SAPIENZA
UNIVERSITÀ DI ROMA

Design, development and evaluation of a framework for recording and synchronizing experiences in VR with physiological signals

Facoltà di Ingegneria dell'informazione, informatica e statistica

Corso di Laurea Magistrale in Engineering in Computer Science

Candidate

Danilo Bernardini
ID number 1544247

Thesis Advisor

Prof. Massimo Mecella

Academic Year 2017/2018

Thesis not yet defended

Design, development and evaluation of a framework for recording and synchronizing experiences in VR with physiological signals
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: danilo.bernardini93@gmail.com

Contents

Chapter 1

Introduction

Chapter 2

Project concept and technologies

This chapter introduces the project idea and the concepts behind it, starting from the use of VR combined with physiological signals and going towards use cases and project requirements. After this, involved technologies - virtual reality and physiological sensors - are illustrated in details and current state of the art is presented, including the available devices on the market. Finally, the last section is about the specific technologies used in this project.

2.1 VR and physiological signals

Virtual reality is a powerful tool not only for commercial use, but also for research. Since the beginning of its lifetime, in fact, people started to think that it could have been a good environment to test and study certain phenomenons, especially concerning human body and behavior. From the first VR basic prototypes to the advanced systems of these last years, researchers have conducted lost of experiments about people reactions and emotions with VR making use of physiological signals. The reason why virtual reality is so appropriate and useful for research is that it can simulate real situations, so it gives researchers countless opportunities: they can recreate real world environments, test new kinds of interaction, make people relive specific scenes, and so on.

As mentioned above, for many years there have been many examples of studies about VR with the use of physiological signals. An early research paper dealing with this subject was the one about fear of flying written by Wiederhold and others and published in 1998 [1]. The authors wanted to test if there are significant physiological differences between people who are afraid of flying and people who are not. In order to accomplish that, they studied the body responses of two groups of people, one suffering from a fear of flying and the other one not, while experiencing a virtual reality flight. They measured heart rate, peripheral skin temperature, respiration rate, sweat gland activity and brain wave activity. The results of the experiment were successful and they could conclude that there exist significant differences between these two groups of people. In addition, the authors also succeeded in reducing arousal of people with fear of flying making them experience the simu-

lation for several times.

The following year another research paper about VR and physiological signals was published by Cobb and others [2]. This was one of the first investigations about VR sickness (see 2.3.3): the authors examined effects and symptoms of virtual reality on people using several methodologies, including physiological parameters.

These examples are part of the several research studies conducted to analyze or treat specific phobias and diseases. More recent ones are, for example, those by Kuriakose and Lahiri [3] and by Seinfeld and others [4], published in 2015 and 2016, respectively. Both are about anxiety: the former aims at measuring anxiety level in adolescents with autism using physiological signals while being in VR, the latter examines the influence of music on anxiety induced by fear of heights in VR.

Finally, we can include in this list a paper about the hand illusion in VR with temperature monitoring by Llobera and others [5] and one about reduction of pain and anxiety in dental procedures using VR distractions [6].

2.2 Idea and requirements

There are many experiments and studies using virtual reality that also include physiological signals; the previous section presented some of them showing that VR applications are limitless and that using physiology with it can lead to important results. There is a common aspect in all these kind of studies: if the researchers want to collect physiological data during a virtual reality experience, they need to set up everything. This means to find a way to follow the experiment, to see what the participant sees in VR, to store data and to be able to analyze it. All these things are possible only if the collected data - VR video, physiological signals and other available sources - are synchronized. This process implies a lot of effort from the researchers because they have to figure out how to design and realize it, wasting time they could invest in the actual research.

This leads to the basic motivation behind this thesis project: designing and implementing a framework for recording and synchronizing experiences in VR with physiological signals. The framework would be a useful tool for whoever wants to physiologically analyze a certain phenomenon in VR. Having a ready-to-use test environment would allow to save time and focus on the actual experiment; the only required thing is to build the virtual scene, everything else is done by the framework.

2.2.1 Use cases

The project requirements come from use cases. We can imagine two main cases that cover the two different parts of the application usage:

1. examiner wants to see in real-time and to record what the participant sees and how the body behaves;

2. examiner wants to replay a previously recorded session.

Let us analyze these points one by one in order to fully understand what the system should do. After a brief description, for each use case is presented a diagram followed by the list of steps required to arrive to the goal, including the extensions.

1. Since the framework is designed for conducting VR researches with physiological signals, who conducts the experiment should be able to see what the participant sees in the virtual world in real-time and how the body reacts to the experience. In addition to this, the examiner should be able to record the whole session including all the signals and streams he has. This leads to an extension of the case, named *Record session*, which can be executed only after the examiner is able to see everything correctly. Once recorded, everything can be synchronized in order to be easily accessible and analyzable in the future. This brings the *Record session* case to be extend in turn. The use case description assumes that the participant is ready, with all the physiological sensors attached to his body and the VR headset correctly placed.

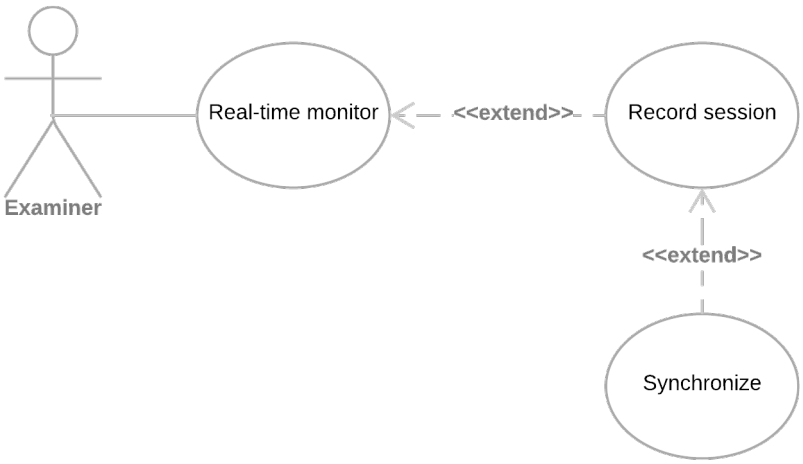

Figure 2.1. *Real-time monitor* case diagram

Name	Real-time monitor
Initiator	Examiner
Goal	Monitor the session in real-time

Main success scenario
1. Examiner runs the application
2. Examiner sets up the sensors in the application
3. Application displays sensors, VR and camera streams
4. Examiner sees everything in real-time

Extensions
4. Examiner wants to record the session
   a. Examiner starts a new recording
   b. VR, camera and sensors streams are stored
   c. Examiner stops the recording

4c. Examiner wants to synchronize the session
   a. Examiner starts the synchronization
   b. Synchronization completes

2. The examiner should have the possibility to offline replay a previously recorded session, being able to see everything as it was during the real-time monitoring. In this replay mode there should be the possibility to find key moments in which some relevant event happened. Since the session sources are synchronized, this task should be easy. The examiner should be able to tag these events placing markers on them: these would allow a simple navigation on the session and would also be very useful for analysis.

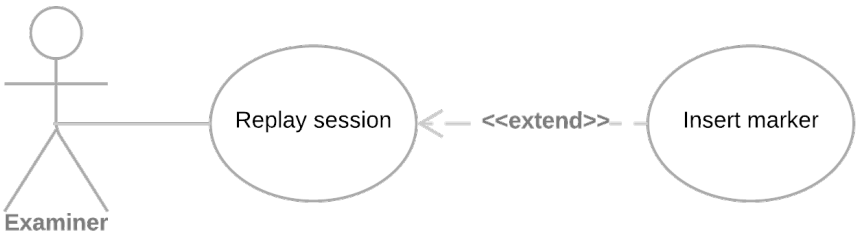

Figure 2.2. *Record session* case diagram

Name	Replay session
Initiator	Examiner
Goal	Replay a prerecorded session

Main success scenario
1. Examiner runs the application
2. Examiner selects the "replay" mode
3. Examiner opens one of the directories containing a previous recorded session
4. Examiner plays the session

Extensions
4. Examiner wants to insert one or more marker
   a. Examiner finds the moments he wants to mark
   b. Examiner inserts the marker and assigns it a label

The whole UML Use Case Diagram is represented in figure 3.3.

2.2.2 Requirements

This section is about the requirements for the project, which result from the above use cases and from some other considerations.

The system should allow to monitor a virtual reality experience while some sensors collect the participant's physiological signals. It would be reasonable to include in the system a camera that films the participant during the experience. This would make it possible to also have an external view on the user, being able to monitor his physical reactions to events. Examiners can see if the participant

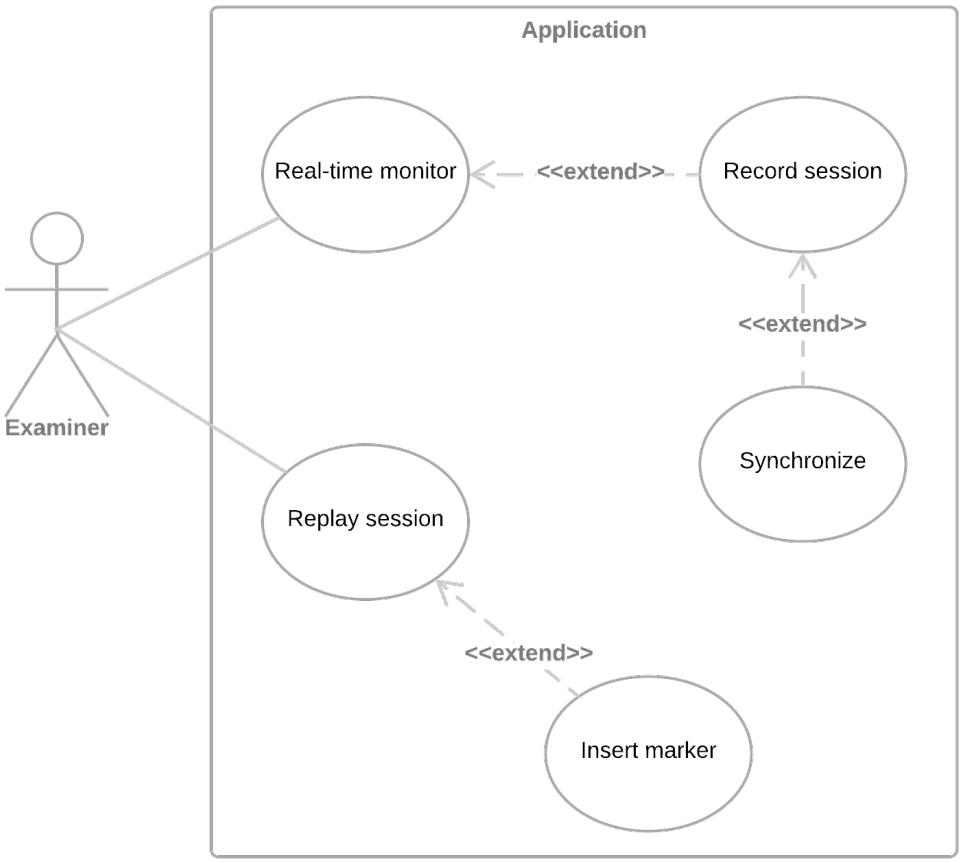

Figure 2.3. Global Use Case Diagram

moves, rotates his body, is not stable, or has any other kind of reaction in response to a virtual stimulus.

