Universitat
de les Illes Balears

DOCTORAL THESIS
2022

ANALYSIS OF USER BEHAVIOR WITH DIFFERENT INTERFACES IN 360-DEGREE VIDEOS AND VIRTUAL REALITY

Antoni Oliver Tomàs

Universitat
de les Illes Balears

DOCTORAL THESIS
2022

Doctoral Programme in Information and Communications Technology

ANALYSIS OF USER BEHAVIOR WITH DIFFERENT INTERFACES IN 360-DEGREE VIDEOS AND VIRTUAL REALITY

Antoni Oliver Tomàs

Thesis Supervisor: Antoni Bibiloni Coll
Thesis tutor: Javier Varona Gómez

Doctor by the Universitat de les Illes Balears

Antoni Oliver Tomàs

Analysis of user behavior with different interfaces in 360-degree videos and virtual reality

2022

Supervisor: Antoni Bibiloni Coll

Universitat de les Illes Balears

Departament de Ciències Matemàtiques i Informàtica

Laboratori de Tecnologies de la Informació Multimèdia

Universitat
de les Illes Balears

I, Antoni Oliver Tomàs, declare that this thesis titled “Analysis of user behavior with different interfaces in 360-degree videos and virtual reality” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Ph.D. degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has clearly been stated.
- Where I have consulted the published work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

For all intents and purposes, I hereby sign this document.

Signed: Antoni Oliver Tomàs

Palma, 2022

Universitat
de les Illes Balears

I, Dr. Antoni Bibiloni Coll of the Universitat de les Illes Balears declare that the thesis titled “Analysis of user behavior with different interfaces in 360-degree videos and virtual reality”, presented by Antoni Oliver Tomàs to obtain a doctoral degree, has been completed under my supervision.

For all intents and purposes, I hereby sign this document.

Signed: Dr. Antoni Bibiloni Coll

Palma, 2022

Abstract

Virtual reality and its related technologies are being used for many kinds of content, like virtual environments or 360-degree videos. Omnidirectional, interactive, multimedia is consumed with a variety of devices, such as computers, mobile devices, or specialized virtual reality gear. Studies on user behavior with computer interfaces are an important part of the research in human-computer interaction, used in, e.g., studies on usability, user experience or the improvement of streaming techniques. User behavior in these environments has drawn the attention of the field but little attention has been paid to compare the behavior between different devices to reproduce virtual environments or 360-degree videos. We introduce an interactive system that we used to create and reproduce virtual reality environments and experiences based on 360-degree videos, which is able to automatically collect the users' behavior, so we can analyze it. We studied the behavior collected in the reproduction of a virtual reality environment with this system and we found significant differences in the behavior between users of an interface based on the Oculus Rift and another based on a mobile VR headset similar to the Google Cardboard: different time between interactions, likely due to the need to perform a gesture in the first interface; differences in spatial exploration, as users of the first interface chose a particular area of the environment to stay; and differences in the orientation of their heads, as Oculus users tended to look towards physical objects in the experiment setup and mobile users seemed to be influenced by the initial values of orientation of their browsers. A second study was performed with data collected with this system, which was used to play a hypervideo production made of 360-degree videos, where we compared the users' behavior with four interfaces (two based on immersive devices and the other two based on non-immersive devices) and with two categories of videos: we found significant differences in the spatiotemporal exploration, the dispersion of the orientation of the users, in the movement of these orientations and in the clustering of their trajectories, especially between different video types but also between devices, as we found that in some cases, behavior with immersive devices was similar due to similar constraints in the interface, which are not present in non-immersive devices, such as a computer mouse or the touchscreen of a smartphone. Finally, we report a model based on a recurrent neural network that is able to classify these reproductions with 360-degree videos into their corresponding video type and interface with an accuracy of more than 90% with only four seconds worth of orientation data; another deep learning model was implemented to predict orientations up to two seconds in the future from the last seconds of orientation, whose results were improved by up to 19% by a comparable model that leverages the video type and the device used to play it.

Resum

La realitat virtual i les tecnologies que hi estan relacionades es fan servir per a molts tipus de continguts, com entorns virtuals o vídeos en 360 graus. Continguts multimèdia omnidireccional i interactiva són consumits amb diversos dispositius, com ordinadors, dispositius mòbils o aparells especialitzats de realitat virtual. Els estudis del comportament dels usuaris amb interfícies d'ordinador són una part important de la recerca en la interacció persona-ordinador fets servir en, per exemple, estudis de usabilitat, d'experiència d'usuari o de la millora de tècniques de transmissió de vídeo. El comportament dels usuaris en aquests entorns ha atret l'atenció dels investigadors, però s'ha parat poca atenció a comparar el comportament dels usuaris entre diferents dispositius per reproduir entorns virtuals o vídeos en 360 graus. Nosaltres introduïm un sistema interactiu que hem fet servir per crear i reproduir entorns de realitat virtual i experiències basades en vídeos en 360 graus, que és capaç de recollir automàticament el comportament dels usuaris, de manera que el puguem analitzar. Hem estudiat el comportament recollit en la reproducció d'un entorn de realitat virtual amb aquest sistema i hem trobat diferències significatives en l'execució entre usuaris d'una interfície basada en Oculus Rift i d'una altra basada en un visor de RV mòbil semblant a la Google Cardboard: diferent temps entre interaccions, probablement causat per la necessitat de fer un gest amb la primera interfície; diferències en l'exploració espacial, perquè els usuaris de la primera interfície van triar romandre en una àrea de l'entorn; i diferències en l'orientació dels seus caps, ja que els usuaris d'Oculus tendiren a mirar cap a objectes físics de la instal·lació de l'experiment i els usuaris dels visors mòbils semblen influïts pels valors d'orientació inicials dels seus navegadors. Un segon estudi va ser executat amb les dades recollides amb aquest sistema, que va ser fet servir per reproduir un hipervídeo fet de vídeos en 360 graus, en què hem comparat el comportament dels usuaris entre quatre interfícies (dues basades en dispositius immersius i dues basades en dispositius no immersius) i dues categories de vídeos: hem trobat diferències significatives en l'exploració de l'espaitemps del vídeo, en la dispersió de l'orientació dels usuaris, en el moviment d'aquestes orientacions i en l'agrupació de les seves trajectòries, especialment entre diferents tipus de vídeo però també entre dispositius, ja que hem trobat que, en alguns casos, el comportament amb dispositius immersius és similar a causa de límits semblants en la interfície, que no són presents en dispositius no immersius, com amb un ratolí d'ordinador o la pantalla tàctil d'un mòbil. Finalment, hem reportat un model basat en una xarxa neuronal recurrent, que és capaç de classificar aquestes reproduccions de vídeos en 360 graus en els seus corresponents tipus de vídeo i interfície que s'ha fet servir amb una precisió de més del 90% amb només quatre segons de trajectòria d'orientacions; un altre model d'aprenentatge profund ha estat implementat per predir orientacions fins a dos segons en el futur a partir dels darrers segons d'orientació, amb uns resultats que han estat millorats fins a un 19% per un model comparable que aprofita el tipus de vídeo i el dispositiu que s'ha fet servir per reproduir-lo.

Resumen

La realidad virtual y las tecnologías que están relacionadas con ella se usan para muchos tipos de contenidos, como entornos virtuales o vídeos en 360 grados. Contenidos multimedia omnidireccionales e interactivos son consumidos con diversos dispositivos, como ordenadores, dispositivos móviles o aparatos especializados de realidad virtual. Los estudios del comportamiento de los usuarios con interfaces de ordenador son una parte importante de la investigación en la interacción persona-ordenador usados en, por ejemplo, estudios de usabilidad, de experiencia de usuario o de la mejora de técnicas de transmisión de vídeo. El comportamiento de los usuarios en estos entornos ha atraído la atención de los investigadores, pero se ha dedicado poca atención en comparar el comportamiento de los usuarios entre diferentes dispositivos para reproducir entornos virtuales o vídeos en 360 grados. Nosotros introducimos un sistema interactivo que hemos usado para crear y reproducir entornos de realidad virtual y experiencias basadas en vídeos de 360 grados, que es capaz de recoger automáticamente el comportamiento de los usuarios, de manera que lo podamos analizar. Hemos estudiado el comportamiento recogido en la reproducción de un entorno de realidad virtual con este sistema y hemos encontrado diferencias significativas en la ejecución entre usuarios de una interficie basada en Oculus Rift y otra basada en un visor de RV móvil parecido a la Google Cardboard: diferente tiempo entre interacciones, probablemente causado por la necesidad de hacer un gesto con la primera interfaz; diferencias en la exploración espacial, porque los usuarios de la primera interfaz permanecieron en un área del entorno; y diferencias en la orientación de sus cabezas, ya que los usuarios de Oculus tendieron a mirar hacia objetos físicos en la instalación del experimento y los usuarios de los visores móviles parecieron influidos por los valores iniciales de orientación de sus navegadores. Un segundo estudio fue ejecutado con los datos recogidos con este sistema, que fue usado para reproducir un hipervídeo compuesto de vídeos en 360 grados, en el que hemos comparado el comportamiento de los usuarios entre cuatro interfaces (dos basadas en dispositivos inmersivos y dos basadas en dispositivos no inmersivos) y dos categorías de vídeos: hemos encontrado diferencias significativas en la exploración espaciotemporal del vídeo, en la dispersión de la orientación de los usuarios, en el movimiento de estas orientaciones y en la agrupación de sus trayectorias, especialmente entre diferentes tipos de vídeo pero también entre dispositivos, ya que hemos encontrado que, en algunos casos, el comportamiento con dispositivos inmersivos es similar a causa de límites parecidos en la interfaz, que no están presentes en dispositivos no inmersivos, como con un ratón de ordenador o la pantalla táctil de un móvil. Finalmente, hemos reportado un modelo basado en una red neuronal recurrente, que es capaz de clasificar estas reproducciones de vídeos en 360 grados en sus correspondientes tipos de vídeo y la interfaz que se ha usado con una precisión de más del 90% con sólo cuatro segundos de trayectoria de orientación; otro modelo de aprendizaje profundo ha sido implementad para predecir orientaciones hasta dos segundos en el futuro a partir de los últimos segundos de orientación, con unos resultados que han sido mejorados hasta un 19% por un modelo comparable que aprovecha el tipo de vídeo y el dispositivo que se ha usado para reproducirlo.

Agraïments

Amb aquesta tesi he assolit una fita important, tant en l'àmbit acadèmic com personal. He arribat fins aquí gràcies a les aportacions de moltes persones i els voldria donar les gràcies amb aquestes línies.

Gràcies als meus companys del Laboratori de Tecnologies de la Informació Multimèdia. En concret, vull agrair a en Toni Bibiloni el seu suport des de fa més de deu anys, a través de múltiples etapes de la meva formació acadèmica i ara com a director d'aquesta tesi. No puc passar sense esmentar el meu company de recerca en Javier del Molino.

A més, gràcies als professors del Departament de Ciències Matemàtiques i Informàtica de la Universitat de les Illes Balears. M'he sentit ben acollit al departament i he rebut diverses mostres de suport de part seva. Vull reconèixer especialment l'ajuda d'en Pere Palmer i en Pedro Bibiloni. També expressar el meu agraïment al grup de recerca Distributed and Interactive Systems del Centrum Wiskunde & Informatica per la seva acollida.

Per acabar, vull donar les gràcies d'una manera especial a la família per haver-me donat suport durant tot aquest camí i haver-me ajudat quan més ho he necessitat.

Scientific contributions

The majority of the results presented in this thesis have either already been published in scientific literature or are in the publication process. Especially, the following journal articles are based on main contributions of this thesis:

- Javier del Molino, Toni Bibiloni, and Antoni Oliver. Keys for successful 360° hypervideo design: A user study based on an xAPI analytics dashboard. 910122 - Multimedia Tools and Applications. 79, pp. 22771 - 22796. (Netherlands): 2020. ISSN 1380-7501. DOI: https://doi.org/10.1007/s11042-020-09059-2.
- Toni Bibiloni, Antoni Oliver, and Javier del Molino. Automatic collection of user behavior in 360° multimedia. 910122 - Multimedia Tools and Applications. 77, pp. 20597 - 20614. (Netherlands): 2018. ISSN 1380-7501. DOI: https://doi.org/10.1007/s11042-017-5510-3.

Additionally, a number of articles have been published in conference proceedings or as book chapters:

- Antoni Oliver, Pere Palmer, Javier del Molino, and Antoni Bibiloni. 2022. Classification of the Video Type and Device Used in 360-Degree Videos from the Trajectories of its Viewers' Orientations with LSTM Neural Network Models. In ACM International Conference on Interactive Media Experiences (IMX '22). Association for Computing Machinery, New York, NY, USA, (pending to be published in print). https://doi.org/10.1145/3505284.3532975.
- Antoni Oliver, Javier del Molino, and Antoni Bibiloni Coll. 2020. Palma360: A 360-degree Web-based Multi-device Hypervideo. In ACM International Conference on Interactive Media Experiences (IMX '20). Association for Computing Machinery, New York, NY, USA, 165–169. https://doi.org/10.1145/3391614.3399394.
- Antoni Oliver, Javier del Molino, Maria Cañellas, Albert Clar, and Antoni Bibiloni. VR Macintosh Museum: Case Study of a WebVR Application. New Knowledge in Information Systems and Technologies. 931, pp. 275 - 284. (Spain): Springer, Cham, 2019. ISBN 978-3-030-16183-5. DOI: https://doi.org/10.1007/978-3-030-16184-2_27.
- Antoni Oliver, Javier del Molino, and Antoni Bibiloni. Automatic View Tracking in 360° Multimedia Using xAPI. Applications and Usability of Interactive Television. jAUTI 2017. 813, pp. 117 - 131. (Switzerland): Springer, Cham, 2018. ISBN 978-3-319-90170-1. DOI: https://doi.org/10.1007/978-3-319-90170-1_9.
- Antoni Oliver, Javier del Molino, Manuel Elviro Vidal, and Toni Bibiloni. 360° Hypervideo: An interactive documentary around the refugee crisis in Greece. Proceedings of the 6th Iberoamerican Conference on Applications and Usability of Interactive TV - jAUTI 2017. 1 - 1, pp. 162 - 171. Aveiro (Portugal): Universidade de Aveiro, 2017. ISBN 978-972-789-521-2.

