating the need for the large camera bars. However, there are still challenges to be addressed regarding space in the headset for the integration of cameras, displays, inertial sensors and a processing unit together with a power source, such as batteries. Given that astronaut training sessions can last for over six hours, a wired solution for an external battery might be a better choice. This could result in a hybrid solution, like still utilizing a wired solution as in this project, but with inside out tracking. This implementation would facilitate the processing of the data on a powerful external workstation, thereby eliminating the requirement for a power source within the headset. As recommended by the participants at the EAC as well, implementing inside-out tracking would allow for a larger training area without the need for additional camera systems. This would only require the placement of more markers in the area. Another proposed improvement from the EAC participants was the integration and tracking of training tools needed for EVA-specific tasks. However, improvements are needed regarding the jumping and the latency of the tracking data to ensure stability within a range of millimeters when utilizing tools to maintain the training effect.

Bibliography

[1] A. Becker. *Kalman Filter from the Ground Up.* Alex Becker, Campbridge, Massachusetts 02142, 2023.

[2] E. Berger. Why is NASA renting out its huge astronaut pool? To keep the lights turned on. `https://arstechnica.com/science/2017/02/as-it-seeks-to-pare-costs-nasa-opens-its-historic-facilities-to-private-companies/`, February 2017. [Last accessed Feb. 10, 2025].

[3] Inc. CEVA Technologies. SH-2-Reference-Manual v1.9. `https://www.ceva-ip.com/wp-content/uploads/2019/10/SH-2-Reference-Manual.pdf`, 2021. [Last accessed Feb. 10, 2025].

[4] Inc. CEVA Technologies. BNO080_085-Datasheet v1.17. `https://www.ceva-ip.com/wp-content/uploads/2019/10/BNO080_085-Datasheet.pdf`, 2023. [Last accessed Feb. 10, 2025].

[5] S. Ennis, F. Rometsch, F. Saling, B. Fischer, P. Mittler, L. Ferra, C. Vizzi, A. Casini, M. Marnat, C. Thevenot, and L. Boyer. Astronaut training on-board the International Space Station using a standalone Virtual Reality headset. In *72nd International Astronautical Congress (IAC 2021)*, IAC '21, October 2021.

[6] ESA. Underwater spacewalk training with Thomas Pesquet. `https://www.esa.int/ESA_Multimedia/Videos/2021/03/Underwater_spacewalk_training_with_Thomas_Pesquet`, 2021. [Last accessed Feb. 10, 2025].

[7] ESA. Underwater VR for astronaut training. `https://www.esa.int/ESA_Multimedia/Images/2022/06/Underwater_VR_for_astronaut_training`, 2022. [Last accessed Feb. 10, 2025].

[8] C. Sinnott et al. Underwater virtual reality system for neutral buoyancy training: Development and evaluation. In *25th ACM Symposium on Virtual Reality Software and Technology*, number 29 in VRST '19, November 2019. 10.1145/3359996.3364272.

[9] A. Garcia, J. Schlueter, and E. Paddock. Training Astronauts using Hardware-in-the-Loop Simulations and Virtual Reality. In *AIAA Scitech 2020 Forum*, number 167 in AIAA '20, January 2020. 10.2514/6.2020-0167.

[10] S. Gupta. Circuit Digest. Passive Low Pass Filter. https://circuitdigest.com/tutorial/passive-low-pass-filter, February 2018. [Last accessed Feb. 10, 2025].

[11] D. Hatsushika, K. Nagata, and Y. Hashimoto. Underwater vr experience system for scuba training using underwater wired hmd. In *OCEANS 2018 MTS/IEEE Charleston*, 2018. 10.1109/OCEANS.2018.8604592.

[12] Interspiro. Divator Vollgesichtsmaske. https://interspiro.de/de-de/produkte/tauchen/divator-vollgesichtsmaske. [Last accessed Feb. 10, 2025].

[13] Canonical Ltd. What is ROS. https://ubuntu.com/robotics/what-is-ros. [Last accessed Feb. 10, 2025].

[14] NASA. Neutral Buoyancy Laboratory. https://www.nasa.gov/johnson/neutral-buoyancy-laboratory/. [Last accessed Feb. 10, 2025].

[15] NASA. Station Facts. https://www.nasa.gov/international-space-station/space-station-facts-and-figures/. [Last accessed Feb. 10, 2025].

[16] NASA. The Astronaut Training Pool. https://www.nasa.gov/podcasts/curious-universe/the-astronaut-training-pool/. [Last accessed Feb. 10, 2025].

[17] T. Nilsson, F. Rometsch, L. Becker, F. Dufresne, P. Demedeiros, E. Guerra, A. Casini, A. Vock, F. Gaeremynck, and A. Cowley. Using Virtual Reality to Shape Humanity's Return to the Moon: Key Takeaways from a Design Study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, number 305 in CHI '23, page 16, April 2023. 10.1145/3544548.3580718.

[18] E. Olson. AprilTag: A robust and flexible multi-purpose fiducial system. Technical report, University of Michigan APRIL Laboratory, May 2010.

[19] PADI. Regulator - The Essentials. https://www.padi.com/gear/regulators. [Last accessed Feb. 12, 2025].

[20] Open Robotics. ROS 2 Documentation: Iron. https://docs.ros.org/en/iron/. [Last accessed Feb. 10, 2025].

[21] S. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Elsevier Science, sixth edition, 2020.

[22] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2nd edition, 2011.

[23] Unity Technologies. Unity Documentation. https://docs.unity.com/. [Last accessed Feb. 10, 2025].

[24] L. Terenzi and P. Zaal. Rotational and Translational Velocity and Acceleration Thresholds for the Onset of Cybersickness in Virtual Reality. In *AIAA Scitech 2020 Forum*, number 171 in AIAA '20, January 2020. 10.2514/6.2020-0171.

[25] A. Tobergte, M. Pomarlan, and G. Hirzinger. Robust multi sensor pose estimation for medical applications. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 492–497, 2009. 10.1109/IROS.2009.5354696.

[26] A. Trang. Introducing: Unity Robotics Visualizations Package. `https://unity.com/blog/engine-platform/introducing-unity-robotics-visualizations-package`, 2021. [Last accessed Feb. 10, 2025].

[27] J. Uri. NASA. Space Station 20th: Expedition 1 Arrives at the International Space Station! `https://www.nasa.gov/history/space-station-20th-expedition-1-arrives-at-the-international-space-station/`, November 2020. [Last accessed Feb. 10, 2025].

[28] J. Wang and E. Olson. AprilTag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, October 2016. 10.1109/IROS.2016.7759617.

[29] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[30] H. Younesy. GitHub. Unity-Robotics-Hub. `https://github.com/Unity-Technologies/Unity-Robotics-Hub`, 2022. [Last accessed Feb. 10, 2025].

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Würzburg, February 2025