Moreover, system structure and usability should not be hard, considering that the potential users will not be engineers or computer science experts. These people, in fact, will reasonably be scientist, doctors or researchers that need to test and analyze something in VR.

The system should allow to to start a new real-time monitoring session which shows what the participant sees in VR, the camera source filming him and his physiological signals. During this "live" session, the examiner should be able to start a new recording: this will store the streams from VR, camera and sensors. At the end of the experiment the examiner should stop the recording and the various data need to be automatically synchronized. Alongside with the real-time mode, the system should have an "offline" part that allows to replay a previously recorded session. This mode should provide classical playback controls such as play, pause and stop, as well as timeline navigation to quickly skip to a specific frame. Since everything is already synchronized, in this modality the examiner can easily track every moment of the experiment; if there are relevant events it is worth to emphasize,

the system should allow to place markers on them. Finally, the stored files (VR video, camera video and sensors data) should remain independent from each other, i.e. they should still be accessible from external programs after the synchronization and markers insertion.

2.3 Virtual Reality

Virtual Reality (VR) is a technology that aims to allow a user to experience a computer-generated simulated environment. It commonly consists of a visual experience, but it can also include audio, haptic, touch and other feedback devices.

Visual devices are usually VR headsets, head-mounted displays (HMD) with two screens - one per eye - that immerse the user in a virtual world simulating the real one. While wearing the headset, the user can look around rotating his head, move in the environment or interact with virtual objects using controllers, and perceive other senses through ad-hoc devices.

The above systems are usually computer-based, i.e. they are physically connected to the computer, so the applications run here and the visual output is transmitted to the headset display. Another kind of solution is the one that uses the smartphone: in this case the headset is just a case that holds the device, the whole computation takes place in the smartphone and thus the applications are much simpler and less realistic compared to the computer-based ones (see 2.3.4).

2.3.1 VR spread

Virtual reality idea is not something new, there have been basic examples of it since the 1960s. The first head-mounted display system was the *Sword of Damocles* [7] created in 1968 by computer scientist Ivan Sutherland and his student Bob Sproull: the graphics were very primitive and consisted only of wireframe rooms, but the device was able to change the showed perspective according to the user head position. Its name comes from its appearance: the whole system was too heavy to be worn by a person so it was attached to the ceiling, suspended on the user's head.

In the next years different VR systems were developed, most of the times for the videogames industry. Examples can be found in some *SEGA* and *Nintendo* products.

The main reason why VR is exploded in the last few years is that technology is now powerful enough to support advanced computation and to give users really immersive experiences. Until a few years ago VR solutions could not satisfy *immersion* requirements (see section 2.3.3): graphics were not good enough and they had low refresh rates, causing the user some sickness (see section 2.3.3).

Conversely, in the present days we can rely on powerful machines and tools that allow us to create and experience good simulations of the world, most of the times without perceiving any issue.

VR spread is also encouraged by the release of not very expensive headsets (see section 2.3.4): a lot of people can now buy VR systems, either for development or entertainment, but also companies are starting to use them for business (see next section).

We can refer to *Gartner Hype Cycle for Emerging Technologies* of 2017 [8] to understand VR position in the market. Gartner is an American research and advisory company that provides IT-related insights and market analysis. The Hype Cycle is "a graphical depiction of a common pattern that arises with each new technology or other innovation" [9], i.e. a chart that shows the maturity and social application of a technology. The plot is divided in five phases representing the stages of a technology life cycle: Innovation Trigger, Peak of Inflated Expectations, Trough of Disillusionment, Slope of Enlightenment, Plateau of Productivity. Gartner releases it every year and in 2017 they stated that transparently immersive experiences are one of the 3 most trending topics, together with AI everywhere and digital platforms, and that VR is currently in the fourth phase of the cycle. This means that VR applications are starting to be understood and adopted by companies, that methodologies and best practices are being developed and that the technology is beginning to spread to the potential audience.

2.3.2 Applications

Virtual Reality has many applications in different fields, from gaming and entertainment to education and healthcare. Here are some examples of industries that are using VR for improving and enhancing their work.

Business: Companies can make clients experience virtual tours of business environments; they can test and show new products before releasing them or train their employees using VR.

Culture and education: VR can be used in museums and historical settings to recreate ancient sites, monuments, cities; it can be very useful for teaching, since students can be immersed in what they are studying and better understand things (e.g. history, geography, astronomy).

Games and entertainment: Gaming and entertaining industry is probably the biggest adopter of VR, since this brings the player in a whole new level of immersion and realism; VR is also used in movies, sport and arts.

Healthcare: VR allows to simulate a human body, so that students and doctors can study and train on it; it can simulate a surgery or perform a robotic one, i.e. control a robotic arm remotely; it can be used to treat phobias and diseases.

Military: VR can be used to perform a combat simulation where soldiers can train and learn battlefield tactics; it can be also useful for flight simulation or medical training on the battlefield.

Science and engineering: Virtual reality technology can help scientists to visualize and interact with complex structures, or it can be used by engineers to design, model and see something in 3D.

2.3.3 Measures

The previous sections mentioned the words *immersion*, *realism*, *sickness*, etc. These are concepts we can analyze and measure during VR experiences.

Immersion and presence

There is a lot of confusion about the concepts of immersion and presence; some people think they are the same thing, some others mix them up. According to Mel Slater, immersion concerns *what the technology delivers from an objective point of view. The more that a system delivers displays (in all sensory modalities) and tracking that preserves fidelity in relation to their equivalent real-world sensory modalities, the more that it is 'immersive'* [10]. This means that immersion is something that can be determined and somehow measured in an objective way, depending entirely on the technology and not on the user. Presence, on the contrary, is a user-dependent concept, something that varies depending on the person, a human reaction to immersion: each person can perceive a different level of presence inside the same system and even a single user can perceive a different level of presence using the same system in different times, depending on the user's emotional state, past history, etc. [11]. We can say that presence is both immersion and involvement: this regards the state of focus and attention of the user and depends on the interaction with the virtual world, the storyline, and other similar features of the experience.

Measuring immersion Since immersion is an objective concept depending only on technology, we can measure it evaluating how close the system video, audio and other features are to the real world [11]. For example if we consider visual source, we can identify some parameters that influence it:

- field of view and field of regard, respectively the size of the visual field that can be viewed instantaneously and the total size of the visual field surrounding the user;
- display size and resolution;
- stereoscopy and head-based rendering, given by head tracking;
- frame rate and refresh rate.

Measuring presence Presence is a subjective measure and is not easy to quantify. During the years some techniques have been developed and tested, the most used ones are the following [12]:

Questionnaires Users participate to the virtual experience and then answer a questionnaire about presence. The questions imply responses between two extremes (e.g. from "no presence" to "complete presence"). The problem with this approach is that asking questions about presence can affect the actual perception of the participant.

Behavioral We can see evidence of presence if users in the virtual world behave as if they were in the real world. This can be triggered by events that cause a physical reaction on the participant, such as a movement reflex or a body rotation.

Physiological signals This is a specialization of the previous approach: if we know how a person physiologically reacts to an event, and we can find the same reaction during a virtual event, then this is a sign of presence. This technique can only be used when we have well-known and easy to measure reactions, for example fear, so it is not ideal for calm and "boring" scenarios where nothing happens.

Breaks in presence (BIP) A break in presence is an event that occurs when the user becomes aware he is in a virtual environment. This can happen if the visual stimuli starts to lag, if the graphics become low-quality, if the user touches a physical object from outside the VR experience, etc. This approach allows to know presence by analyzing the moments when BIP occur, and it is a good alternative to the physiological one because it can be used in every kind of environment (calm, stressfull,scary, and so on).

Co-presence

When more than one participant share the same virtual environment we can measure co-presence (also known as shared presence). It is the feeling that the other participants are really present and that the user is interacting with real people [13]. As well as presence, also co-presence is measured with questionnaires. It is not proven that co-presence is related to presence, because some studies [14, 15] seem to find correlation between them and others [13] do not.

VR sickness

As anticipated, virtual reality can cause some sort of sickness to the user. The common symptoms are similar to motion sickness symptoms: general discomfort, nausea, sweating, headache, disorientation, fatigue [2]. Sickness varies from person to person and it is usually caused by conflicts between perceived movement and actual movement: if the player walks in VR the eyes say he is walking while the ears do not detect any movement, creating confusion to the brain. Other aspects that can induce sickness are low refresh rate and poor animations: both these things have the same effect on brain, which processes frames at higher rates or expects better animations.