The candidate has benefitted from the fellowship *Contractes Predoctorals UIB 2016-2017* during the first year of his research project and from an Assistant position in the Mathematics and Computer Sciences Department of the University of the Balearic Islands in the successive years.

A la meva família, especialment a na Carme pel seu suport constant.

Contents

Index of figures

Index of tables

Chapter 1. Introduction

Virtual Reality (VR) and related technologies are nowadays being used by the general public for many kinds of content, like virtual environments, augmented reality applications or 360-degree videos, which are part of the Extended Reality (XR) model. There are many applications of these technologies, although we will focus on two: interactive 360-degree videos and a virtual museum.

Omnidirectional, interactive, multimedia is consumed with a variety of devices: non-immersive devices such as computers or mobile devices, or immersive devices like specialized VR gear. This last group has seen a recent surge in popularity thanks to the rapid development of VR headsets intended for the general public and to the invention of gadgets designed to enhance the screen of a smartphone, becoming a mobile-based VR visor.

Studies on user behavior with computer interfaces are an important part of the research in the field of human-computer interaction. As users find additional difficulties when they navigate omnidirectional environments [1], their behavior in these environments has drawn the attention of researchers in the field. Nevertheless, the vast majority of these studies is focused in one of these devices at a time, and little attention has been paid to compare the users' behavior in multiple interfaces based on different devices to reproduce virtual environments or 360-degree videos. We are confident that the differences in the interface with various devices have an impact in their users' behavior which in turn, has an effect in their experience and usability, which must be considered when designing this kind of virtual experiences.

The analysis of the users' behavior on this novel multimedia format is also important to design more efficient streaming techniques for 360-degree videos, as they require significantly higher quotas of bandwidth than regular videos. A line of research are adaptive streaming techniques for 360-degree videos, that leverage a prediction of the future orientation of the user. Current work considers only one of these devices at a time, so the effect of incorporating data of multiple devices into the model can improve the results of such prediction techniques, as long as differences in behavior are found between users of different devices.

1.1. Objectives

The motivation of this thesis is to collect, study and use the users' behavior with different devices in virtual reality environments and 360-degree videos. The following objectives are set for this thesis:

Objective 1. *Collect the users' behavior in a virtual reality environment and 360-degree videos with multiple devices.*

To be able to study such behavior, it first has to be generated on an interactive multimedia production which is to be played with the devices that are to be tested. This behavior then needs to be collected so it can be studied.

To play the multimedia productions and to automatically collect the users' behavior, a system is to be developed, with an interoperable specification to store and retrieve the data that is collected.

A number of different kinds of actions and data in which this behavior is based are to be collected, such as the selection of virtual items, the users' position in the virtual environment or the orientation of their heads, etc. Multiple contextual variables should also be stored, such as the device that was used in the computer interface, the element that is receiving the interaction, the kind of video that is being played, etc.

The development and validation of such a system is the first milestone of this thesis.

Objective 2. *Analyze the users' behavior in virtual reality environments and 360-degree videos to identify differences between the multiple interfaces.*

Once the behavior on virtual reality environments and 360-degree videos with multiple devices has been collected, we are interested in finding differences in them according to the value of the contextual variables that we recorded.

A comparison of the users' behavior between devices would already extend the current state of the art, as studies that take under consideration multiple interfaces in 360-degree videos or virtual environments are very scarce. Nevertheless, differences in behavior are to be searched for between users of different devices in both cases and between video types in the case of 360-degree videos.

Objective 3. *Propose an improvement of a prediction model for the users' future orientations in 360-degree videos played with multiple devices which considers the differences between them.*

Finally, an important application of a specific kind of user behavior in 360-degree videos revolves around the prediction of the future orientation of the users when they play these videos, to improve the streaming of these multimedia contents over the Internet. While this has been already studied, different devices are not considered together in the same experiment.

We are to study if an improvement of the prediction made by a model for the users' future orientations can be achieved by leveraging the contextual information we collected before, especially, the device which was used to play the 360-degree video but also the type of video that was being played.

A model to recreate this contextual information will also be developed and evaluated with the collected data, so as to be able to improve these prediction models even when the contextual data is not collected.

1.2. Contributions

This thesis is structured in several chapters, whose body is devoted to the contributions made to the objectives we introduced.

We first provide a background overview of the current state of the art on virtual reality and related technologies, 360-degree videos and hypervideos, adaptive streaming and the collection of users' behavior in Chapter 2. We report the use of multiple devices to play virtual reality environments and 360-degree videos, the efforts to collect and analyze users' behavior in these multimedia applications and how that behavior can be used to develop models to predict future orientations of their users.

We report the creation of four different multimedia productions, three based on 360-degree videos and one in a virtual reality environment in Chapter 3. We introduce a system to aid in their creation, reproduction, and in the collection of the users' behavior when they were played. The development of this system follows an evolution through these four experiments. This contribution pursues Objective 1 by demonstrating the use of such a system to collect that kind of user behavior.

In the following two chapters, we work towards Objective 2 by processing and analyzing the data we collected with the system we described on two of the productions we reported in the previous chapter.

In Chapter 4, we study the users' behavior in a virtual reality environment with two different devices, a standalone VR headset and a mobile-based VR headset. We analyze and discuss the differences we found in their behavior. This analysis contributes to Objective 2, as differences between interfaces, although both immersive, are noted, so they need to be taken under consideration if a similar user experience is to be perceived by the two groups of users in environments of virtual reality.

In Chapter 5, we analyze the users' behavior in 360-degree videos with four interfaces: two based on non-immersive devices, a mouse-based computer interface and a drag-based mobile interface; and two based on immersive devices, a standalone VR headset and a mobile-based VR headset. In this case, we observe the differences in exploration, orientation, movement and trajectory between video types and

devices. This study also contributes to Objective 2 by showing how the users' actions are dissimilar between not only the kind of video that is played but also when they are played with different devices.

Our final contributions are made in Chapter 6, as we explore deep learning models based on the data we collected. We first introduce a deep learning model to classify that behavior in 360-degree videos into the contextual variables we collected (type of video that was being played and device which was used), with an accuracy above 90% for both variables with four seconds of trajectory. We then then improve a deep learning model to predict the users' future viewports leveraging that contextual data we collected (or that we can accurately reproduce, if it were unavailable) by up to 19% to a comparable deep learning model that does not use that information. These two contributions work towards Objective 3 by showing that the prediction models that are used in the state of the art nowadays are not using contextual information that could improve them significantly.

We conclude this thesis in Chapter 7 with an overall reflection about the consecution of these objectives and some thoughts of future work which could take advantage of the contributions made in this project.

Chapter 2. State of the art

In this chapter we provide an overview of the literature in Extended Reality, immersive devices, immersive technologies on the web, 360-degree videos and hypervideos, adaptive streaming and viewport prediction, and the Experience API.

2.1. Extended Reality

In recent years, there have been many advancements in Human-Computer Interaction (HCI). Terms like Virtual Reality, Augmented Reality or Mixed Reality have become popular nowadays.

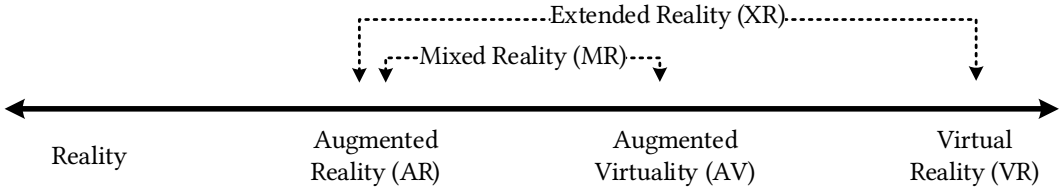

Figure 2.1. The reality-virtuality continuum, adapted with the modern Mixed Reality and Extended Reality concepts.

Paul Milgram introduced the reality-virtuality continuum in 1994 [2], see Figure 2.1, which describes the continuous transition between environments composed only by elements in the real world to environments consisting solely of virtual elements (Virtual Reality). In this range, the terms Augmented Reality, and Augmented Virtuality appear.

Virtual reality (VR) is defined as "a medium composed of interactive computer simulations that sense the participant's position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation (a virtual world)" [3]. According to them, the following are the key elements in any VR experience: the virtual world, the immersion, the sensory feedback, and the interactivity. As understood by Mel Slater [4], the sense of presence contributes to the realistic response required in this immersive medium.

Augmented virtuality (AV), a term not so popular, is a medium in which real elements (e.g. a picture) are included in the virtual environment [5].

Augmented reality (AR), conversely, is a medium in which virtual elements are included in the real environment [5]. According to Sherman and Craig [3], it is "a medium in which real-time interactive digital information is overlaid on the physical world that is in both spatial and temporal registration with the physical world".

Mixed reality (MR) was first introduced by Milgram [2] to include the range between AR and AV, both included, without including neither reality or VR [5].

Distinctions between these terms are blurry and are becoming less significant as the technologies that support them evolve, as they are often displayed by similar devices [6], which are sometimes globally named "VR". The new umbrella term "Extended Reality" (XR) was introduced to unite these terms under a global name.

Museums in virtual reality have already been explored. A system that allows museums to be built and manage VR and AR exhibitions is presented in [7]. A VR museum based on gestures to obtain additional information is described in [8], a consumer technologies museum implemented in JanusVR [9], a system to create and experience VR exhibitions using Unity3D [10] and even a location-based VR museum that uses Samsung Gear VR [11]. Other projects use virtual reality inside the museum themselves [12]–[14]. In [15], a mobile based VR experience in a museum is reported, which uses gaze

as the interaction technique. Shehade *et al.* [16] describe the experiences of museum professionals about VR, which reported a lack of engaging social experiences. The description and analysis of an archaeological exhibition is supplemented with a virtual reality experience for informal learning in [17], implemented in Unity3D and played with Samsung Gear VR glasses and an HTC VIVE HMD; they registered the users' actions and found that efficiency in performing gestures and amount of movement is correlated with the confidence of the player with the system and that most novice users did not leave the initial area.

The present thesis studies immersive environments based on virtual reality: a virtual museum and 360-degree hypervideos. They fully consist of virtual elements in a scene in which the user is immersed into. While it is true that most of these virtual images are obtained from real elements (pictures, videos), real elements are never blended in the scene in real time.

2.2. Immersive devices

Virtual Reality experiences require specialized gear to immerse the user in the virtual world [18]. These devices are common throughout Extended Reality experiences, in various degrees, contributing to the aforementioned confusion. There are two big families of devices that provide this immersion: Spatially Immersive Displays (SID) and Head-Mounted Displays (HMD).

The first group, SID, surround the viewer with a panorama of the virtual environment through a video projection on a room-like device, in which the user walks in. The first application of SID to VR is the CAVE [19]. These devices offer a wide field of view without the need of head tracking devices and the possibility of a group experience between various users but are very expensive and require a large amount of space.

The second group, HMD, also referred to as VR headsets or Head-Based Displays, have been a key part of VR since their introduction. These devices are worn over the eyes of the user and render the virtual environment according to the various sensors which they contain, at least a sensor that captures the rotation of the user's head.

Recently, the introduction of the Oculus Rift[1] [20] has popularized VR headsets [21] which, in turn, has resulted in various alternatives and competitors such as the HTC Vive[2] or the PlayStation VR[3]. As smartphones are also very popular nowadays, mobile VR headsets were also introduced, in which a smartphone is to be placed inside the visor, like the Google Daydream[4], the Samsung Gear VR[5] or even the low-cost Google Cardboard[6] and variations of it [22].

The sensors and actuators available in these devices constrain the interface with the virtual world: at least three degrees of freedom are required for immersivity, which correspond to the rotation angles of the user's head: yaw, pitch, roll. More advanced headsets feature six degrees of freedom, accounting for those three and the spatial position of the user, which are often missing in mobile VR headsets. The availability of controllers also defines the actions that can be performed with the interface: most VR headsets nowadays include controllers in one way or another, which can be used to perform various gestures [23], with the notable exception of the Google Cardboard [24].

Immersive devices are extensively used in the present thesis, as the study of the differences in the users' behavior between them and non-immersive devices is one of the objectives of this thesis.

[1] https://www.oculus.com/rift/
[2] https://www.vive.com/us/
[3] https://www.playstation.com/ps-vr
[4] https://arvr.google.com/daydream
[5] https://www.samsung.com/galaxy/gear-vr/
[6] https://arvr.google.com/cardboard/

2.3. Immersive technologies on the web

The delivery of virtual environments through the Internet has been supported by open Web3D standards like VRML [25] and X3D [26]. An early study of these technologies as a learning tool is given in [27].

WebXR [28] (previously WebVR [29]) is an HTML5 specification that enables browsers to use VR devices, including head-mounted displays, sensors and controllers on the Web, so immersive applications can be played with this kind of VR hardware, which helped popularize and distribute VR applications through the Internet even more.

WebGL [30] is an HTML5 specification that provides a low-level 3D graphics API to web pages. This specification is often used through the immensely popular 3D library three.js[7].

Multiple libraries and frameworks wrap around WebXR or WebVR and the either WebGL directly or indirectly through three.js, like A-Frame[8], React 360[9] or BabylonJS[10].

WebXR is being used in the literature for 360-degree videos [31], virtual reality [32], social experiences in VR [33] and interactive visualization of complex data [34], to name a few examples.

The multimedia system with which the users' behavior will be collected, based on a web application, will use these technologies to display an immersive environment and use the immersive devices we previously discussed.

2.4. 360-degree video

360-degree videos [35] are a multimedia format that display a video that was recorded in all the directions of the space. The name for this kind of video is a reference of the 360 degrees that are available from left to right. Other names with which this format is often referred to are spherical videos, omnidirectional videos, panoramic videos, immersive videos, VR videos, etc. Some of these names are based on the availability of the video content: as a sphere, in all directions, as a panorama, while others are based on the interaction which is often used to reproduce them, such as immersive interfaces or VR devices.

These kinds of videos contain recorded images through special equipment, such as ad-hoc camera rigs [35], where each camera records a video in one direction and they are later joined together in a process called *stitching* [36], or with special cameras that are already equipped with multiple lenses and can perform this stitching process while the video is being recorded. According to the geometric projection used in this process, the final result will differ: a sphere is often used, but a cylinder can be used when image data in the top and bottom is not available, a cube can also be used, and several other more complex geometries, like the pyramid-based geometry proposed in [37], an example of a viewport-dependent projection. The image data is then represented in a plane with a 2d projection, like the equirectangular projection, the cubemap projection, etc.

There are many techniques to reproduce 360-degree videos, which ultimately provide the user with the choice to control what part of the video they currently see [38]. Some of these interfaces are based on non-immersive devices, such as a drag interface [39] in which the video viewport shown in a computer screen is rotated by dragging a mouse, but other interfaces are based on immersive devices, also used for virtual reality applications.

The fact that they are called VR videos and that they are often played with immersive devices which are also used to play virtual reality interfaces can lead to the assumption that spherical videos are a virtual reality experience [3], but there are some key differences and limitations between 360-degree videos and virtual reality experiences, such as the lack of perspective in occluding objects [40]. The introduction of the concept of Extended Reality (XR) should ease this confusion.

[7] https://threejs.org/
[8] https://aframe.io/
[9] https://facebook.github.io/react-360/
[10] https://www.babylonjs.com/

The collection of user behavior in 360-degree videos consists of recording the users' head or head and eyes positions to determine the consumed viewports of the omnidirectional videos. There exist commercial solutions like Wistia [11] or LiveHEAT [12], which display the viewed portions of the omnidirectional videos with heatmaps. Similar techniques are used in [41] to evaluate the behavior around an spherical display and in [42] for a telepresence omnidirectional video. 360-degree images viewed through HMD are analyzed in [43]–[45].

Head motion trajectories in omnidirectional video using an HMD are extracted in [46]. A dataset of head movements when playing 360-degree videos using an HMD is reported in [47], showing that, in contrast to videos recorded with a static camera, in videos in which the camera follows a path, viewers follow a well-defined region of interest. A head and eye trajectory dataset is reported in [48], establishing a relationship between camera movement and Temporal Perceptual Information. Head and eye trajectories are used in [49] to assist a convolutional neural network to detect saliency in 360-degree videos. In [50], reproductions of 360-degree videos are continuously annotated with the valence and arousal experienced by the viewer.

Different interfaces are used to play 360-degree videos in different studies: drag interfaces for mouse-based computers, VR headsets and mobile VR headsets, but the literature has paid little to no attention to the comparison of the users' behavior between these devices.

360-degree videos are a core part of this thesis: the users' behavior when playing them with different devices will be collected, analyzed, and used to improve this kind of content.

2.5. Hypervideo

A hypervideo [51] is a navigable stream of video that offers the viewer the possibility of choosing a path in the narrative combining several multimedia tracks through spatiotemporal hyperlinks present in the media.

360-degree videos can be used in hypervideos, resulting in 360-degree hypervideos [39]. This change introduces challenges in the interaction with the hyperlinks [52] and the annotation required to create them [53]. A web-based 360-degree hypervideo used to navigate a tour through the city is presented in [54].

A number of the virtual environments produced and analyzed in this thesis are 360-degree hypervideos, where the user is able to select hyperlinks to navigate the content or perform other actions.

2.6. Adaptive streaming

Delivery of video through the Internet has moved from Real-Time Transport Protocol (RTP) to streaming over the HTTP protocol. As audiovisual quality and bit rates increase and multiple devices with diverse capabilities are able to reproduce these contents, there was a need to provide a system to stream different representations of the same multimedia content, which is why MPEG Dynamic Adaptive Streaming over HTTP was developed [55]. This protocol is widely used by major video streaming sites, such as YouTube [56] or Netflix [57].

360-degree videos require a much higher bit rate than conventional videos. This is due to the fact that a very large part of the 360-degree space, that is not viewed by the user, is streamed with the same quality than the part that is actually seen.

Current research strives in identifying the viewport that is going to be displayed so that bandwidth can be saved, or additional bit rate can be assigned to the areas that are actually displayed if the areas that are not likely to be displayed are assigned a lower bit rate budget. This results in a form of viewport-adaptive streaming [58], which is typically based on dividing the 360-degree space in regions known as *tiles*, resulting in tile-based viewport-adaptive streaming techniques.

[11] https://wistia.com/
[12] http://www.finwe.mobi/main/liveheat/

A system based on MPEG-DASH SRD (Spatial Relationship Description [59]) is reported in [60], [61], which could save up to 72% of the required bandwidth [62]. A system based on regions with better quality than the rest is presented in [63], with the goal to provide the best quality for a given bit rate budget. In [64], a potentially non-uniform tiling scheme is explored, with savings of up to 73% of the required bandwidth. A model based on High Efficiency Video Coding (HEVC) [65] is proposed in [66]. A comparison between the tile-based technique and a viewport-dependent projection based on a truncated pyramid geometry is reported in [67]. A system based on HTTP/2 is studied in [68]. A cellular-friendly streaming scheme is proposed in [69].

Adaptive streaming is used in a number of the 360-degree video experiments reported in this thesis. Moreover, schemes specially designed for 360-degree videos often require an accurate prediction of the users' future orientation, which is discussed next.

2.7. Prediction of future orientations

One of the most important aspects of viewport-adaptive streaming techniques lies on the correct prediction of the future viewports that will be displayed when a video is streamed. The literature makes the distinction of Field of View prediction techniques as content-based and trajectory-based. In the first category, the audiovisual content itself is used to predict the future viewport of a user (such as [37]), while in the second, this prediction is performed with the trajectory of the past viewports that the viewer has been watching. Techniques that combine both features have also been studied.

Linear regression and a 3-layer perceptron based on past orientations were used in [70]. A neural network that leverages past fixations and video features is introduced in [71] to predict a future window of two seconds, using Scikit-learn[13], Keras[14] and the Long Short-Term Memory neural network architecture (LSTM) [72]–[74]. Saliency is used to predict head movement in 360-degree videos in [75]. Deep reinforcement learning is used in [76] to predict potential orientation values from past orientations both offline and online, based on LSTM. [77] predicts future viewports with the aid of other users' actions based on Linear Regression and k-near neighbors. In [78], other users' viewports are clustered and a Support Vector Machine is used to fit the current user to a cluster. A probabilistic approach to improve Quality of Experience is presented in [79], with possible quality changes every second.

The last objective in this thesis is based on improving the results of state-of-the-art prediction techniques by leveraging the knowledge of the distinct devices used to play the 360-degree videos.

2.8. Experience API

The Experience API (xAPI) [80] is an open specification designed to collect data about the wide range of experiences a person has, which allows different systems to communicate and share the data experienced by the users of an application. xAPI is a Representational State Transfer (RESTful) API in which a statement of experience is tracked as a learning record. Although the term *learning* appears throughout the elements of xAPI, any experience may be tracked, not only those related to e-learning.

The xAPI registers the experience in statements. These statements are simple constructs, represented in JavaScript Object Notation (JSON), which contain, at least, an actor, a verb, and an object [81]. Optional properties, such as a timestamp, a context or a result can also be specified in a statement, as well as some properties may be extended, to better reflect in the dataset the nature of the experience. Some of these parts, like the verb or the activity type are identified by an Internationalized Resource Identifier (IRI). The Registry [82] contains several of these identifiers which are free to use by any application that uses xAPI, although custom identifiers can be created if needed.

Three agents are identified in the xAPI model: the Learning Record Store, the Learning Record Provider, and the Learning Record Consumer. A Learning Record Provider (LRP) is any application that delivers data to an LRS. The Learning Record Store (LRS) is the server responsible for receiving,

[13] https://scikit-learn.org/
[14] https://keras.io/

securely storing, and providing access to xAPI statements, which can be standalone, such as lxHive[15] or part of Virtual Learning Environments (VLE). Finally, a Learning Record Consumer (LRC) is any application that accesses the data within the LRS and processes it.

xAPI has been used in many studies, mainly related to learning technologies, such as those included in [83]–[86]. However, due to its generic nature, xAPI may be used in the most diverse areas.

Virtual reality is used in learning applications, often with xAPI, such as a game-based learning using virtual reality, which collected students' learning process with xAPI [87]; augmented reality has been used in learning and teaching, and xAPI can be used to exchange information about learning experiences [88]; virtual blended learning scenarios are used to integrate 3D worlds into University lectures, also tracking the learners' progress with xAPI [89].

Finally, the Experience API is used throughout the present thesis to store and query the users' actions in the immersive systems we developed, so it can be later analyzed. Concrete examples of the use of this technology are provided in the following chapter.

[15] https://github.com/g3i/lxHive

Chapter 3. Automatic collection of user behavior in VR multimedia

In this chapter, we introduce the design, implementation, and initial results of the evolution of a system to play interactive immersive multimedia while it automatically registers the users' behavior, which has been extended to work with various multimedia projects (see Figure 3.1).

The users' behavior collected this way can be used to perform various analyses on the usability and user experience with virtual reality and 360-degree videos, like cybersickness [90], immersion [91] and presence [92], but also to know how the users interact with the system, what they prefer using and which areas they explore at a given time, as we will show in the following sections.

These projects were used to introduce new capabilities to the system incrementally, as will be reported in the following subsections.

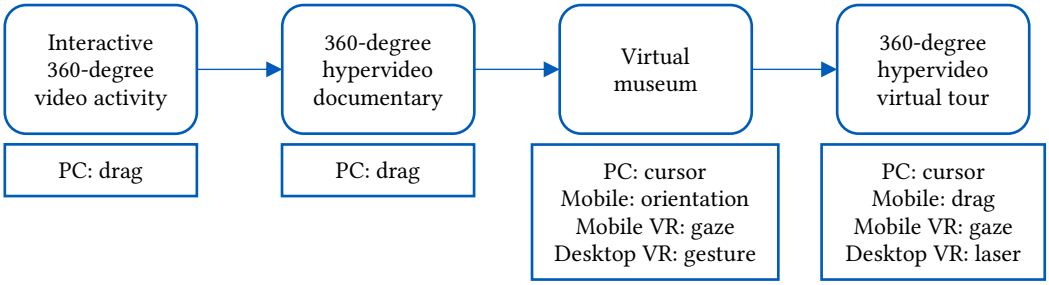

Figure 3.1. Evolution of the system to play omnidirectional multimedia and register the users' behavior, and the user interfaces which have been supported.

In general, the overall design of the system has remained the same (Figure 3.2): first, an interactive 360-degree experience is designed with a creation module; then, that experience is played in a web application that tracks the user behavior and submits that information to a behavior database; finally, the user data is queried by an analysis tool which produces insights about the users' actions.

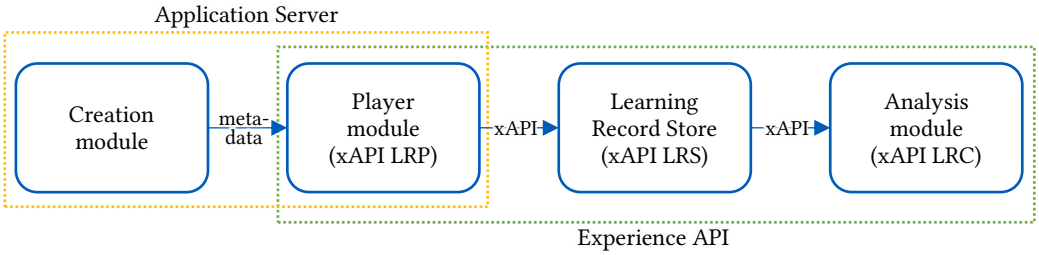

Figure 3.2. General architecture of the omnidirectional multimedia system.

The first iteration of the system was created for educational interactive activities based on 360-degree videos, which were played in a drag interface using computers. Users were asked to perform a selection-based task on the approximate location in the 360-degree space of the video in which specific elements appeared. An initial analysis of the spatiotemporal consumption of the video and distribution of successful and unsuccessful selections was performed.

In a second iteration, a 360-degree hypervideo documentary was created, which displayed markers in the 360-degree space of the video that displayed additional multimedia content. A drag-based interface using computers was used to play the production. The users' data was helpful to answer questions about 360-degree hypervideo consumption.

We diverged from 360-degree videos to full virtual reality in the third iteration to demonstrate that our system was capable of supporting them: a virtual museum of vintage Macintosh computers was designed to be played with four different user interfaces: a cursor-based with positional keyboard movement interface for computers, a drag-based interface for smartphones, a gaze-based interface for mobile VR visors and a gesture-based with joystick movement for desktop-based VR HMD. A study comparing the interaction in the virtual reality experience using mobile VR headsets and a desktop-based VR HMD was performed with the collected data.

Finally, the fourth iteration consists of a 360-degree hypervideo that links 20 omnidirectional videos together in a virtual tour, which was played with four user interfaces: a cursor-based interface for computers, a drag-based interface for smartphones, a gaze-based interface for mobile VR visors and an interface based on a laser pointer for desktop-based VR HMD. A study comparing the interaction in the 360-degree videos using the four interfaces was performed with the collected data.

In the following subsections, we will explain specific design and implementation details of this platform, in each of the four versions, leaving the complete analysis of the collected data for posterior chapters in this document.