Since it is a subjective feeling, the most common way of measuring VR sickness is through questionnaires. The standard methodology for measuring sickness is the *Simulator Sickness Questionnaire (SSQ)* by Kennedy, Lane, Berbaum and Lilienthal [16]. An interesting alternative approach is to monitor the postural activity of the participant [17]: it seems that motion sickness is related to postural stability and that there are differences in postural activity between people who are experiencing sickness and people who are not.

Situation awareness

A general measure that applies also for VR is situation awareness. It consists in having the awareness of what is happening in the surrounding environment and what may happen in the future, being ready to handle the situation that will arise. A famous approach to measure it is the *Situational Awareness Rating Technique (SART)* [18] originally developed by Taylor and Selcon in 1990 for evaluating pilots. It is a post-trial questionnaire that asks the user to rate 10 dimensions with a number from 1 to 7.

Workload

Another measure that can be useful in VR is workload, meant as the effort needed to complete a task. The most used technique is the *NASA Task Load Index (NASA-TLX)* [19], a questionnaire divided in two parts: the first one consists of 6 rating subscales, the second one is a personal weighting of these subscales.

2.3.4 VR headsets

This section presents the most important available VR devices, starting from the computer-connected (tethered) headsets and continuing with mobile ones. Finally, standalone VR devices are introduced.

Tethered VR headsets

Computer-connected VR systems take advantage of the computing power of the machine they are connected to, so they can give the user complex environments and experiences. The headsets provide head tracking and motion tracking - generally through external base stations - so the user can move with 6 degrees of freedom (DOF) and can interact with the virtual world in a lot of possible ways.

HTC Vive HTC Vive system consists of a headset, two controllers and two base stations to track body movements. The display has 90 Hz refresh rate and 110 degree field of view, with a resolution of 1080x1200 per eye. The base stations also track the controllers, allowing an advanced interaction with the virtual environment. Together with several sensors, the headset also includes a front-facing camera. In 2018 HTC launched the Pro version of the Vive, fitted with a higher-resolution display, attachable headphones and a second camera.

Oculus Rift Oculus initiated a Kickstarter campaign in 2012 and in 2014 it was acquired by Facebook. The system includes two Touch controllers and the headset hardware is basically the same as the HTC Vive's one. Tracking is obtained with *Constellation*, an optical-based tracking system that detects IR LED markers on the HMD and controllers.

Sony PlayStation VR PlayStation VR is Sony's proprietary VR system only compatible with PlayStation 4. It supports only PS4 ad-hoc games but there is the possibility to play any other PS4 game like if it was on a very large

screen. Headset display has a resolution of 960x1080 per eye, a 100 degrees field of view and a native refresh rate of 90 or 120 Hz. Regarding the controller input, the system supports both regular *DualShock 4* or *PlayStation Move* controllers. Players need a *PlayStation Camera* to track the HMD and the Move controllers.

Mobile VR headsets

A mobile VR system consists of a case that holds the smartphone and two lens that separate the display in two parts, one per eye. Since the only available sensors are those included in the smartphone, these VR systems can not track body movements and therefore they can just count on 3 DOF (head rotation). Since they are generally simpler and less powerful than the tethered ones, mobile VR systems are usually quite cheaper.

Google Daydream View Daydream is the enhanced successor of the *Google Cardboard*, a very basic cardboard-made headset with two lenses that turns the smartphone into a VR system. Daydream software is built into the Android operating system since the Nougat version. The package comes with a wireless touchpad controller that can be tracked with on-board sensors.

Samsung Gear VR Gear VR is a system only compatible with some high-level Samsung devices, it contains a controller equipped with a touchpad and a motion sensor. The development was carried out by Samsung in collaboration with Oculus, which took care of the software distribution.

Standalone VR headsets

Standalone headsets are a new type of VR systems that does not require a computer or a smartphone in order to work. Since they have almost the same technical specifications, these devices computing power can be compared to that of smartphones, even if they are optimized for VR applications.

Oculus Go This device was developed by Oculus in collaboration with Qualcomm and Xiaomi. The HMD has 3 degree of freedom and is equipped with a 5.5-inch display with a resolution of 1280x1440 per eye, a Snapdragon 821 processor and comes with a 32GB or 64GB storage. The system also includes a wireless controller.

Lenovo Mirage Solo Mirage Solo is a brand new device and it works with Google Daydream platform. Technically speaking, its display is similar to the Oculus Go's one but the device is powered by a Snapdragon 835 with a 4GB RAM and it has a better tracking system, that gives the headset 6 DOF (but only 3 for the controller).

2.3.5 Tracking devices

Together with headsets, a lot of different sensors and devices can be used to enhance VR experience. The most significant improvement in interaction is probably

tracking. Most HMDs have an integrated head tracking system that allows 6 DOF movement, and controllers are usually tracked making it possible to interact with the environment. However, there exist external systems that enable advanced tracking functionalities such as full body tracking, hand tracking or eye tracking.

Body tracking

Body tracking makes possible to follow movements of the whole body, included arms and legs. It is a technique often used in movies and videogames to animate digital characters in CGI (in this case it is known as *motion capture*). There are several companies providing body tracking solutions, here follows a list of some of the most valuable products currently available.

HoloSuit It is a suit that allows full body motion tracking and capturing thanks to the many sensors it has embedded. It also provides haptic feedback and it is compatible with the most important operating systems and development environments (*Unity*, *Unreal Engine*).

HTC Vive Trackers Trackers are small disc-shaped devices that work with HTC Vive and they can be attached to any real physical object in order to track its movements. They can be also connected to the user to obtain body tracking or to replace Vive controllers.

Optitrack Optitrack is one of the largest motion capture companies in the world, providing powerful tools and devices able to track small markers from long distances.

Perception Neurons This system is composed of small units called Neurons that are placed to the various parts of the body, for example to fully track a person 11 to 32 Neurons are required. It is compatible with most 3D modeling and animation programs.

PrioVR PrioVR uses sensors attached to key points of the body to provide full tracking without the need of a camera. It comes with small controllers and it works with both Unity and Unreal Engine.

Orbbec Orbbec produces long-rage depth cameras that can be used with the proprietary body tracking SDK to fully detect the human body. It works with Windows, Linux and Android.

Senso Suit Senso Suit is a kit composed of 15 modules, each containing a sensor and a vibrating motor, that make it possible to achieve full body tracking. SDK available for Unity, Unreal Engine, C++ and Android.

VicoVR VicoVR is a wireless device that provides full body tracking to mobile VR headsets without the need of body-attached sensors. The device look is similar to *Microsoft Kinect* (discontinued) and is compatible with Android, iOS and the main mobile VR headsets.

Xsens Another big name in tracking industry, Xsens provides a variety of solutions for full body tracking, including suits or strap-based sensors kits. It does not need a camera and works with almost every 3D animation program.

Hand tracking

Hand tracking is a complex aspect of artificial intelligence that allows to detect and track hand movements only with a camera, without the need of any controller or external device. Here are presented two devices and a software library capable of detecting hand movements.

LeapMotion LeapMotion tracking system is a camera that needs to be attached to a VR headset in order to track hands with a field of view of 135 degrees. It works with both HTC Vive and Oculus Rift.

uSens Fingo Fingo consists of a camera similar to LeapMotion that works with both tethered and mobile VR headsets, but it is optimized for mobile. Currently it is only compatible with Unity.

OpenCV OpenCV (Open source Computer Vision) is a software library that provides several computer vision features, including hand tracking. It is written in C++ but there exist wrappers for other programming languages.

Eye tracking

Eye tracking is the process of detecting eye position and gaze. It is usually implemented with lights and cameras that analyze eye movement and detect the direction of gaze. Eye tracking is used in a lot of different fields such as safety (for example in a car), advertising, marketing and psychology.

aGlass aGlass is an eye tracking module compatible with HTC Vive. It is composed of two lenses that are placed above the Vive lenses and provide low-latency eye tracking with a field of view of 110 degrees.

Fove Fove is a VR headset with an incorporated eye tracking module. It is compatible with SteamVR and OSVR and supports both Unity and Unreal Engine.

Pupil Pupil Labs produces binocular eye tracking add-ons for the leading VR headsets. It comes with open source software and it works with Unity.

Tobii Tobii proposes an hardware eye tracking development kit for HTC Vive or a retrofitted version of the Vive with an eye tracking integration.

2.4 Physiological signals

Physiological parameters are vital signals that describe a person's functions and activities state. There exist many different physiological signals, measurable with a lot of different sensors and devices. This section presents a wide variety of physiological sensors, from cheap wearable devices to expensive professional kits.

2.4.1 Wearable devices

Empatica E4 Wristband Wearable band and app for visualization and analysis, it comes with mobile API and Android SDK.

Sensors: photoplethysmogram (PPG), accelerometer, EDA, thermometer.

EQ02 LifeMonitor Device that can be worn on the chest and that stores or transmits the data to a mobile phone or a computer.

Sensors: ECG, respiration, skin temperature, accelerometer.

Helo LX Smartband and app, it is compatible with Windows, Android, iOS.

Sensors: heart rate, breath rate, blood pressure.

Microsoft Band 2 Smartband by Microsoft, it can also work with Cortana. The app is available for Android, iOS and Windows Phone.

Sensors: heart rate

Polar H10 Heart rate sensor to attach on the chest, it comes with an internal memory and it also works with third-party applications.

Sensors: heart rate

Xiaomi Mi Band 2 Band mainly used to monitor fitness exercises and to track sleep activity.

Sensors: accelerometer, heart rate

2.4.2 Professional kits

Biopack Systems Hardware and software platform that provides professional measurements and analysis for research and education. The main module can acquire up to 16 different channels but it has no internal memory and it needs an external power supply.

BiosignalsPlux BiosignalsPlux makes research kits that include 4 or 8 sensors and a 4 or 8 channels wireless hub, which has internal memory and battery. It works with third parties applications and sensors and comes with Android, C++, Python, Java and MATLAB APIs.

Libelium MySignals Software (uses an includes box screen) or hardware (uses Arduino) versions, acquired data is sent to the cloud and can be visualized on the web or on the mobile app. Up to 18 different sensors can be connected to the system. Available SDK for the hardware version.

MindMedia NeXus-10 Wireless acquisition system that supports 8 different signals, it includes a software for visualization and synchronization, together with a SD card and a battery for ambulant recordings.

Thought Technology Biofeedback System 5 or 8 channels system, it comes with proprietary software for data visualization and includes a memory card for offline acquisitions.

Figure 2.4 shows a complete comparison between these professional kits.

		Number of sensors	Sensors	APIs	Wireless	Power supply	Third-party integration	Offline acquisition	Resolution	Sampling rate	Cost
Biosignalplux	Explorer	4	4 to choose from: accelerometer, EMG, ECG, EEG, EDA, respiration, force, temperature, light	Android, C++, Java, Python, C#	Yes	Battery	Yes (with other Biosignalsplux devices or third-party sensors - optional cable)	Yes (optional internal memory)	Up to 16 bit per channel	Up to 3000 Hz	1 060/1 459 € (internal memory)
	Researcher	8	8 to choose from: accelerometer, EMG, ECG, EEG, EDA, respiration, force, temperature, light	Android, C++, Java, Python, C#	Yes	Battery	Yes (with other Biosignalsplux devices or third-party sensors - optional cable)	Yes (optional internal memory)	Up to 16 bit per channel	Up to 3000 Hz	3 560/3 959 € (internal memory)
	Professional (includes all add-ons for data analysis)	8	8 to choose from: accelerometer, EMG, ECG, EEG, EDA, respiration, force, temperature, light	Android, C++, Java, Python, C#	Yes	Battery	Yes (with other Biosignalsplux devices or third-party sensors - optional cable)	Yes (internal memory)	Up to 16 bit per channel	Up to 3000 Hz	4.965 €
Thought Technologies	Biofeedback Intermediate System	5	To choose. Predefined: respiration, skin conductance, temperature, blood volume pulse, EMG	Developers Tools	No	Battery	There are some possible solutions	Yes (flash memory card)	14 bit	2 channels 2048 Hz, 3 channels 256 Hz	7.442,30 €
	Biofeedback Expert System	8	To choose. Predefined: 2x respiration, skin conductance, temperature, blood volume pulse, 2x EMG, EKG	Developers Tools	No	Battery	There are some possible solutions	Yes (flash memory card)	14 bit	2 channels 2048 Hz, 6 channels 256 Hz	About 8 500 €
Libelium MySignals	Software	11	Snore, body position, ECG, EMG, GSR, breathing, temperature, spirometer, SPO2, blood pressure, glucometer	REST	Yes (BLE)	External power supply	Yes (only third-party sensors)	Yes (cloud)	10 bit ?	?	1 967/2 080 € (BLE version)
	Hardware	11	Snore, body position, ECG, EMG, GSR, breathing, temperature, spirometer, SPO2, blood pressure, glucometer	REST or SDK	Yes (BLE)	External power supply	Yes (only third-party sensors)	Yes (cloud or SDK)	10 bit ?	?	1 532/1 627 € (BLE version)
MindMedia NeXus-10		8	4 ExG channels of EEG, EMG, ECG and EOG and 4 AUX channels for peripheral signals like heart rate, relative blood flow, skin conductance, respiration, temperature	Proprietary or C++	Yes	Battery	Yes (third-party sensors)	Yes (flash memory card)	24 bit	Up to 8192 Hz	7.015 €
Biopac System		16	ECG, respiration, EEG, EDA, gioniometer	Proprietary	No	External power supply	?	?	16 bit	400 kHz (aggregate)	31.180 €

Figure 2.4. Professional physiological kits comparison

2.5 Used technologies

This section presents the technologies used in the project and explains the reasons why they were chosen.

Virtual reality

The framework is meant to be used on a desktop environment, so for what concerns virtual reality the only possible choices were the Oculus Rift and the HTC Vive. The latter was the chosen one for the project. The main reason behind this decision was that it allows the user to physically move better than the Oculus, considering that its base stations can track a room-scale environment. Since the application can be used in a variety of different projects and experiments, this choice was the best for a general-purpose framework.

For what concerns the development environment of the virtual experience, the chosen tool was *Unity3D*, a game engine available for Windows and MacOS that also supports virtual reality. Its scripting language is C#, it is easy to use, compatible with HTC Vive and the created games can be launched in stand-alone mode, so it was the best choice for this project.

Programming language

The application will handle the majority of the logic of the framework (see section 3.2). It needs to be able to play videos easily and with the regular control commands (play, pause, stop), it has to support signals graphs representation and the development phase should not cost too much effort. From these considerations we can conclude that the best programming language to use is Java. There are a lot of useful external libraries that can be imported from it, it has ready-to-use media player classes, nice graphical interfaces tools and it is cross-platform.

Physiological sensors

The last choice to make is about which sensors to use in the framework. There were several aspects that were taken into account in order to decide which professional kit was the best among those reported in figure 2.4. The kit should support the larger possible number of different sensors; it should be wireless, so that it is easier to connect and more practical to use; it should have internal memory and a battery, so that it can be also used for other kind of applications (e.g. external experiments); it should be compatible with third-party hardware or software and provide APIs in order to be usable with custom services and applications. The only option that satisfied all these condition was the BiosignalsPlux Researcher kit: 8 sensors, wireless, portable and with well documented API for several programming languages.

Chapter 3

Design

This chapter describes the design of the system, starting from the initial idea and continuing with the multiple steps until the final scheme.

3.1 Similar systems

Before thinking about the design itself, a research on similar projects was conducted in order to better understand how this kind of systems are realized. The list includes not only projects that are close to this one as a whole, but also projects that provide just some similar features. Here follows the research result, divided in the various fields regarding the functionalities of the system.

PhysioVR [20] is a framework developed to integrate physiological signals in mobile VR applications. Signals are acquired using wearable devices connected to the smartphone via Bluetooth and include heart rate, electroencephalography (EEG) and electromyography (EMG). The system allows to record data and to communicate with the VR content allowing an external event triggering using a real-time control.

MuLES [21] is a tool for acquiring and streaming EEG signals that aims at facilitate the development of brain-computer interface programs. The system allows to replay a session and there are clients developed for use on different platforms and in various programming languages.

PhysSigTK [22] is a toolkit for accessing low-cost physiological signals hardware from Unity. It aims at helping developers to exploit engagement in their effective games.

RehabNet [23] is a system that helps patients rehabilitation through serious videogames that monitor physiological parameters. It supports a large variety of external physiological sensors.

Bicycle4CaRe [24] is a tool for domestic physical training that uses VR and sensors to monitor user's activity. It simulates a home environment and collects data both from it and from the user, providing real-time feedback.

Cube [25] is a VR simulation platform for the supervision of personnel working in critical infrastructures. It collects physiological data during the virtual experience to identify the optimal physiological profile of personnel for using it in real scenarios.

Athena [26] is an analytics platform that acquires data from a combat simulator, combining this data with physiological signals of the player, and sends him body feedback.

VAST [3] is a virtual reality system for social communication that also collects physiological signals to determine the user's anxiety level.

VR-SAAFE [27] is a VR-based system developed for understanding facial emotions in subjects suffering of schizophrenia. The system uses an eye tracker and physiological signals to monitor user's eye gaze and emotions.

3.1.1 Design

The above systems use different approaches for what concerns design.

- *PhysioVR* is composed of two layers: the first one deals with synchronizing the devices and streaming the data, the second one is a Unity package that receives and analyzes data and physiological signals. There are also other two modules for external control and data recording.
- *MuLES* acts as the server part of a client-server architecture. It handles the communication between the several EEG sensors and transmits their data to the client applications. The devices are connected with their own drivers and the connection with the clients is realized through TCP/IP.
- *RehabNet* architecture is based on three building blocks: an hardware layer that deals with devices connection, a control panel that filters and cleans data, and a web interface for accessing the rehabilitation tools. The whole system is organized as a client-server application.
- *Bicycle4CaRe* is composed of different biofeedback devices connected to the PC via an Arduino board,ca desktop application and a VR simulated environment. The app contains the main functionalities and allows to control the experience.
- *Athena* is consists of a data acquisition layer that collects the signals from the sensors and the game data from the combat simulator. This layer sends data to IBM's Infosphere Streams, that processes it and generates analytics for storage.
- *VAST* is divided in three modules. The first one presents the tasks, the second one adapts and changes tasks according to the users behavior and the last one is the data acquisition layer.
- *VR-SAAFE* is divided in three components too: task presentation environment based on Unity, eye tracking application and the physiological signals acquisition module.

3.1.2 Physiological sensors

This section presents the physiological sensors used by similar projects. This could help understanding which are the most used and useful signals in this kind of applications. *PhysioVR* uses only three signals:

- heart rate, acquired through Android wearable devices such as smartbands, smartwatches or camera-based sensors;
- EEG, obtained from a low-cost -wearable headband called Muse BCI;
- EMG, acquired with Myo Armband, a device that contains 8 sensors and that recognizes gestures.

PhysSigTK uses sensors considering criteria such as ease of use, price, software support and low-level access to data. Therefore the employed devices are four, measuring several different signals:

- Empatica E3/E4, that measures EDA, blood volume, temperature and movement;
- Wild Divine Iom, that provides skin conductance level (SCL) and heart rate;
- e-Healt, an Arduino-based device that supports different sensors. In this case heart rate, galvanic skin response (a.k.a. EDA), ECG and breathing activity;
- NeuroSky Mindwave, a low-cost EEG sensor that also provides a heart rate sensor for the ear.

For what concerns *RehabNet*, it uses protocols such as the *Virtual-Reality Peripheral Network (VRPN)* and *Open Sound Control (OSC)* for integrating a large variety of external devices and sensors (trackers, haptic devices, analog input, sound, etc). In addition to this, it natively supports three sensors:

- EEG acquired with Emotiv EPOC, a wireless EEG headset;
- EMG, collected through Myomo mPower 1000, a prosthetic arm that also integrates sensors;
- kinematic data, acquired with the Microsoft Kinect.