3.1. Immersive multimedia in the web

The inclusion of the WebGL [30] API in web browsers has permitted the rendering of 3D graphics which are essential in the presentation of immersive multimedia such as virtual reality or 360-degree videos, which were, until recently, exclusive to native applications. WebGL is often leveraged by graphics libraries such as the extremely popular three.js.

Figure 3.3. Still frame from a 360-degree video in the equirectangular projection in 16:9 aspect ratio.

To produce a 360-degree video, visual data in all the directions needs to be recorded. Usually, either multiple cameras or cameras with several lenses are employed and the individual videos have to be joined together in a process called *stitching*, which can be done manually or automatically in some modern cameras.

360-degree videos are available in a variety of formats, but the equirectangular and cubemap projections are the most common techniques to represent a spherical video in a standard video file. In this chapter we will always use videos in the equirectangular projection (Figure 3.3).

These video files can be then loaded into a player that renders them as textures of a sphere (or a cube if they follow a cubemap projection or other geometries in other cases). The 3D camera is placed inside the geometry (see Figure 3.4), so the user can only see part of the spherical video, the viewport, with

its center defined by the current rotation of the camera. There are multiple ways to select a viewport, which range from dragging a cursor with a pointing device (like a mouse) or touchscreen to using orientation sensors in virtual reality headsets. In this thesis, we will represent the center of these viewports as two angles in degrees: latitude and longitude. The latitude represents the vertical rotation (pitch), and the longitude represents the horizontal rotation (yaw).

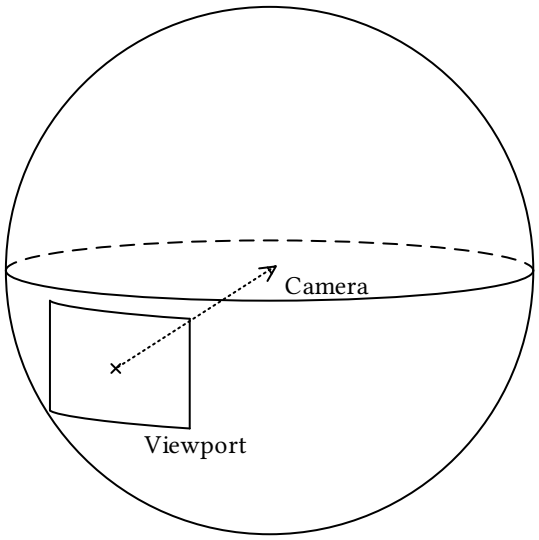

Figure 3.4. Representation of the viewport in the display of a 360-degree video.

Once 3D graphics are available, this kind of multimedia can be displayed in non-immersive devices, such as computers or mobile devices. But to be able to make use of truly immersive devices, such as Virtual Reality headsets, support for these devices needed to land into web browsers.

The WebVR [29] API, now WebXR [28] API, introduced support for these devices, effectively making possible to create Virtual Reality experiences in web pages, displaying them in immersive gear with orientation sensors, positional tracking, and gesture-packed controllers, such as the Oculus Rift. Support for mobile Virtual Reality was also introduced, which are gadgets designed for smartphones to be inserted into them, so the user is also immersed into the experience leveraging the built-in sensors in most smartphones, like the Google Cardboard.

Finally, a 360-degree video can also be displayed with these immersive devices. In this case, it is a kind of a virtual reality experience, in the sense that the user is immersed into the experience, but it still is the reproduction of a spherical video. In any case, what we exposed before still applies: a viewport must be selected and is rendered in the device.

3.2. The Experience API

The general architecture of the system (Figure 3.2) follows that of the xAPI, with modules that act as the Learning Record Provider, Store and Consumer. In all the cases, an external Learning Record Store (LRS) has been used, while a specific web-based development has been made for the player module, which acts as the Learning Record Provider (LRP) and for the Analysis module, which acts as the Learning Record Consumer (LRC).

The communication between xAPI modules (between the LRP and the LRS and the LRS and the LRC) is done via statements. These statements are a structure which represent the experience that has been performed and contain, at least, an actor, which represents the user that performed the experience; a verb, which represents which action was performed; and an object, which represents on which activity the experience happened. Additional information can be provided in the form of contextual and results extensions, which are key to report specific details about these interactions.

We can see an example of one of these statements used to report a change in the current orientation in a 360-degree video in Figure 3.5: in this case, the *actor* is identified with a local *account*, which is identified with a v4 UUID; the *verb* is an URI which represents the verb “interacted”, which is used to identify the action performed (other verbs are used for other actions, such as to report that a video

has started playing, has been paused, etc.); the *object* contains which kind of activity the experience was performed on (in this case, a video) and what specific video was it performed on (the URI detailed in the *id* property); then a statement can have *context* and *result* properties, which are used according to the action which was performed, in this case, the *context* property has a set of *extensions* (identified by their URI), which detail a *session-id*, which links this statement to the others of this specific reproduction of the 360-degree video; a *latitude* and *longitude*, which specify the current orientation in degrees in the 360-degree video; and the current *time* of the video, in seconds; finally, a *timestamp* that details the specific moment in time in which the experience was performed is included in the statement.


```
{
  "actor": {
    "objectName": "Agent",
    "account": {
      "homePage": "https://ltim.uib.es/360/",
      "name": "75442486-0878-440c-9db1-a7006c25a39f"
    }
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/interacted"
  },
  "object": {
    "id": "https://ltim.uib.es/files/palma360/p1/dash.mpd",
    "definition": {
      "type": "https://w3id.org/xapi/video/activity-type/video",
    },
    "objectType": "Activity"
  },
  "context": {
    "extensions": {
      "https://w3id.org/xapi/video/extensions/session-id": "cf0f3f7e...c182",
      "http://id.tincanapi.com/extension/latitude": 9.11,
      "http://id.tincanapi.com/extension/longitude": 22.09,
      "https://w3id.org/xapi/video/extensions/time": 4.5
    }
  },
  "timestamp": "2019-12-18T13:21:57.519Z"
}
```


Figure 3.5. Example of an xAPI statement.

Once both the LRP and the LRC agree on a specific structure of the statements, the communication between them is straightforward: the LRP posts these statements with the experiences of the users into the LRS and then the LRC just needs to query these statements and extract the relevant data which is included in them. This approach made it possible for us to develop the LRP and the LRC independently.

In this chapter, we report which xAPI statements are created and when they are posted into the LRS by the player application (the LRP) but we do enter into the details of how the analysis module (the LRC) queries them, as this is reported in [93]. For the last two evolutions of the platform, we performed in-depth studies with data that was exported from the analysis modules, which are reported in Chapter 4, Chapter 5 and Chapter 6.

3.3. An interactive 360-degree video activity

This is the first iteration of the omnidirectional platform. The paradigm of the interaction were educational activities, which consisted of a task to be performed in the reproduction of a 360-degree video: users were asked to identify specific features in the 360-degree video and double-click in the approximate location where they appeared (no visual hints, such as markers were provided) [94]. In this project, the creation module was divided in two separate tools: the 360-degree video annotation tool and the activity catalog management tool. The player module featured a drag interface designed to be used with computers. An initial exploration of the analyses that can be performed with this data is performed with the analysis module, which is explained in greater detail in [93].

3.3.1. The creation module

First, 360-degree videos were prepared in the *de facto* standard of MPEG4-AVC video stream with the spherical content projected following the equirectangular projection, MPEG AAC audio stream contained in MP4 files. They were paired with an XML metadata that includes the definition and location of the identified features in the video, converting them into annotated 360-degree videos.

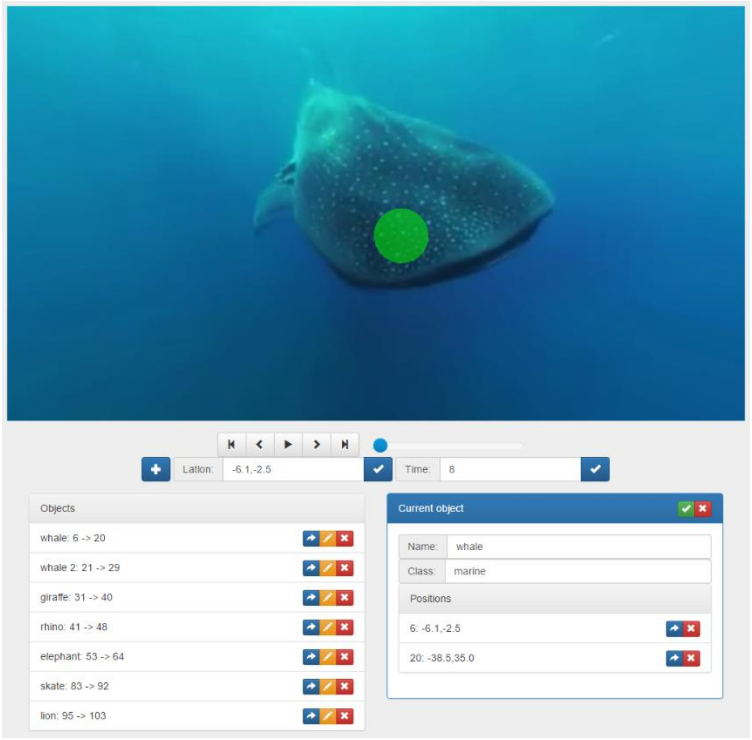

Figure 3.6. Screenshot of the 360-degree video activity annotation tool, with the list of objects on the left and the details of a specific object on the right, with two key positions.

This XML file was generated by the 360-degree video annotation tool, a web application that facilitates the upload of the base multimedia content and modify their metadata by adding new objects (or modifying existing ones) by specifying its name, classes and selecting positions in the spherical and temporal coordinates of the video $\{time, latitude, longitude\}$, using a drag-based user interface (see Figure 3.6). A set of $N \geq 2$ spatiotemporal coordinates define the trajectory of the object over time, from an initial to a final time: $\{\{t_1, \phi_1, \lambda_1\}, \ldots, \{t_N, \phi_N, \lambda_N\}\}$, where t denotes time; ϕ, latitude; and λ, longitude.

Activities are created from an annotated 360-degree video in the activity catalog management tool, which contains a web form that enables the activity creator to select a class of the objects that had been identified as the objective of the activity. Activities are then saved in an XML file, so an annotated 360-degree video can be used to create multiple interactive, measurable, and evaluable activities.

As an example, a 360-degree video documentary was used as the base media and the animals featured in it were identified and classified according if they were “marine animals” or “terrestrial animals”, so these were the two classes that were used in this first example, that resulted in two activities: one in which “marine animals” had to be selected and another one in which “terrestrial animals” were the required ones. In this same base multimedia content, with the same objects, other categorizations could have been possible, which would have resulted in a different set of activities (such as “mammals”, “fish”, “invertebrates” or “herbivores”, “carnivores”, “omnivores”).

3.3.2. The player module

This is the application that displays the activity as a web application (see Figure 3.7), so the users can interact with it and, together with the application server, acts as the Learning Record Provider in the xAPI paradigm.

The activity player requires users to log in with their e-mail address so they can be identified in the xAPI statements. Then, a list of available activities is displayed and once one of them is selected, it is launched, displaying a message with the task to be performed and the 360-degree video, which can be controlled spatially through a drag-based interface with a computer mouse and temporally through the play/pause button and a temporal slider. The user can double-click anywhere on the 360-degree space of the omnidirectional video to select in that direction, a circle is displayed to provide feedback of the action and, if the activity is configured to do so, a green or red glow may be displayed to denote that the selection was correct or incorrect. When the user feels that they have finished the activity, they can complete it by clicking the specified button. If the connection with the server is lost or an amount of time passes after the video is completed, the session is automatically terminated. All the objects that have not been selected are considered skipped at that point.

The video is rendered in a Three.js [95] scene, as the texture of a sphere, inside which the camera is placed. The drag of the user's mouse over the canvas triggers the rotation of the camera, so other areas of the 360-degree video can be explored. The application server is a Node.js application that uses Express as the HTTP server and Socket.io for real-time communication over WebSockets.

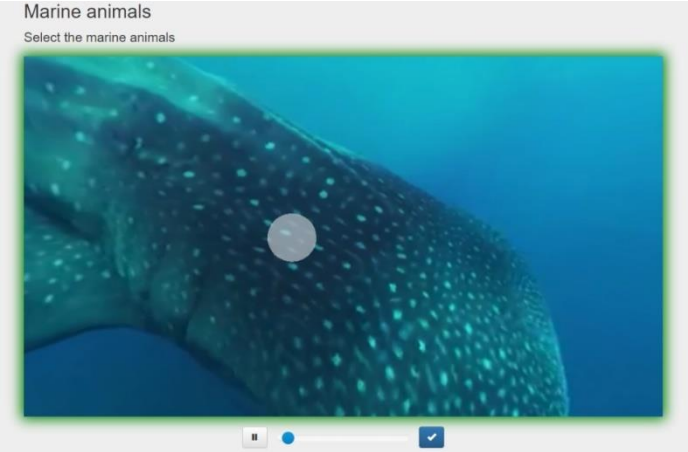

Figure 3.7. Screenshot of the 360-degree video activity player application, in the moment of a correct selection that displays positive feedback.

As the player application is connected to the application server via a WebSocket connection, a message is sent every time one of these actions is performed which, in turn, may generate a xAPI statement so the session can be analyzed. Information is provided through the return channel of the WebSocket connection, such as the list of activities, the details of the activity or (optionally) the feedback after issuing a selection.