VR-SAAFE monitors physiological signals and eye gaze:

- Biopac Bionomadix measures PPG (from which heart rate can be extracted), skin temperature, EMG, respiration and galvanic skin response;
- Tobii X120 for eye tracking.

Bicycle4CaRe measures both user's parameters and environmental conditions. The latter are not relevant for the project of this thesis; the former are:

- heart rate acquired with a pulse-oximeter sensor;
- breath rate measured through a respiration monitor belt;
- blood pressure collected with a sphygmomanometer.

3.1.3 Connectivity and libraries

The project application will need to connect to the devices and to acquire signals. In order to be able to do it, similar systems techniques and methodologies were explored. Here are reported some examples of connectivity and external libraries approaches.

- *PhysioVR* transmits data over the UDP protocol. In this way the communication can be multicast and the transmission of all the physiological signals is done using a single port. Data can be accessed even from external devices with UDP capabilities. Users can access from remote and interact with the system bidirectionally, thus receiving data and sending feedback. The application also provides client scripts for different programming languages such as Java, C#, Python, MATLAB.
- *MuLES* acquires data from the hardware using third party APIs, SDKs, drivers or implementations of the specific device communication protocol. On the other hand, transmission to the client application is realized through the TCP/IP protocol, which allows to send data to a local network or over the Internet. This means that client scripts can be written in any programming language supporting basic socket programming, such as Java or Python.
- *RehabNet* sends data to a smartphone running the app using a custom implementation of the UDP protocol. It also connects to an analysis and tracking tool. As anticipated above, the system makes use of VRPN and OSC protocols in order to connect to other external devices or libraries, for example the OpenViBE BCI software.
- The interesting thing about *Cube* is that it provides different connectivity solutions that can be used simultaneously or not. An example of it is ECG acquisition: it can be wired or wireless (Bluetooth) transmitted to a local machine, transmitted to a smartphone over the Internet or using a satellite connection

3.1.4 Events and triggers

Since the system should deal with key events and markers, here are presented the solutions adopted by *PhysioVR* and *Bicycle4Care*.

- *PhysioVR* acquires data from the UDP connection and makes it available inside the Unity environment. This brings developers to have physiological data usable inside the experience, allowing some game parameters to be bound to physiological signals, or events to be triggered when certain conditions are verified. Another way of creating events is to have external inputs: the VR user can receive triggers influencing the experience scenario from an external application that monitors in real-time the person's vitals.
- *Bicycle4Care* reads data in real-time and guides the user through the experience with ad-hoc messages. Like for PhysioVR, events can be triggered when some parameter exceed a certain threshold.

3.1.5 Synchronization

After the acquisition, data needs to be synchronized. Among the presented systems, only *Athena* and *VAST* use synchronization or explicitly write about it.

- *Athena* acquires data from the combat simulator and the sensors and then it synchronizes them. This data is composed of physiological signals, game events like firing and target hits and haptic device activations. The adopted synchronization technique is not mentioned.
- *VAST* acquires real-time physiological signals and VR data in a synchronized manner for offline analysis. Markers are used to synchronize the different signals and the virtual environment.

3.1.6 Extensions

This section is not related with similar systems but presents two ideas for extension and improvement of VR experiences that can be useful for the project development.

- The first idea comes from S. Otsuka and others, who think that physiological signals could be easier acquired if the sensors are embedded in a VR headset [28]. They propose a new HMD that contains some sensors: their first prototype consists of a VR headset with an integrated PPG sensor that can measure heart rate. This system was tested and gave positive results, so this kind of devices could be a real and interesting thing for the future of VR and physiological signals research.
- The second one is about VR and eye tracking. A paper about this is the one by Pradeep Raj and others [29]. It presents the design of a VR-based social communication platform integrated with eye tracking technology. The authors believe that this kind of technology can be useful for presenting and analyze social situations and communication.

3.2 Design

After analyzing similar systems with their potentially useful features, now we can introduce the actual design of this thesis project. Since there have been problems and changes during the development phase, the general design has been modified during the implementation of the system. The development was divided into three prototypes, each adding a new functionality to the system:

1. real-time visualization of the VR video, the camera video and the signals stream;
2. recording and synchronization of the streams;
3. replay of the streams and insertion of key markers.

Here are presented all the design prototypes, from the initial idea to the final result.

3.2.1 Initial design

General architecture

The system architecture follows the actors involved in the process: the user and the examiner. The participant is the person who takes part in the experiment: he is equipped with a VR headset (HTC Vive), physiological sensors (BiosignalsPlux kit) and a webcam pointing to his face. The communication between these sensors and the examiner's computer is realized through controllers. The examiner component is constituted by the controllers, a recording and synchronization layer that saves the data on the storage and a Java application used to see in real-time what the user is experiencing and to start/stop the recording. Concerning the offline mode of the system, it is represented by two additional blocks. The first one deals with the offline part of the application itself, while the other represents the external applications that can be used to analyze data (e.g. MATLAB or Python). This last part is not covered by the project. Figure 4.1 shows an overall view of the system.

Controllers

Regarding the controllers, in the initial design they communicate with the Java application, that executes every task. In this case the application represents both the real-time and offline Java applications showed in figure 4.1. Starting from VR, the development platform is Unity and to send the video stream to the Java application we are going to use the *RockVR* plugin, which allows to record a video of what the user sees through the headset and send it to a streaming server in real time. The application is then going to read the stream and play it using the *MediaPlayer* Java class. For the physiological sensors we can use the official *Plux* API, which allows to easily acquire the signals data from Java. Finally, for the webcam we are going to use the *JavaCV* library, a wrapper of the *OpenCV library*, for visualizing and recording the video stream. For a detailed description of these tools and libraries see section 4.2. The controllers schema is represented in figure 4.2.

3.2.2 First prototype

Original schema

The first prototype only concerns the real-time visualization of the VR video, the camera video and the signals stream. Therefore, we can focus on the left side of figure 4.1: we only consider the user, the controllers and the real-time visualization application (figure 4.3). For what concerns the controllers, we are interest only in the real-time tasks, so the diagram is the one in figure 4.4.

Implementation changes

The original diagrams changed when implementing the prototype. The main problem was about the VR video streaming: encoding it, sending it to the streaming server (localhost) and decoding it generated a delay of about 3 seconds. This resulted in an out of sync visualization, since camera and physiological signals were