The general xAPI vocabulary registry [96] and ADL's video vocabulary [97] were used in this application. New pieces of vocabulary have been defined: an activity type, *http://alumnes-ltim.uib.es/xapi/activity-type/360videoactivity*, extends the video activity defined in ADL's video vocabulary; the verb *http://alumnes-ltim.uib.es/xapi/verbs/lookedat* is used to report that the user has changed its orientation in the 360-degree video; the extension *http://alumnes-ltim.uib.es/xapi/extensions/activity* is used to report the title, class and video attributes of the activity in an object $\{title, class, video\}$, as it was created in the activity management tool; the spherical coordinates of the current viewport of the user in the 360-degree video are contained in an object $\{lat, lon\}$ in the extension *http://alumnes-ltim.uib.es/xapi/extensions/orientation*; finally, the extension used to report which object was selected or skipped is *http://alumnes-ltim.uib.es/xapi/extensions/object*, which contains an object $\{name, class\}$. In the rest of this section, we will refer these elements by the last part of their URI, such as *logged-in*, refer the vocabularies referenced above or this list.

The actor in each statement contains the users' e-mail address, as they introduce it when they log into the application. The object type *360videoactivity* is used in the *initialized* statements onwards, properly extended with *activity*. The field timestamp was explicitly specified due to time precision issues when relying in the storage time.

The flow of the player application is detailed next, refer Table 3.1 for a summary of these events and the xAPI statements that are generated.

Table 3.1. Summary of the events in the interactive activity player reported to the application server, the response received and the significant parts of the generated xAPI statements.

	xAPI statement		
Event → response?	**Verb**	**Context extensions**	**Result extensions**
Logged into the player → activity list	*logged-in*		
Selected an activity → activity details	*initialized*		
Pressed play	*played*	*orientation, session-id*	*time*
Pressed pause	*paused*	*orientation, session-id*	*time*
Sought the video	*seeked*	*orientation, session-id*	*time-from, time-to*
Changed the video orientation	*lookedat*	*session-id*	*time, orientation*
The video ended	*completed*	*orientation, session-id*	*time*
The user selected something → feedback?	*selected*	*orientation, session-id*	*time, object?*
Before finishing the session	*skipped**	*session-id*	*object*
The user finished the activity	*terminated*	*session-id*	
The activity expired	*abandoned*	*session-id*	

Many events do not have a response; those marked with ? may not have a response; an extension marked with ? may not be present; a statement marked with * may be fired multiple (or zero) times.

The first event is issued through the WebSocket channel when the user logs into the application, their e-mail address is received through oauth2 and the list of currently available activities is provided as an answer. A statement with the verb *logged-in* is issued when this happens. Once the user selects an activity, that information is submitted to the application server, which responds the metadata needed for the player to reproduce the activity and issues a statement with the verb *initialized*, in this case the type of the statement's object is already set to *360videoactivity*, identified by the IRI of the activity and extended with *activity*, so its title, class and video are specified in the xAPI statements. A reference to the id of this statement is stored in the server and used in future statements related to that session with the activity, in the *session-id* context extension.

When the user starts or resumes the video playback, a statement with the verb *played* is generated, and when the user pauses it, one with *paused* is submitted. They include the *time* result extension documented in the video vocabulary and the *orientation* extension in the context. In the case of a time position seek, a statement with the *seeked* verb is registered, which contains the *time-from* and *time-to* extensions in the result, according to the video vocabulary specification, and its context is extended by *orientation*.

The user is free to change the orientation of the 360-degree video at any moment. Orientation changes are tracked, at a maximum frequency of four times per second, and submitted to the application server to be reported in xAPI statements with the verb *lookedat*. The statement's result is extended by *time* and *orientation*.

When the video ends, a statement with the *completed* verb is generated to denote that the end of the video has been reached (this does not mark the end of the activity: the user can resume the playback from the start or seek to a previous position in time). The result extension *time* is added as required by the video vocabulary and its context is extended by *orientation*.

When a double-click is issued, a *selected* event is dispatched through the WebSocket connection, containing the spatiotemporal coordinates of the interaction: the spatial position, in spherical coordinates, of the selection and the time of the video in which that selection was made. The application server processes that information, $p_s = \{t_s, \phi_s, \lambda_s\}$ to check if an object has been selected (see Figure 3.8): first, the objects whose trajectory starts before and ends after t_s are considered ($t_1 \leq t_s < t_N$); then, for each pair of consecutive key positions $p_i = \{t_i, \phi_i, \lambda_i\}$, $p_{i+1} = \{t_{i+1}, \phi_{i+1}, \lambda_{i+1}\}$, in the trajectory of an object that $t_i \leq t_s < t_{i+1}$, the interpolated $p_t = \{t_t, \phi_t, \lambda_t\}$ is obtained through linear interpolation of ϕ and λ for t_s; finally, a selection window of $\phi_s \pm 15°$ in latitude and $\lambda_s \pm 30°$ in longitude is built over the selected position p_s, so any object whose interpolated position for t_s falls

inside this window is considered selected. An xAPI statement with the *selected* verb is issued in these cases, with its context extended by *orientation* ($\{\phi_s, \lambda_s\}$) and its result by *time* (t_s) and, if an object fell inside the selection window, the *object* description. The success property of the result is set to whether the classes of the activity match the classes of the selected object. An answer detailing if the selection was successful or not is sent to the player application if the activity is configured to do so.

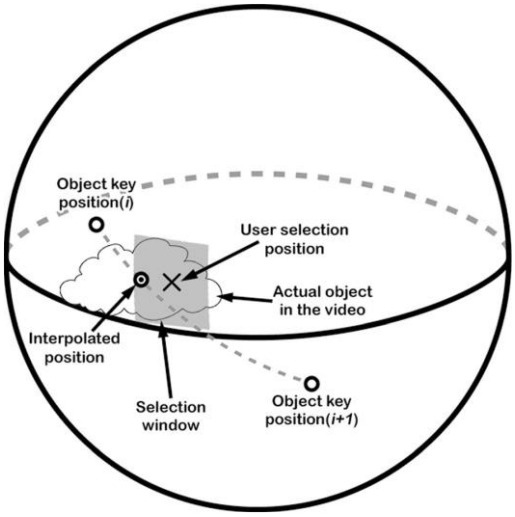

Figure 3.8. Illustration of the process of determining if an object is contained in the selection window built around the user-supplied selection position in the 360-degree video activity player application.

The last event is the completion of an activity. This can happen when the user chooses to end it by pressing the specified button or when it automatically expires for one of these reasons: a new activity is being launched, the WebSocket connection was closed or the assigned time to complete the activity ran out. When this happens, first, an xAPI statement with the verb *skipped* is issued for each object not selected at least once during the activity, extended with the *object* details as a result extension; second, a statement with either the verb *terminated* (the user chose to end the session) or *abandoned* (the session expired) is posted to indicate that the user has completed the activity. The result of the statement is detailed with the completion property set as true if it was issued by the user and to false if it was expired, and its duration is set to the length of time since the activity was launched.

3.3.3. Results

The proposed system is able to successfully create, reproduce, register, and analyze an interactive experience based on 360-degree videos in a web environment.

One of these graphs that can be obtained through the analysis module plots the spatiotemporal trajectories of the sessions (Figure 3.9), in which the x axis shows the relative timestamp since the launch of the activity (left) or the longitude in the users' current orientation (right), while the y axis reports the current time in the video. Thick lines are built with statements with spatiotemporal information (those with verbs *played*, *paused* and *lookedat*), markers denote other events and pointed lines mark time seeks. In the right, we can appreciate the annotation of the objects in the video (indigo for valid objects, pink for invalid objects) and the spatiotemporal position of valid and invalid selections.

Other kinds of graphs include time and longitude histograms and contextual charts that process the data in the context field of xAPI statements.

We found that spatiotemporal trajectory graphs (such as Figure 3.9) provide a quick glance to the performance of the users in the temporal dimension and in the spatial dimension; histograms can help identify the most visited portions of the 360-degree videos, both in the temporal and spatial dimensions; finally, contextual charts process the data in the context in which the actions were performed, such as the direction of temporal seeks (forward or backwards) or the direction of viewport changes (clockwise or anticlockwise).

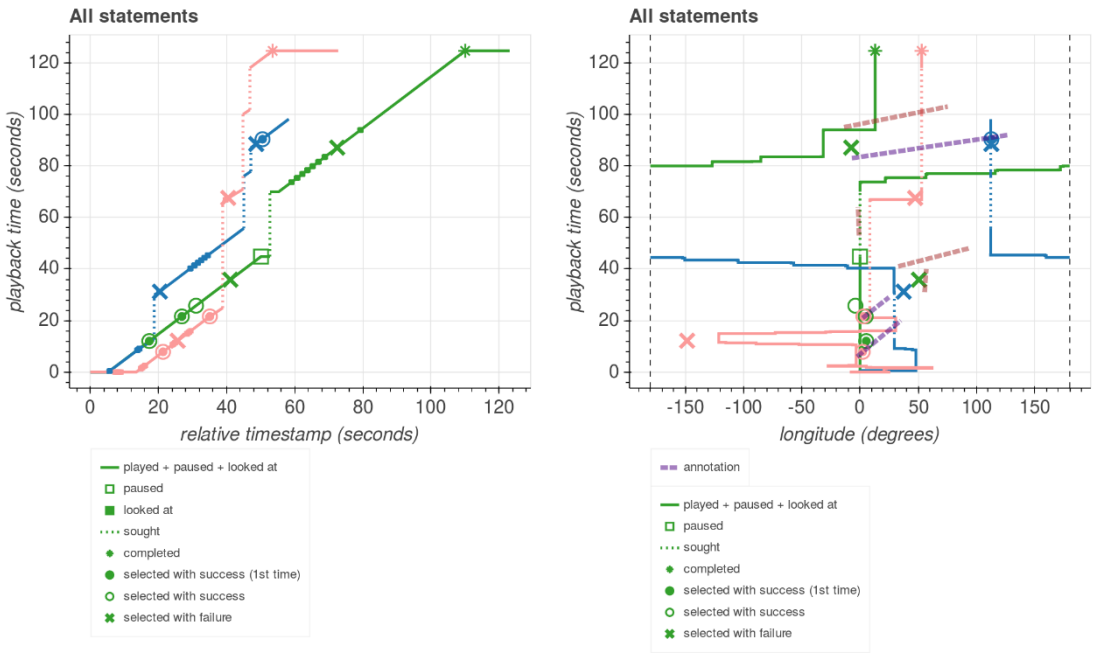

Figure 3.9. Session spatiotemporal trajectories in the 360-degree video activity player in video time (left) and longitude (right).

3.4. A 360-degree hypervideo documentary

In this second iteration of the multimedia platform, we diverged from the educational activities and produced an interactive documentary delivered on a 360-degree hypervideo. One of our colleagues collaborated with PROEM-AID[16], a group of emergency professionals who voluntarily helped during the European migrant crisis in Greece and recorded a few 360-degree video clips and some other regular clips that we turned into an interactive hypervideo production [98], [99].

The hypervideo consists of a main 360-degree video that acts as a panorama, allowing the user to pan the scene and look at any direction, and acts as a basis of the documentary, as it displays links to additional media in specific points in the space-time of the main track.

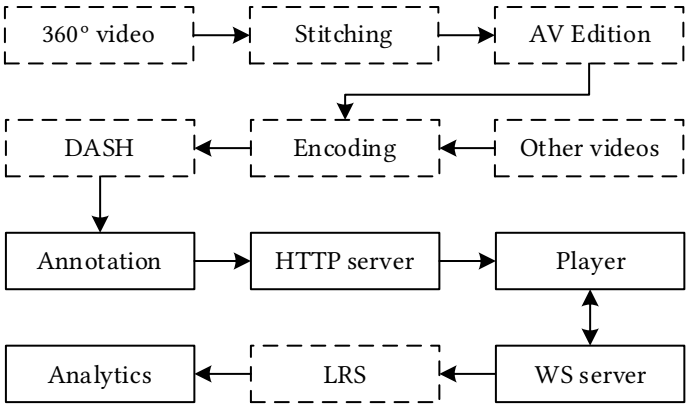

Figure 3.10. General overview of the 360-degree hypervideo documentary system.

An overview of the system is provided in Figure 3.10, which shows solid boxes for processes for which a tool was developed to perform them. Multimedia sources have undergone a prior process of edition and encoding with state-of-the-art tools. The hypervideo was created with the aid of the annotation module and is reproduced in the player module, which collects the users' actions in the LRS. That information is later queried by the analysis module, which provides some insight of the users' behavior and was used to perform a more in-depth user study [100]. The annotation and the player modules

[16] https://www.proemaid.org/en/lesbos/

are React web applications and use the same component to display the 360-degree video track: to annotate it or to reproduce it.

3.4.1. The creation module

Two kinds of video footage were recorded in the refugee camps of Lesbos and Piraeus, Greece: 360-degree video and linear clips. Four 360-degree scenes of approximately 90 seconds each were captured using a 6-camera array, rigged on a stick. The equirectangular media was stitched together with Kolor's Autopano Video Pro in a single video file with a duration of approximately six minutes. Eleven linear videos with durations between 30 and 137 seconds, amounting to more than 14 minutes in total, with different testimonies of the refugees, were prepared to extend the production.

The 360-degree video and the linear clips were encoded as MPEG4-AVC video and MPEG AAC audio. As the source multimedia was captured in high quality and the stitched 360-degree video was obtained from six cameras, the original bit rate of the videos was high, which could be a problem in limited network conditions. The MPEG DASH [55] adaptive streaming technique was employed: multiple representations of each video (linear clips and the 360-degree media) were encoded with different bit rates, logically connected by an MPD manifest for each video.

A new annotation tool was developed following the same principle as the one from the previous iteration. This time, a React web application was designed, which interfaced a component that displays the 360-degree video in a Three.js 3D scene with the forms required to annotate the points of interest in the 360-degree video. A point of interest is represented by a marker on the 360-degree video defined by $N \geq 2$ key positions in the spatiotemporal coordinates of the video, $\{\{t_1, \phi_1, \lambda_1\}, \ldots, \{t_N, \phi_N, \lambda_N\}\}$, where t denotes time; ϕ, latitude; and λ, longitude. The additional metadata for each point of interest includes a descriptive name, an IRI (Internationalized Resource Identifier) to identify that point in the xAPI statements, the IRI of the video that should play when it is selected (which is the URL of the MPD file that was previously generated) and an optional image to customize the appearance of the marker on the 3D scene. This information is stored in a JSON file and retrieved by the player.