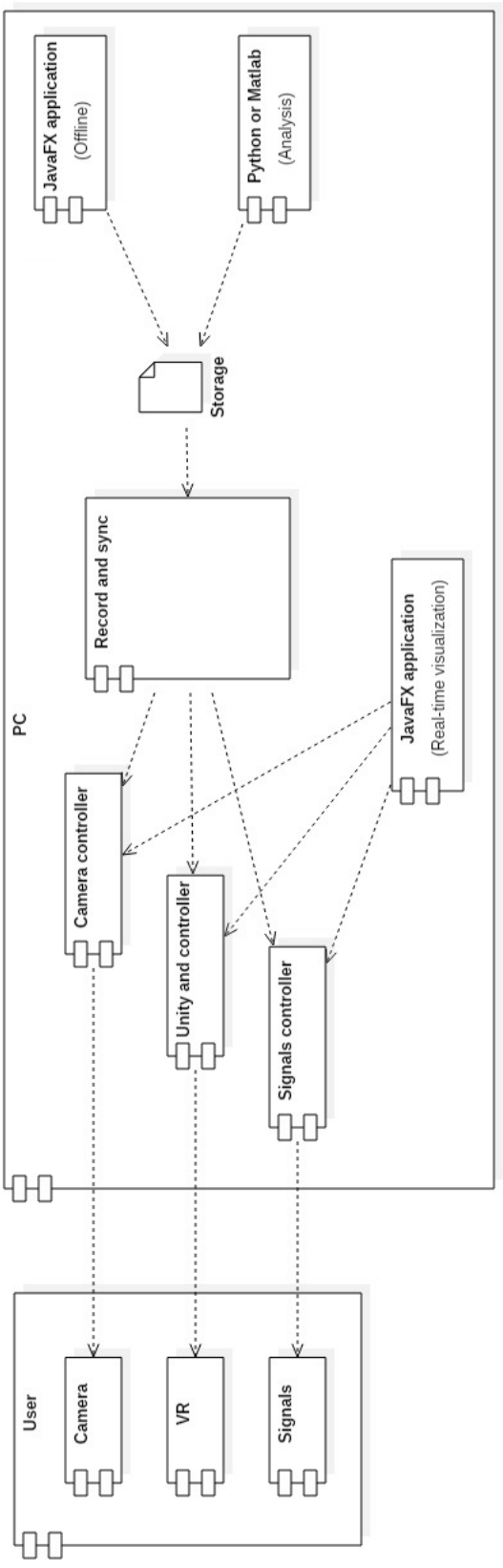

Figure 3.1. Initial general architecture

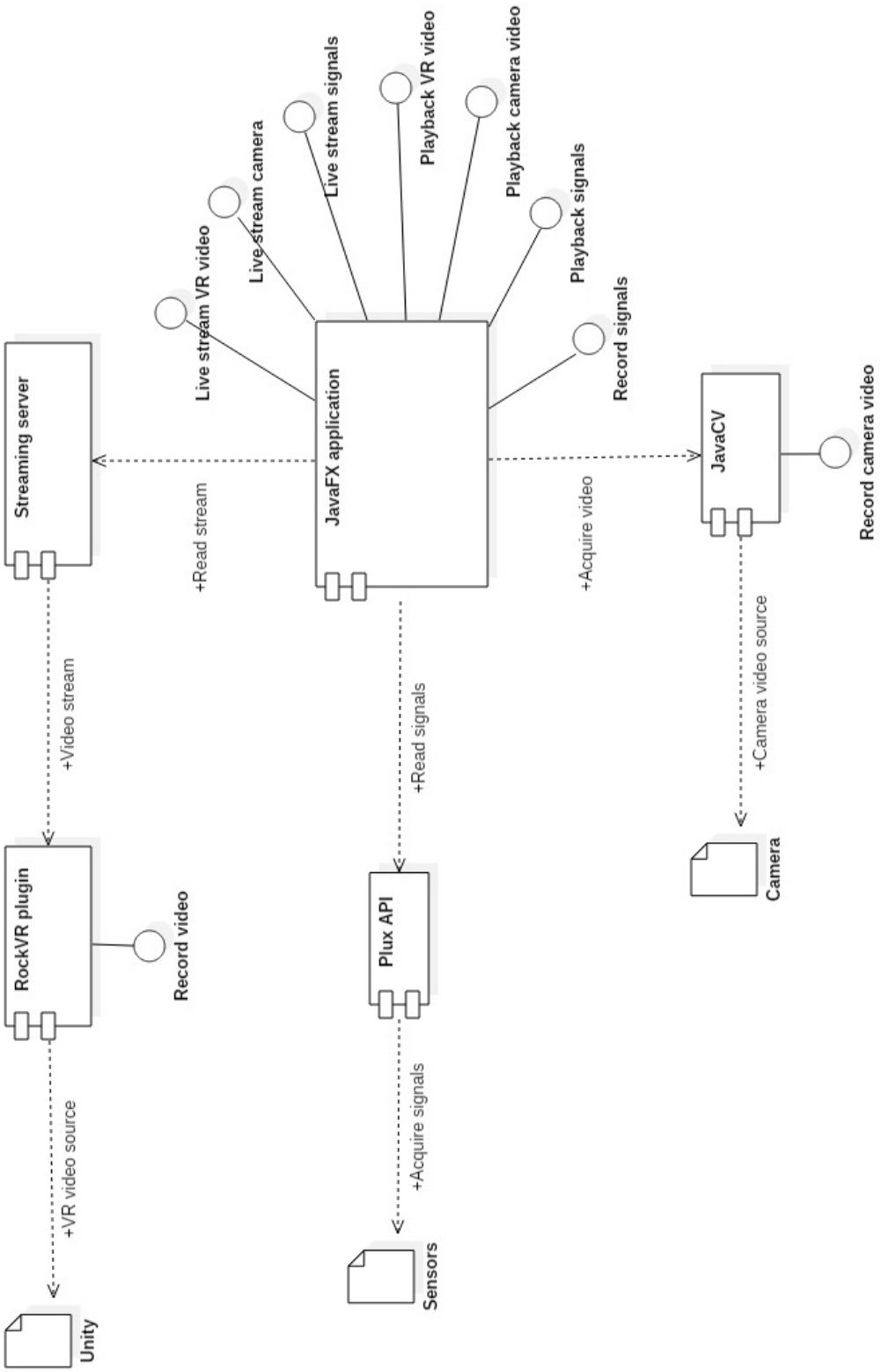

Figure 3.2. Controllers initial schema

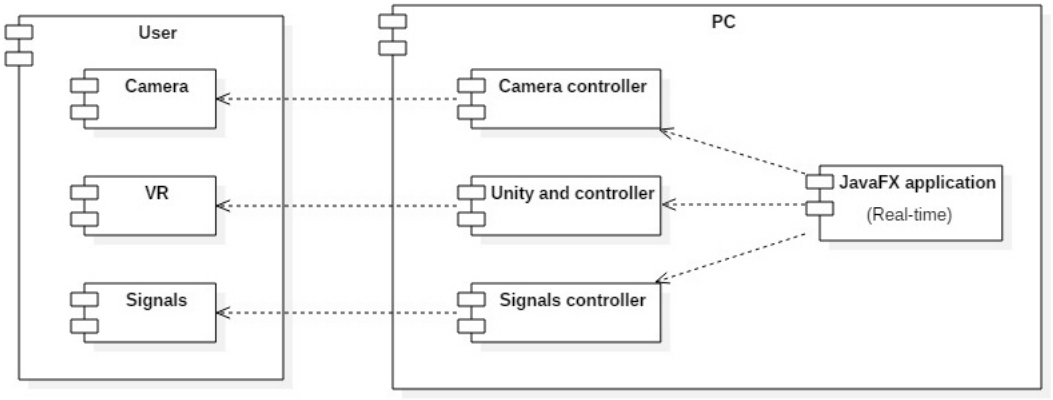

Figure 3.3. First prototype original architecture

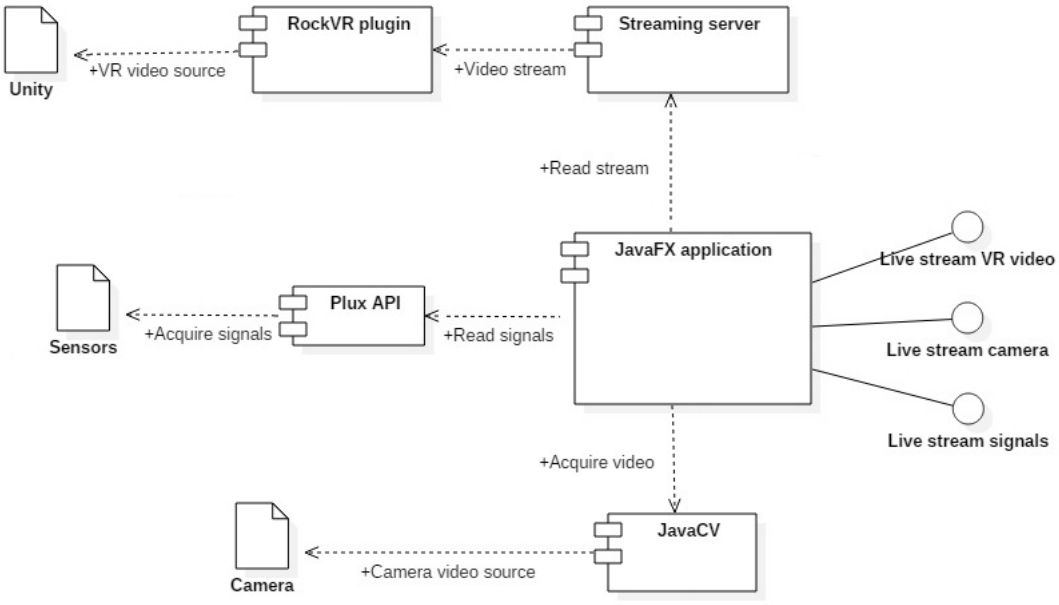

Figure 3.4. First prototype controllers original schema

synchronized but VR was not. This behavior was not acceptable. Even if it could have been possible to synchronize the three sources after the recording, it was not possible to see everything in real-time, failing to satisfy the first requirement.

Trying to find a solution to this problem was not easy. Playing the VR video stream from the Java application means to export what happens in Unity and to read it from an external application. The only suitable technique found was the above one. With it not working, the only reasonable solution was to use two screens and split the real-time part of the application in two pieces: one handling camera and signals visualization and one the VR visualization. The former will be handled by the Java application itself on the first screen and the latter by Unity on the second screen (figure 4.5).

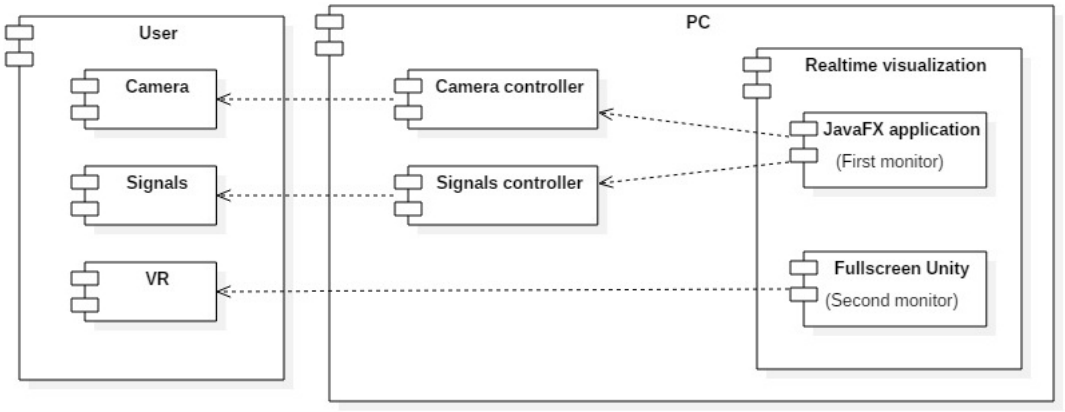

Figure 3.5. First prototype architecture implementation

As regards the controllers, we can simply get rid of the VR part because it has no more connections with the main application (figure 4.6).

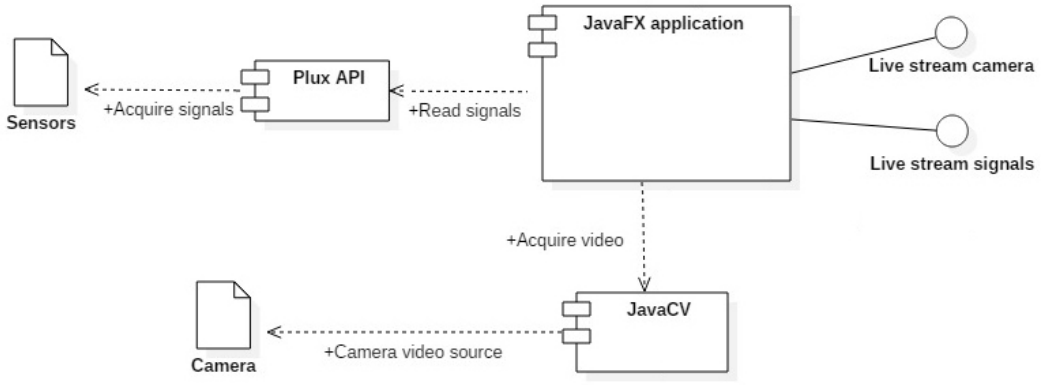

Figure 3.6. First prototype controllers implementation

3.2.3 Second prototype

The second prototype was supposed to add the recording functionality to the system. The general schema is exactly the same as the one in figure 4.5, but with the addition of the *Record and sync* module and of the storage (figure 4.7).