3.4.2. The player module

Once the 360-degree hypervideo has been defined, it can be reproduced in the player web application (see Figure 3.11), which was implemented in React, with the same component for the scene as the annotation module, but this time it displays the video controls and the help area. Events in the player application are forwarded to the application server through a WebSocket connection, so xAPI statements are generated to register the users' actions.

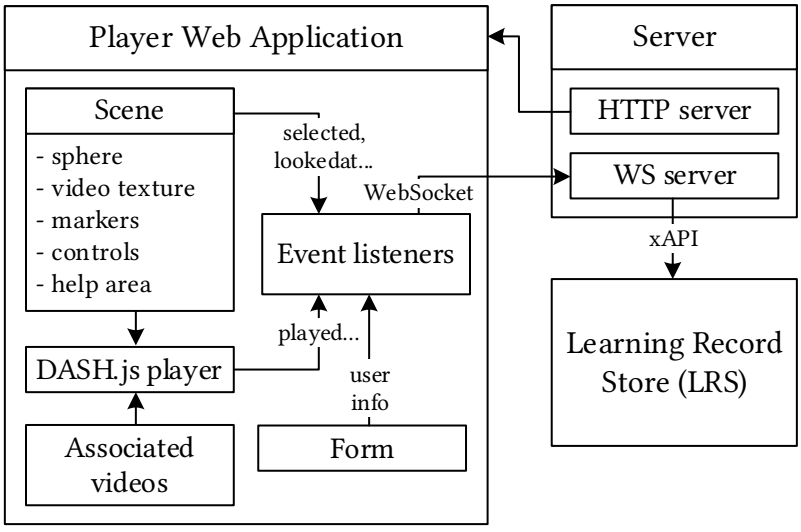

Figure 3.11. Detail of the architecture of the player application in the 360-degree hypervideo documentary.

When the player is started, a form prompts the user to voluntarily share some basic personal information (gender, age, profession, country) with us, which is used to build the actor part in the

xAPI statements. The user can choose not to fill this form, which makes the difference between registered and unregistered consumers.

After that, the user is free to navigate the video in space (by dragging the scene with a pointing device, such as a mouse or the finger over the screen on a smartphone) and time (with the video controls shown in the lower part of the screen). When a point of interest appears, a marker is displayed according to the trajectory defined in the annotation tool, so the user can find it in the 360-degree video and select it. If they do, the main playback is paused and the associated media is played in a pop-up view, until it ends or the window is closed by the user, so the main playback continues. Whenever any of these events happen, information is submitted to the application server through a WebSocket connection, so xAPI statements are posted to the LRS to keep track of the users' actions. Finally, since video files follow the DASH specification, we leverage the DASH.js player to parse the MPD files and automatically adjust to a representation with a suitable bit rate.

The Scene component, which is shared between the creation and the player modules, uses a WebGL scene managed by Three.js. The core of the scene is the sphere geometry used to display the frames of the 360-degree video texture, provided by the DASH.js player. A list of points of interest with their trajectories is provided so the scene can display the markers when it is needed. This list is updated dynamically in the editor, but it is obtained from the JSON produced by the creation module when the scene is in the player module.

$$d = \operatorname{acos}[\cos\phi_0 \cdot \cos\phi_1 \cdot \cos(\lambda_0 - \lambda_1) + \sin\phi_0 \cdot \sin\phi_1] \tag{3.1}$$

The position of a given marker in a time t, when there exist two consecutive key positions in its trajectory $p_0 = \{t_0, \phi_0, \lambda_0\}, p_1 = \{t_1, \phi_1, \lambda_1\}$ that $t_0 \leq t < t_1$, is calculated via linear interpolation in spherical coordinates (3.2), which requires the angular distance between two points in spherical coordinates (3.1). Both formulas are adapted from the "Intermediate points on a great circle", and "Distance between points", respectively, both described in [101].

$$\begin{aligned}
f &= \frac{t - t_0}{t_1 - t_0} \\
A &= \frac{\sin\big((1-f)\cdot d\big)}{\sin d} \\
B &= \frac{\sin(f \cdot d)}{\sin d} \\
x &= A \cdot \cos\phi_0 \cdot \sin\lambda_0 + B \cdot \cos\phi_1 \cdot \sin\lambda_1 \\
y &= A \cdot \sin\phi_0 + B \cdot \sin\phi_1 \\
z &= A \cdot \cos\phi_0 \cdot -\cos\lambda_0 + B \cdot \cos\phi_1 \cdot -\cos\lambda_1 \\
\phi_t &= \operatorname{atan2}\left(y, \sqrt{x^2 + z^2}\right) \\
\lambda_t &= \operatorname{atan2}(x, z)
\end{aligned} \tag{3.2}$$

Since Three.js uses a cartesian coordinate system, we cannot use $p_t = \{t_t, \phi_t, \lambda_t\}$ in spherical coordinates. (3.2) already provides the cartesian coordinates of p_t as unit vectors (the values of x, y, z), so they need to be scaled to the radius, R, of the sphere on which the markers should be drawn (3.3). Note that R should be smaller than the radius of the sphere on which the texture of the 360-degree video is drawn to avoid clipping the markers out of the view of the camera.

$$p_t = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3.3}$$

In this case, ϕ_t and λ_t from (3.2) are never used and there is no need to calculate them.

Once markers are displayed on the 3D scene and their position is updated accordingly, selections on them are detected with a ray casted from the middle of the scene (where the camera is) to the position selected with the pointing device (a computer mouse or the touch on a touchscreen), which fire an event to launch the associated video.

Other player-specific features are enabled on the scene component, which are not enabled in the creation module, (see Figure 3.12): the video controls serve to adjust the temporal dimension of the

360-degree video, with a play/pause button and a seekable progress bar that hints the appearances of points of interest; while the help zone displays in a specific location the markers that are drawn somewhere in the 3D scene (as we just explained), so that users cannot miss them if they appear out of their viewport.

Figure 3.12. User interface of the 360-degree hypervideo documentary player.

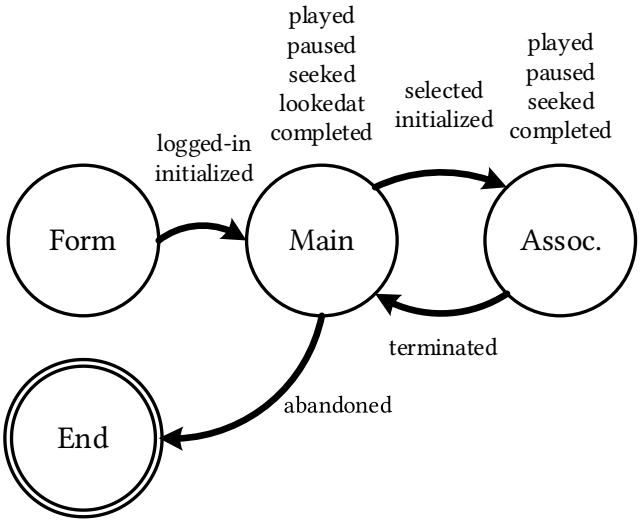

Figure 3.13. State diagram of the 360-degree hypervideo documentary player, with the names of the verbs in the submitted xAPI statements.

The player application follows a set of states, which are transitioned after certain events and xAPI statements are issued in transitions and some statements are generated in one state or the other (refer Figure 3.13). The general xAPI vocabulary registry [96] and ADL's video vocabulary [97] were also used in this application, with a similar principle as in the previous iteration. Refer Table 3.2 for the relation of the statements and their parts. The activity type *https://ltim.uib.es/xapi/activity-type/360augmentedvideo* is used throughout the statements, except in those related to associated videos, in which the activity type *https://w3id.org/xapi/video/activity-type/video* is used. The activity id is set to the URL of the video except in *logged-in* statements, which use the URL of the web application. The users' information collected in the form is included in the *logged-in* statement in the result extension *https://ltim.uib.es/xapi/extensions/user-data*, which contains an object {*name*, *email*, *gender*, *age*, *profession*, *country*, *accept-language*, *user-agent*, *uuid*}. Information about the video that is initialized is attached in the result extension of *initialized* statements as an object {*duration*} in *https://ltim.uib.es/xapi/extensions/video-data*. The following are used as context extensions in video statements (*played*, *paused*, *seeked*, *completed*, *lookedat*, *selected*, *terminated*), following the xAPI video specification: *https://w3id.org/xapi/video/extensions/quality* to report the current quality in vertical pixels in which the video is played, such as 720 or 1080; if the video is played or not in full screen is reported in *https://w3id.org/xapi/video/extensions/full-screen*; the size of the screen is included in *https://w3id.org/xapi/video/extensions/screen-size*, like 1920x1080, 1280x720, or 640x480; *https://w3id.org/xapi/video/extensions/user-agent* tells the User-Agent of the browser; *https://w3id.org/xapi/video/extensions/volume* reports if the mute button was used; *https://ltim.uib.es/xapi/extensions/orientation* is used (except in *lookedat*) to report the current

orientation in statements related to the main video track; since events can be fired while the video is paused, *https://ltim.uib.es/xapi/extensions/playing* reports if that is the case. The current time in the video is specified in *https://w3id.org/xapi/video/extensions/time* (except in *seeked*) as a result extension (except in *lookedat* and *selected*, in which it is context). The result extension *https://w3id.org/xapi/video/extensions/time-from* is used as a result extension to specify the point in time from which the user performs a seek and, similarly, the destination is specified in *https://w3id.org/xapi/video/extensions/time-to*. Analogously, the changes of orientation are reported in *lookedat* with *https://ltim.uib.es/xapi/extensions/orientation-from* and its destination counterpart *https://ltim.uib.es/xapi/extensions/orientation-to*. Finally, the URI of a point of interest is indicated as a result extension in *selected* statements in *https://ltim.uib.es/xapi/poi*. As in the past section, in the rest of this one, we will refer these elements by the last part of their URI.

Table 3.2. Summary of the events in the 360-degree hypervideo documentary player and the significant parts of the generated xAPI statements.

	xAPI statement		
Event	**Verb**	**Context extensions**	**Result extensions**
The form is submitted	*logged-in*		*user-data*
A video track is loaded	*initialized*	*session-id*	*video-data*
A video track is resumed	*played*	*orientation** *session-id*	*time*
A video track is paused	*paused*		
A video track ends	*completed*		
A video track is sought	*seeked*	*orientation** *session-id*	*time-from* *time-to*
The orientation is changed	*lookedat*	*time* *session-id*	*orientation-from* *orientation-to*
A marker is selected	*selected*	*time* *orientation* *session-id*	*poi*
The associated video is closed	*terminated*	*session-id*	*time*
The player is closed	*abandoned*	*session-id*	

The *orientation** is only included in statements related to the main video track.

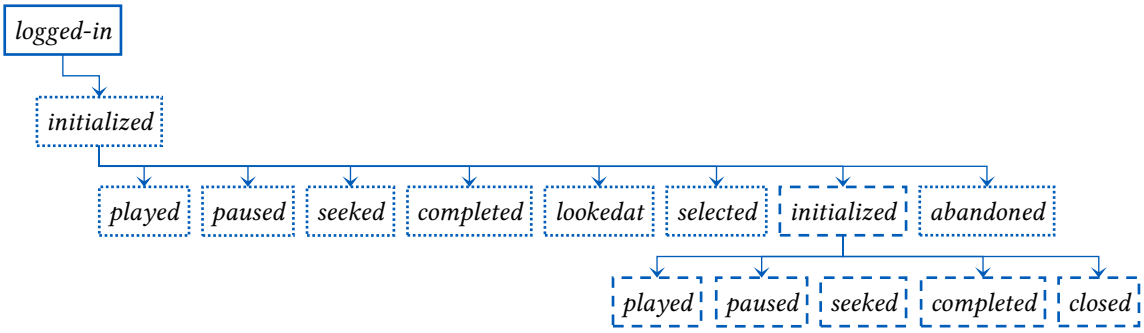

Figure 3.14. Hierarchical sessions in the xAPI statements of the 360-degree hypervideo documentary player. Statements in dotted boxes are related to the main 360-degree video, while those in dashed boxes are related to the associated videos.

The actor for these statements is specified with either the e-mail address they provide in the form or a version 4 UUID as the name of a local account, to deal with anonymous users who dismissed the form. A hierarchical system of sessions is used in the statements, influenced by the xAPI video vocabulary (see Figure 3.14): first, the id of the *logged-in* statement is used as the *session-id* on the *initialized* statement of the main video track; second, the id of that *initialized* statement is used as the *session-id* in the statements that are performed on the main video track (*played*, *paused*, *seeked*, *completed*, *lookedat*, *selected*), the *initialized* statement when an associated video starts and the *abandoned* statement when the application is closed; finally, the id of these *initialized* statements of associated videos are used as *session-id* in the statements fired from that video track (*played*, *paused*,

seeked, *completed*, *terminated*). This enables us to relate all statements to a single session and to distinguish behavior from the main video track to each associated video.

The application starts in the *Form* state, in which the aforementioned form is displayed. Once it is submitted (or dismissed), an xAPI statement with the verb *logged-in* is issued extended with the submitted information (if any). This triggers the change to the *Main* state, so a statement with the *initialized* verb is fired as soon as the main video track is ready.

Statements with verbs *played*, *paused*, *seeked* and *completed* are used in the *Main* state accordingly to the video vocabulary. The extension *orientation* is included to specify the current viewport of the user in the 360-degree video. Statements with the verb *lookedat* are sent when the user changes its current view. Successive submissions are throttled to 250 ms so as not to flood the LRS. A statement with *selected* verb is issued when a marker is selected and the application transitions to the *Associated* state, so a window is created to play the video associated to the selected point of interest and a statement with the *initialized* verb is fired as soon as that video track is ready, starting an associated video subsession (see Figure 3.14).