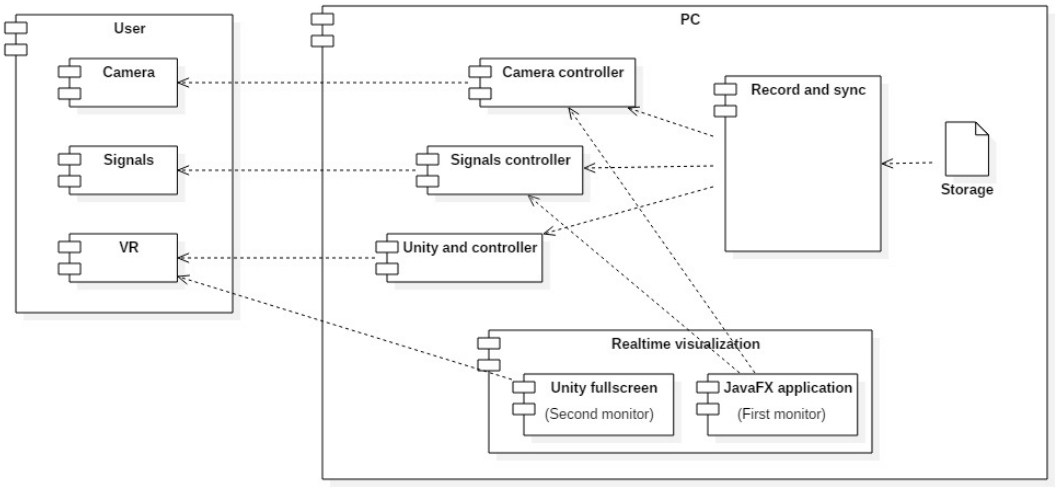

Figure 3.7. Second prototype architecture

Since from the first prototype architecture change we can not visualize the VR video in real-time from the Java application, we can not record it either, so we need to do it from Unity itself. This is the reason why there is also a Unity controller in the diagram (figure 4.8): this component is the RockVR plugin, and it will handle the recording of what the user sees and store it on the hard drive.

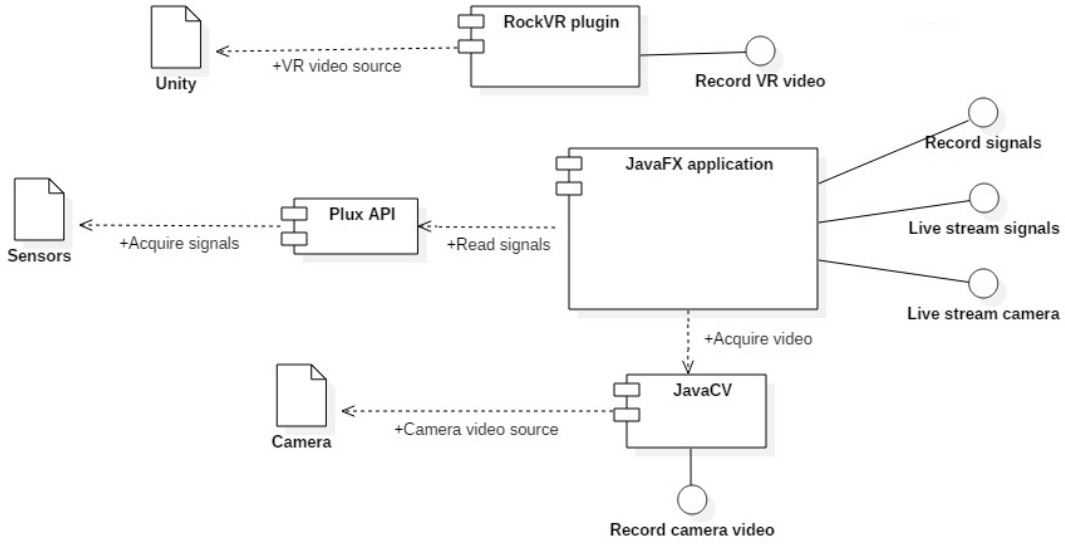

Figure 3.8. Second prototype controllers

This new consideration brings the system to be composed of two modules: the first one is a Java application that plays and records the camera and signals streams, the second one is a Unity package that records the VR video. At the end of the acquisition, the 3 files – VR video, camera video and signals data – are synchronized (see section 3.3) in order to be replayed and analyzed properly.

3.2.4 Third prototype

The third and last prototype represents the final version of the system. It adds the replay feature to the application, included the option to insert markers on specific moments of the playback.

Both the architecture and the controllers schemas are similar to the initial idea one, except for the separation of Java and Unity blocks (figure 4.9 and 4.10).

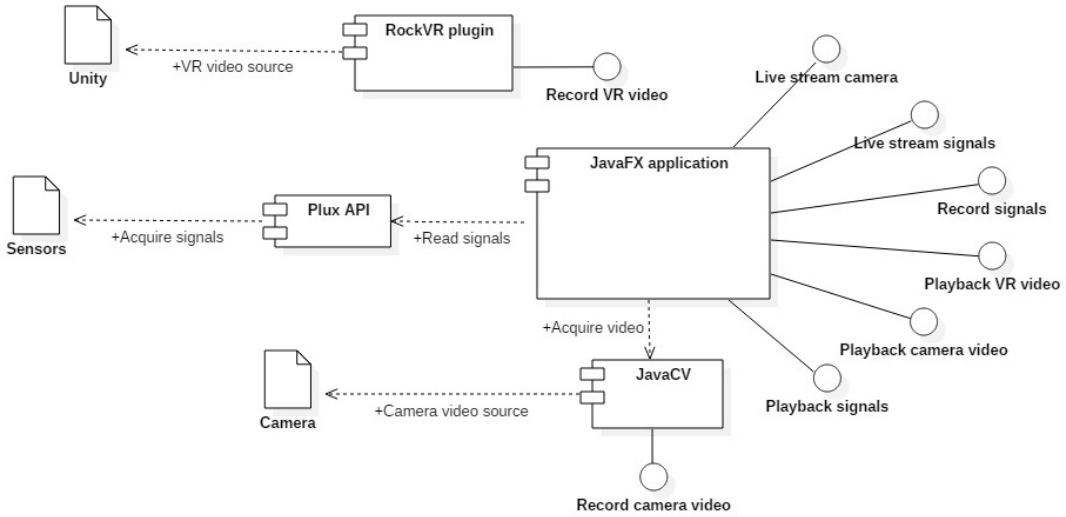

Figure 3.9. Third prototype controllers

3.3 Synchronization

Synchronization can be considered as one of the most important parts of this project. The value of the framework is that it gives researchers the possibility to have a ready-to-use environment that allows them to save valuable time, since everything they record can be easily synchronized. This section gives an overview of the synchronization problem and describes the solution adopted to solve it.

Synchronization issue

The camera video and the signals are acquired from the same application. Since the recording starts when the examiner clicks a button, both the sources start being stored approximately at the same time. There should be no relevant delays (i.e. more than one or two tenths of a second), therefore they should be already

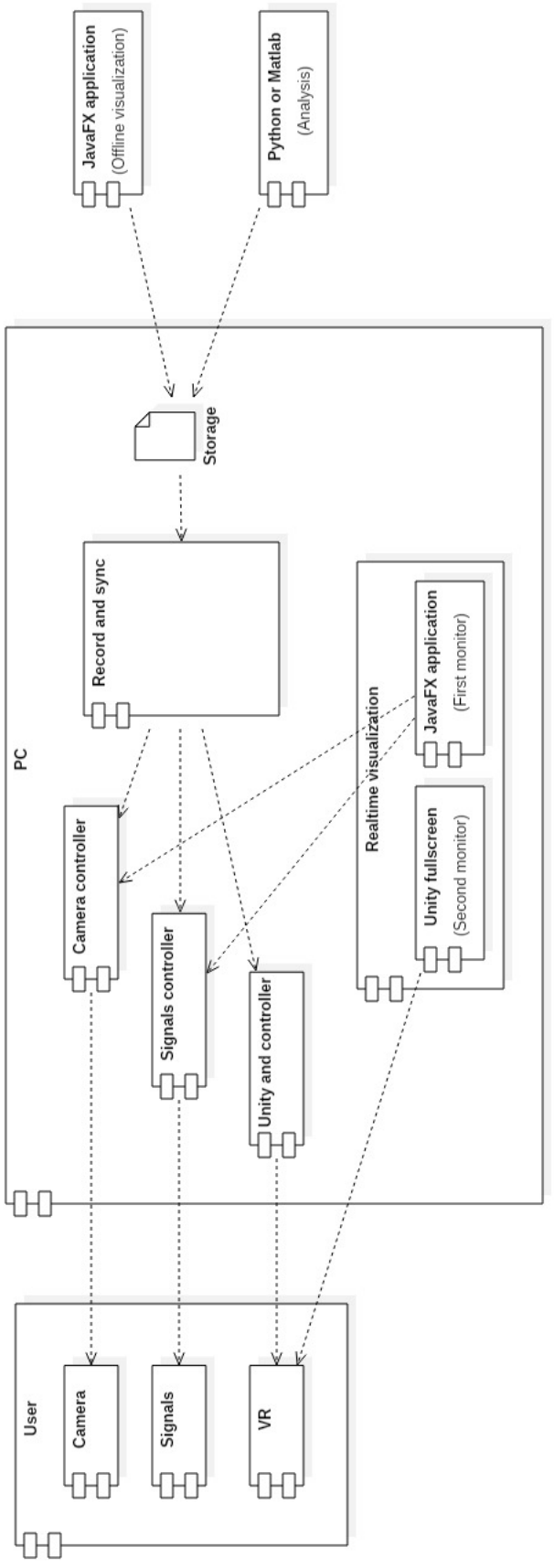

Figure 3.10. Third prototype architecture

synchronized. This means that if the implementation is done correctly, the camera video and the signals file do not need further processing in order to be synchronized.