In the *Associated* state, a linear video is played in a window. Statements with the verbs *played*, *paused*, *seeked* and *completed* are issued similarly as before, but without the *orientation* extension and specifying the id of that last *initialized* statement as its *session-id*. That subsession ends when the video window is closed, a *terminated* statement is issued, and the player returns to the *Main* state. More points of interest can be selected, starting a new subsession, or the user can simply close the player, so an *abandoned* statement is generated, signaling the end of the session.

3.4.3. Results

The production was used 71 times during 25 days after the link was shared through the University community, generating a total of 11139 xAPI statements. Of these 71 sessions, 37 were registered consumers (33 of them unique).

A summary of the findings is provided below, please refer [93] for more details.

The population mean of different selected markers per session is between 1.6 and 2.8 at a 95% confidence level, notably less than the 11 points of interest available. In 69% of the sessions, at least 1 point was selected, which is 70% of the sessions from registered consumers and 68% of the sessions from unregistered ones.

A strong, negative, and statistically significant correlation was found between the number of sessions in which a marker was selected and the time in which they appeared first ($\rho = -0.82, p = 0.002$).

Associated videos tend to be watched in its full length: the 95% confidence interval of the mean proportion of playback is between 56.2% and 90.8%. 55% of video plays had at least 93% of playback time visualization.

The main 360-degree video tends not to be watched in its full length, either because users skip parts of it with seeks or because they close it before it ends. The 95% confidence interval of the mean proportion of playback time is between 30.8% and 45.9%. The correlation between the proportion of playback time visualization of the main 360-degree video and the number of unique selected markers was positive, strong, and statistically significant in registered sessions ($\rho = 0.65, p < 0.001$), and positive, moderate, and statistically significant in unregistered sessions ($\rho = 0.47, p = 0.005$).

Finally, the 95% confidence interval of the mean proportion of longitude visualization per session is between 57.2% and 77.3%. The correlation between the proportion of longitude visualization of the main 360° video and the number of unique selected markers was positive, strong, and statistically significant in registered sessions ($\rho = 0.60, p < 0.001$), but insignificant in unregistered sessions ($\rho = 0.25, p = 0.147$).

3.5. A multi-device virtual reality museum

A virtual museum [102] was conceived as the use case of the third iteration of the interactive platform to demonstrate that our system is capable of supporting full VR environments. Although 360-degree

videos are left aside this time, the interaction in a 3D scene is very similar, so improvements in this iteration would be easily translated to the reproduction of 360-degree videos. The most important addition was the possibility to interact with the virtual scene with four kinds of devices: desktop computers, smartphones, desktop VR headsets (we used the Oculus Rift) and mobile VR headsets (such as the Google Cardboard).

As the IT service of the University started a project to document vintage computers used by professors in the past, we thought it would be interesting to be able to interact with these computers that are now either out of service or in display. A virtual museum, in which the users could turn these computers on, seemed a good experiment to bring these computers back to life again. Since most of these computers were made by Apple, we decided to only consider Macintosh computers to make it more consistent. As pictures of these were captured using a greenscreen, we could easily import them in a virtual scene, so users would be able to explore these vintage computers. As a twist, we downloaded the original TV commercials in which these computers were advertised years ago from the Internet, so they would appear to be played inside the screens of the relevant computers.

The system (see Figure 3.15) evolves from the previous iteration and follows roughly the same architecture. We finally took under consideration four different devices to play the application, which would present differences in the user interface. The player application, which is run inside a web browser, is obtained from the application server via HTTP protocol and transmits usage data back to the application server through a WebSocket connection. The monitor application is an addition which was made to perform a simple live analysis of the users' movement. To do so, a WebSocket connection is also stablished with the application server, so the data generated by the player application is forwarded to the monitor. The application server also submits this data to the Learning Record Store (LRS) with xAPI statements to be later retrieved by the analysis module, so a proper analysis can be conducted on that data.

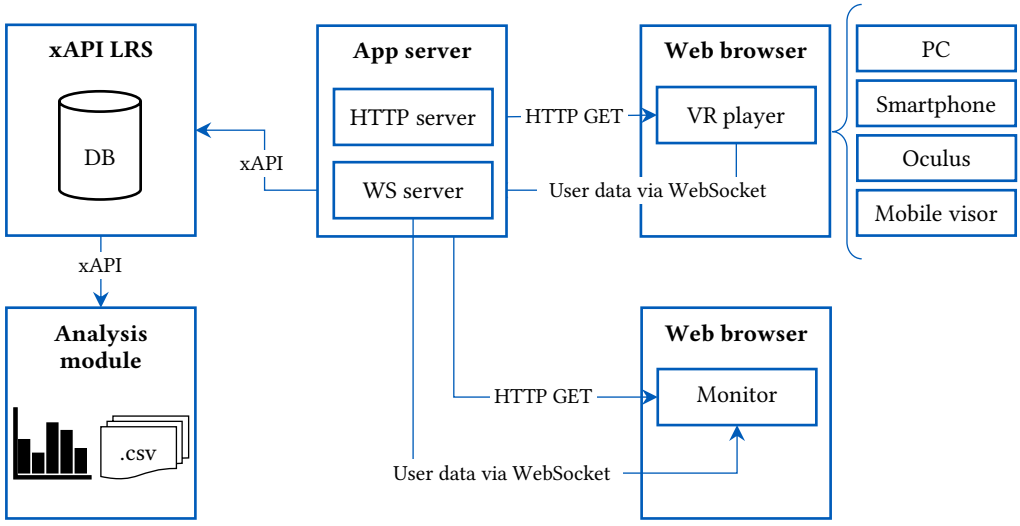

Figure 3.15. General overview of the system in the virtual museum.

In contrast with previous iterations, no creation module was developed for this system, as a single museum was intended to be implemented, so the data of the objects to be displayed in it was specified in a configuration file.

3.5.1. The player module

This is the web application that displays the virtual museum. 39 Apple Macintosh computers released between the decades of 1980 and 2000 are shown in the virtual scene, grouped by *family*, so that users are able to observe them and interact with them, so they are turned on, the advertisement which was aired when it was released is displayed in its screen and its technical specification is displayed next to it.

The virtual world is displayed very similarly in all the devices (see Figure 3.16): the player stands in a first-person view in the middle of a large room, surrounded by Macintosh computers that have their

name below them. A virtual cursor appears in the middle of the screen, which changes its size to notify the user when it points to a valid target, is used to select the computers. The "Enter VR" button in the corner is used to display a stereoscopic version of the scene, so it can be viewed with a VR headset, such as the Oculus Rift or a mobile headset like the Google Cardboard.

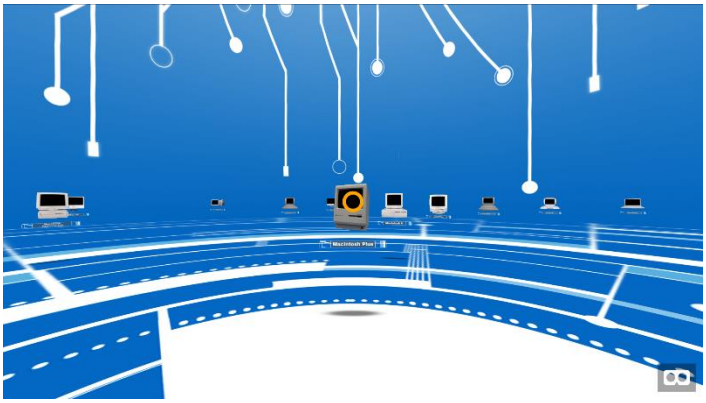

Figure 3.16. The virtual museum. Note the cursor in the middle and the "Enter VR" button on the bottom right corner.

Despite the similarity in the virtual world, the user interface was designed with important differences between devices. A summary is given in Table 3.3, but the interaction is detailed next. To display the application in non-immersive devices (PC, smartphone), the user simply needs to browse to the application's URL, while in immersive devices (Oculus, mobile visor), the VR mode needs to be enabled by selecting the "Enter VR" button.

Table 3.3. User interfaces and interaction techniques in the virtual museum.

Interface name	Hardware	Orientation	Movement	Selection
PC	Computer, monitor, mouse, keyboard	Mouse	Keyboard	Click
Smartphone	Smartphone	Gyroscope	By selection	Tap
Oculus	Computer, Oculus Rift + Touch	Head	Joystick	Virtual hands
Mobile visor	Smartphone, Cardboard-like headset	Head	By selection	Gaze

When the application is accessed with a personal computer, no special hardware is required: the user can look around by moving the mouse (there is no need to drag, as the application captures the cursor), turning their viewport similarly to a first-person videogame; movement is performed using the keyboard WASD or arrow keys, so they can navigate the environment freely; selection is issued with a click when the cursor is on a target.

On a smartphone, the hardware changes drastically: the user can benefit from the gyroscope of the device to look around the scene; their movement is limited and is performed through the selection of a target: when the user looks at a computer and taps the screen, the selection is performed and the player will automatically move next to that object, so they can watch the video and read the technical information.

Figure 3.17. Detail of the selection of a computer with the Oculus interface (left) and the others (right) in the virtual museum.

When a desktop VR headset is used, like the Oculus rift, the user enjoys a true VR experience and is able to look around by turning their head; movement is performed with the joysticks in the Oculus Touch controllers (labeled as 1 in Figure 3.18-left); selection is performed by touching the computer with a virtual hand while the user makes the "point gesture" (see Figure 3.18), which is performed by touching the thumb rest (2), not pressing the trigger (3) and pressing the grip button (4). The use of this gesture eliminates the need of the virtual cursor. In this case, the technical description appears on the palm of the other virtual hand, as one would read a document on their hand (see Figure 3.17).

Figure 3.18. Detail of the Oculus Touch controller and the "point" gesture in the virtual museum.

Finally, if a mobile VR headset is used, the user should place their smartphone inside the headset after pressing the "Enter VR" button. Although the interface is similar to the smartphone interface, as the screen is now inaccessible, this introduces a significant constraint. Orientation is performed similarly, thanks to the sensors of the device, which is now attached to the user's head; movement is also performed when a target is selected; the difference lies in how they are selected, as the user cannot interact with the touchscreen of the device: a "gaze" control is introduced, which will fire a selection when the user looks at an object during a dwell time of two seconds, as in [15].

As we needed to use VR devices in the web application, we chose to leverage a WebXR (then, WebVR) framework. A-Frame was chosen because it was already relatively mature, has a rich example set, good documentation and community support. A-Frame internally uses Three.js to interface with WebGL. To build the web application, React was chosen as a framework. We used the aframe-react package to interface these two libraries. The communication with the WebSocket protocol is leveraged by the Socket.IO library.

The Macintosh computers are included in the application with a YAML file that specifies their name (used to choose their image, alpha and video textures), the family to which they belong, the sound that is played when they boot, and the adjustment done to the bounding box so selections in the Oculus interface roughly correspond to what is actually visible.

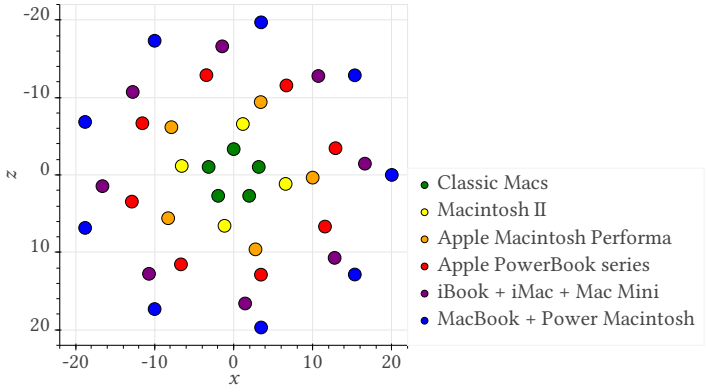

Figure 3.19. Macintosh computer layout in the virtual museum, grouped by families.

The computers are displayed in six concentric rings, one for each family, so oldest computers are placed in the middle of the scene (see Figure 3.19). Each computer is rendered in an entity that implements the custom *look-at* component to make them face the camera, plays the computer's sound when the event *power-on* is received and listens the events *power-on* and *power-off*. As a child of that entity, the currently selected computer is defined in an *a-video* entity, while the others use a *a-image*, all using the same components: an *alpha-map*, to display transparency; a *transparent* material; *width*,

height, *offset* and *repeat* set according to the bounding box in the configuration file; and listens to *click* events to know when it has been selected. Other entities displayed as children of that first entity are their title, shadow and description as *a-image*s, this last one only in the currently selected computer and in all interfaces but Oculus.

The player entity, which represents the user, implements the custom components *insideradius* to prevent the user to move farther; *joystick* to handle the Oculus Touch input; and *my-wasd-controls* to handle the keyboard input considering the current orientation of the player. It has three children: the camera and the two hands. The camera entity implements the *look-controls* component to handle the orientation controls and contains the cursor entity, which implements the *cursor* component with the *fuse* property set whether the mobile visor interface is in use, to automatically select a computer by looking at them; the *raycaster* component to know which computer the user is pointing at; and listens to the events *raycaster-intersection* and *raycaster-intersection-cleared*; and the *click*, *focusin* and *focusout* animations are used to provide visual feedback. Both hands implement the *hands-controls* component (with *left* and *right* properties, respectively) to display interactive, virtual hands and the custom *hand-collider* component to be able to select objects with them. Their children implement the *raycaster* component to be able to select the computers with a virtual finger and display an *a-image* entity on the other hand with the description of that computer.