The problem arises when we also consider VR video. As stated above, it is displayed on the Unity window itself and is recorded through a plugin. Just as the other recording, also this one is initiated by the examiner: since the two recordings events are generated by a human actor, there is no way they start at the exact same time.

We can consider the following as an example of a real usage of the platform. The examiner has already run the application and Unity, configured everything and started seeing the streams in real-time. Now he wants to record the whole session. He first starts recording from the Java application, acquiring signals and camera video, and then switches to the second screen and starts recording from Unity. Even if this two actions are executed very quickly, there will be an unavoidable delay (at least half a second) between the two recordings. This fact is not acceptable with respect to the framework idea and requirements, thus we need to find a way to really synchronize the files. Synchronization means to align the files or to cut them in order to display at the same moment events and data that happened at the same time. The ideal solution would have been to find a way of starting the two recordings simultaneously, without the double intervention of the examiner. This would have required some kind of connection between Java and Unity and was not easy to achieve. The solution adopted to address the problem was another: employ system timestamps.

System timestamps

Timestamps are sequences of characters representing a time unit, generally indicating a date and a time. In this case the used timestamps are the *Unix timestamps*, referring to the total number of seconds elapsed since midnight, January 1, 1970 UTC. When retrieved from a computer, this value depends on the underlying operating system. OS in fact measure time with different milliseconds granularity, for example Windows used 55 milliseconds until the 98 version and 10 milliseconds after.

The general idea about how to use timestamps for synchronization is to associate to each recorded file the timestamp, in milliseconds, of the moment when the recording starts. This gives us the indication about the time when the recordings started and therefore we can compute the difference between the values to get the exact delay between a source an another. The same steps are applied to the end of the recordings.

An overview on the synchronization phase can be described as follows. Without loss of generality we can assume that the examiner starts (and stops) recording from the Java application and then continues with Unity:

1. examiner starts the recording from the Java application;
2. current timestamp is associated to signals and camera video files as *initial timestamp*;
3. signals and camera video streams begin to be saved on the storage;

4. examiner starts the recording from Unity;
5. current timestamp is associated to VR video file as *initial timestamp*;
6. VR video stream begins to be saved on the storage;
7. examiner stops the recording from the Java application;
8. current timestamp is associated to signals and camera video files as *final timestamp*;
9. signals and camera video streams stop to be saved on the storage;
10. examiner stops the recording from Unity;
11. current timestamp is associated to VR video file as *final timestamp*;
12. VR video stream stops to be saved on the storage;
13. difference between signals (same as camera) and VR initial timestamps is computed (we can name this value);
14. difference between signals (same as camera) and VR final timestamps is computed (we can name this value *delayEnd*);
15. first *delayStart* milliseconds of signals and camera video files are cut;
16. last *delayEnd* milliseconds of VR video files are cut.

This procedure guarantees a good level of synchronization, which basically relies on timestamps accuracy. All the information about source files and timestamps is stored on simple text files. Implementation details about synchronization are reported in section 4.1.

Chapter 4

Realization

4.1 Synchronization

4.2 Third parties software

4.3 Code snippets

Chapter 5

Validation

5.1 Proof of concept test

5.1.1 Protocol

5.1.2 Results

5.2 Usability test

5.2.1 Protocol

5.2.2 Results

Chapter 6

Conclusions

Bibliography

[1] B. K. Wiederhold, R. Gevirtz, and M. D. Wiederhold, "Fear of flying: A case report using virtual reality therapy with physiological monitoring," *CyberPsychology & Behavior*, vol. 1, no. 2, pp. 97–103, 1998.

[2] S. V. Cobb, S. Nichols, A. Ramsey, and J. R. Wilson, "Virtual reality-induced symptoms and effects (vrise)," *Presence: Teleoperators & Virtual Environments*, vol. 8, no. 2, pp. 169–186, 1999.

[3] S. Kuriakose and U. Lahiri, "Understanding the psycho-physiological implications of interaction with a virtual reality-based system in adolescents with autism: a feasibility study," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 4, pp. 665–675, 2015.

[4] S. Seinfeld, I. Bergstrom, A. Pomes, J. Arroyo-Palacios, F. Vico, M. Slater, and M. V. Sanchez-Vives, "Influence of music on anxiety induced by fear of heights in virtual reality," *Frontiers in psychology*, vol. 6, p. 1969, 2016.

[5] J. Llobera, M. V. Sanchez-Vives, and M. Slater, "The relationship between virtual body ownership and temperature sensitivity," *Journal of the Royal Society Interface*, vol. 10, no. 85, p. 20130300, 2013.

[6] M. D. Wiederhold, K. Gao, and B. K. Wiederhold, "Clinical use of virtual reality distraction system to reduce anxiety and pain in dental procedures," *Cyberpsychology, Behavior, and Social Networking*, vol. 17, no. 6, pp. 359–365, 2014.

[7] I. E. Sutherland, "A head-mounted three dimensional display," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pp. 757–764, ACM, 1968.

[8] Kasey Panetta, "Top trends in the gartner hype cycle for emerging technologies, 2017." `https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/`, 2017. [Online; accessed 26-June-2018].

[9] Gartner, "Hype cycle." `https://www.gartner.com/it-glossary/hype-cycle`, 2017. [Online; accessed 26-June-2018].

[10] M. Slater, "A note on presence terminology," *Presence connect*, vol. 3, no. 3, pp. 1–5, 2003.

[11] D. A. Bowman and R. P. McMahan, "Virtual reality: how much immersion is enough?," *Computer*, vol. 40, no. 7, 2007.

[12] M. V. Sanchez-Vives and M. Slater, "From presence to consciousness through virtual reality," *Nature Reviews Neuroscience*, vol. 6, no. 4, p. 332, 2005.

[13] J. S. Casanueva and E. H. Blake, "The effects of avatars on co-presence in a collaborative virtual environment," in *Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT2001). Pretoria, South Africa*, 2001.

[14] J. Tromp, A. Bullock, A. Steed, A. Sadagic, M. Slater, and E. Frécon, "Small group behaviour experiments in the coven project," *IEEE Computer Graphics and Applications*, vol. 18, no. 6, pp. 53–63, 1998.

[15] M. Slater, A. Sadagic, M. Usoh, and R. Schroeder, "Small-group behavior in a virtual and real environment: A comparative study," *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 1, pp. 37–51, 2000.

[16] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness," *The international journal of aviation psychology*, vol. 3, no. 3, pp. 203–220, 1993.

[17] G. E. Riccio and T. A. Stoffregen, "An ecological theory of motion sickness and postural instability," *Ecological psychology*, vol. 3, no. 3, pp. 195–240, 1991.

[18] S. Selcon and R. Taylor, "Evaluation of the situational awareness rating technique(sart) as a tool for aircrew systems design," *AGARD, Situational Awareness in Aerospace Operations 8 p(SEE N 90-28972 23-53)*, 1990.

[19] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*, vol. 52, pp. 139–183, Elsevier, 1988.

[20] J. E. Muñoz, T. Paulino, H. Vasanth, and K. Baras, "Physiovr: A novel mobile virtual reality framework for physiological computing," in *e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on*, pp. 1–6, IEEE, 2016.

[21] R. Cassani, H. Banville, and T. H. Falk, "Mules: An open source eeg acquisition and streaming server for quick and simple prototyping and recording," in *Proceedings of the 20th International Conference on Intelligent User Interfaces Companion*, pp. 9–12, ACM, 2015.

[22] S. Rank and C. Lu, "Physsigtk: Enabling engagement experiments with physiological signals for game design," in *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pp. 968–969, IEEE, 2015.

[23] A. Vourvopoulos, A. L. Faria, M. S. Cameirao, and S. B. i Badia, "Rehabnet: A distributed architecture for motor and cognitive neuro-rehabilitation," in

e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on, pp. 454–459, IEEE, 2013.

[24] D. Baldassini, V. Colombo, D. Spoladore, M. Sacco, and S. Arlati, "Customization of domestic environment and physical training supported by virtual reality and semantic technologies: A use-case," in *Research and Technologies for Society and Industry (RTSI), 2017 IEEE 3rd International Forum on*, pp. 1–6, IEEE, 2017.

[25] C. Cepisca, F. C. Adochiei, S. Potlog, C. K. Banica, and G. C. Seritan, "Platform for bio-monitoring of vital parameters in critical infrastructures operation," in *Electronics, Computers and Artificial Intelligence (ECAI), 2015 7th International Conference on*, pp. E–7, IEEE, 2015.

[26] C. McGregor, B. Bonnis, B. Stanfield, and M. Stanfield, "Integrating big data analytics, virtual reality, and araig to support resilience assessment and development in tactical training," in *Serious Games and Applications for Health (SeGAH), 2017 IEEE 5th International Conference on*, pp. 1–7, IEEE, 2017.

[27] E. Bekele, D. Bian, J. Peterman, S. Park, and N. Sarkar, "Design of a virtual reality system for affect analysis in facial expressions (vr-saafe); application to schizophrenia," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 6, pp. 739–749, 2017.

[28] S. Otsuka, K. Kurosaki, and M. Ogawa, "Physiological measurements on a gaming virtual reality headset using photoplethysmography: A preliminary attempt at incorporating physiological measurement with gaming," in *Region 10 Conference, TENCON 2017-2017 IEEE*, pp. 1251–1256, IEEE, 2017.

[29] P. R. KB, P. Oza, and U. Lahiri, "Gaze-sensitive virtual reality based social communication platform for individuals with autism," *IEEE Transactions on Affective Computing*, 2017.