The scene is applied the *mac-scene* component to forward the position and orientation changes of the player to the application server through the WebSocket connection (throttled at four times per second). The other components also forward data to the application server: when a session is created when the connection is stablished; when the scene is loaded; when a video in a computer starts, stops, or finishes playing; when the user enters or leaves the VR mode with the "Enter VR" button; and when the session ends when the connection is closed.

The information is submitted to the application server via the WebSocket connection which, in turn, posts it into the LRS with xAPI statements. The statements are composed of an *actor* (which is identified by a server-generated UUID, which is stored in a cookie, so returning users can be identified); the *verb* that defines the action which is performed; the *object* on which the action is performed (a virtual reality application or a video, identified by an URI); and additional information in the *context* and *result* fields. The general xAPI vocabulary registry [96] and ADL's video vocabulary [97] could be adapted to work with a Virtual Reality application. We introduced some new pieces of vocabulary, which we will detail here so we can refer them by the last part of their URI. To record some of the data from the HTTP headers in the logged-in statement, the context extension *https://ltim.uib.es/xapi/extensions/accept-language* was included together with *referrer* and *user-agent*. The *interacted* events include the kind of device which was used in the context extension *https://ltim.uib.es/xapi/extensions/vr-device*: it contains an object which tells us if the player is in VR mode, if the device has positional tracking, if it is a headset and if it is a mobile device. Finally, *https://ltim.uib.es/xapi/extensions/trigger* is included in the context of *played* and *paused* statements to specify how the action was performed (was it a *click*, a *fuse* or which *hand* was used).

Following Table 3.4, a statement with the verb *logged-in* is submitted as soon as the connection is stablished. An *initialized* statement is submitted when the application finishes to load, which contains metadata extracted from the HTTP headers. When the position or orientation of the player changes (throttled at four times per second), or when they toggle the VR mode of the application, an *interacted* statement is posted with the current data of the device used to play the application and their position and orientation. If the video which is played in a computer starts, is paused, or ends, *played*, *paused* and *completed* statements are submitted with the previous information and data about the video playback: its time, volume and how it was *triggered* (except when it ends, as it is automatic). Finally, when the connection is lost, an *abandoned* statement is posted. Note that the object in the statement changes according to the action performed: when a video is interacted, the *video* activity type is used, while in the rest of the cases, the *virtual-reality* activity type is used. In both cases, they are identified by their URI. The use of *session-id* is very similar to our previous experiment (see Figure 3.14): the ID of the *logged*-in statement is used as *session-id* for *initialized* and the ID of *initialized* is used for the rest.

Table 3.4. Summary of the events in the virtual museum reported to the application server and the significant parts of the generated xAPI statements.

Event	xAPI statement			
	Verb	**Object**	**Context extensions**	**Result extensions**
Connection	*logged-in*	*virtual-reality*	–	–
Scene loads	*initialized*	*virtual-reality*	*accept-language* *referrer* *user-agent* *session-id*	–
Moves	*interacted*	*virtual-reality*	*vr-device* *latitude* *longitude* *position* *session-id*	–
Looks around				
VR mode is changed				
Video played	*played*	*video*	*trigger** *volume* *vr-device* *latitude* *longitude* *position* *session-id*	*time*
Video paused	*paused*	*video*		
Video ends	*completed*	*video*		
Disconnection	*abandoned*	*virtual-reality*	*session-id*	–

*trigger** is only included in *played* and *paused*.

Finally, meanwhile xAPI statements are posted to the LRS, the server forwards in real time some of the events in Table 3.4 to another web application: the Monitor application. This application serves as a map that shows in real time where all the users are in the 3D scene. The final goal of this small application was to be a first step of real-time analysis of the users' data.

3.5.2. The monitor application

The Monitor application was also implemented with the A-Frame framework, so it borrows many features from the VR Player, but in this application, the view is locked in a top-down perspective and players are drawn on the scene as triangles (so their current location and orientation can be observed) and a line is drawn behind them, which represents their past locations in the 3D scene (see Figure 3.20).

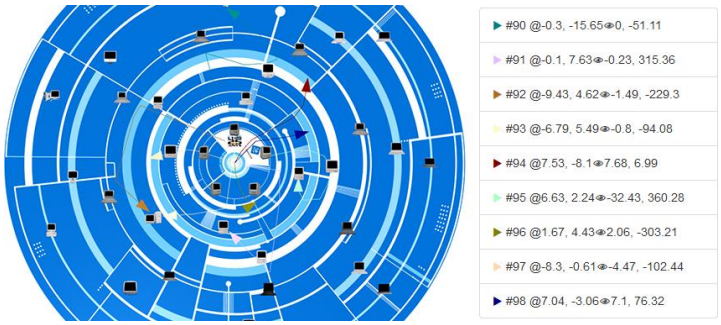

Figure 3.20. The Monitor application of the virtual museum, displaying all players in the virtual scene

The communication is performed via a WebSocket connection with the application server: the Monitor application notifies the server that it is an application of this kind (multiple monitors are supported at the same time) and then the server simply forwards the changes emitted by the VR players to the Monitors, so the 3D scene and the list on the right of Figure 3.20 are updated accordingly: a new player is added when they connect and they are removed when they disconnect, in the other cases, their position and orientation is updated according to the data that is supplied to the application server.

3.5.3. The user experiment

The system was demonstrated at the science fair at the University. Hundreds of students between 6 and 16 years old attended the sessions on three consecutive days, of which many visited our stand.

Figure 3.21. The setup of the virtual reality museum study: on the left, a player uses the Oculus interface, while what she sees is displayed on the left screen and the Monitor application is shown on the right screen; on the right, a group of students use the Cardboard interface.

Of the four available interfaces, two were present in our venue (see Figure 3.21): we set up a computer with the Oculus interface in front of a table (which had two screens, one that displayed what the user sees through the VR headset and another that displays the Monitor application) and several Cardboard-like mobile VR headsets on another table, so attendants could browse to the web application with their own devices and slide them into those visors in groups of approximately 6 people. The participants were given a short training before they started using the system, but they were not given a specific task to perform more than to explore the virtual world during a slot of time of approximately 5 minutes.

3.5.4. Results

A primary analysis of the data that was collected during that study shows that 175 mobile sessions which selected at least a computer were recorded. As the Oculus interface was overwhelmingly successful (a single device was available), participants did not respect the instructions so individual sessions per participant could not be recorded and the data is aggregated. In total, computers were selected 1436 times from mobile devices and 995 times from the Oculus interface.

A visual analysis of the user behavior (computer selections and movement) can be done from Figure 3.22: in Figure 3.22 (a), we can observe that computers in the upper part are more selected with the Oculus interface than those in the lower part. Their paths in the 3D scene can be observed in Figure 3.22 (b), with a clear difference between the upper and lower part of the figure. When the Cardboard interface is used, the results are different: those computers in the middle of the scene appear to be selected more often than those in the outer parts in Figure 3.22 (a). Their paths in Figure 3.22 (b) show that the center of the scene was more transited than the exterior.

It can be observed in Figure 3.22 (a) that selections with the Cardboard interface describe circles around selections with the Oculus interface. The reason for this behavior revolves around the design of the interfaces: players with the Oculus interface need to move next to the computers to select them with a gesture, while with the Cardboard interface they could not move arbitrarily and had to teleport by looking at the computers. The selections of that interface are drawn on the destination of that teleport, which is 1.5 m away from the actual computer, in the direction of the origin of the movement.

As it can be observed in Figure 3.22 (b), users of the Oculus interface explored more the upper part of the scene, while those who used the Cardboard interface explored more its center, which affected the frequency of computer selections according to their location. The exploration with the Oculus interface can be explained in the setup of the experiment: since participants were explained how the system worked in front of a table with the screens on it, they tended to look and move towards that direction, resulting in a higher exploration and selection rate in that part of the scene. The same did not happen with the Cardboard interface because participants joined the experiment in group, so they interacted among themselves while they were preparing their devices, resulting in random directions. The difference between the exploration and selection between the middle of the scene and its outer

parts with the Cardboard interface is also explained with the design of the interface: since users started from the middle of the scene and they needed to select computers to move around, it is reasonable that those computers in the middle are selected with a higher rate than those in the outer parts, as participants needed to perform a series of jumps from computer to computer to reach those farther away from the middle. We can also observe that the movements in Figure 3.22 (b) follow narrower paths between computers with the Cardboard interface, while they are less directed with the Oculus interface, as participants could roam freely in the 3D scene.

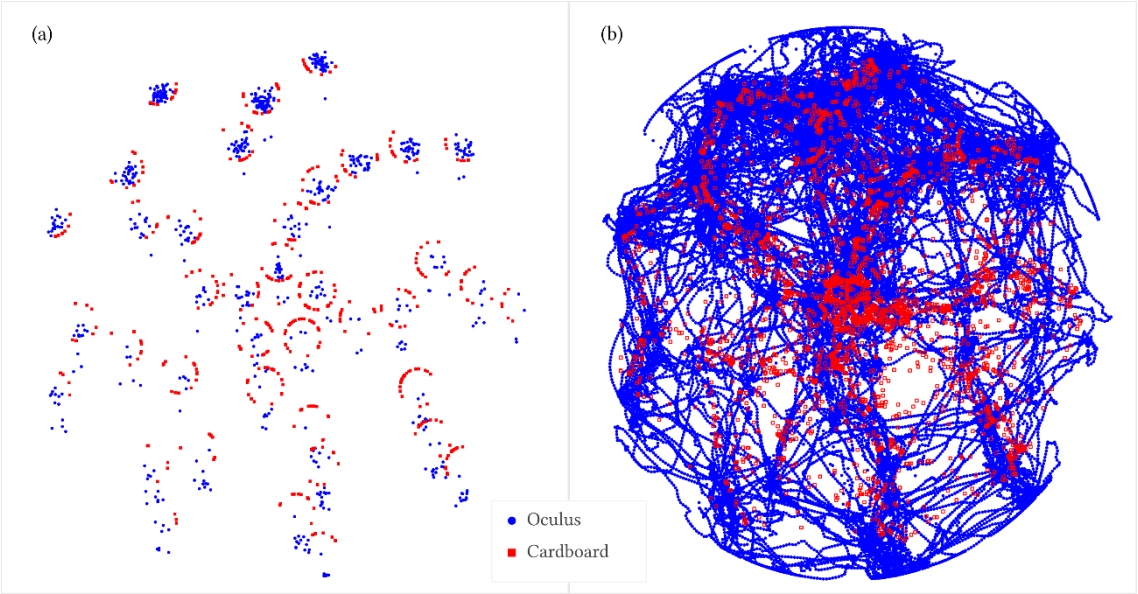

Figure 3.22. Spatial representation of the data collected in the virtual reality museum, grouped by the device which was used: (a) computer selections; (b) paths followed by participants.

These initial findings open the door to a more in-depth analysis of this dataset, which is performed and reported in Chapter 4.

3.6. A multi-device 360-degree hypervideo tour

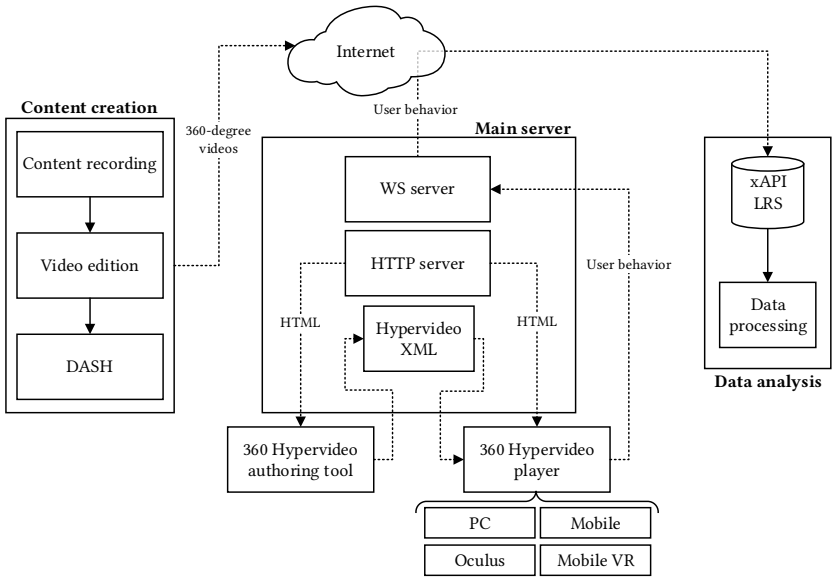

Figure 3.23. Architecture of the multi-device 360-degree hypervideo platform.

As a final step in the interactive multimedia platform, we designed TIM360, a system to create, visualize and analyze 360-degree hypervideos. Following Figure 3.23, 360-degree content is recorded using specialized equipment, such as an array of video cameras or cameras with wide lenses that already produce omnidirectional content in the equirectangular projection. These multimedia files are

then processed with traditional video editing software to, for example, cut them in suitable scenes. Multiple representations of each 360-degree video are encoded with different qualities in terms of bitrate and resolution, which are used to generate a Dynamic Adaptive Streaming over HTTP (MPEG DASH) manifest so the player can dynamically choose the best quality when playing the videos. All these files are uploaded to the Internet.

The 360-degree hypervideo authoring tool is a web application that is hosted in the main server, which provides an interactive interface to create and save a definition of the hypervideo in an XML file. The 360-degree hypervideo player, also a web application hosted in the main server, displays the interactive production following its definition in the XML file. The users' behavior when playing the 360-degree videos is submitted to the main server via a WebSocket connection, which formats the data as xAPI statements and posts them to the Learning Record Store (LRS), the element in the xAPI architecture that stores the data, so it can later be queried by our data processing tools.

3.6.1. The multimedia contents

A use case was designed to validate the system. A tour through the center of Palma was planned, consisting of routes through the streets and their intersections (see Figure 3.24). 35 intersections and points of interest were identified and recorded in 360-degree videos where the camera was static in front of that point, while a total of 22 routes between those were defined, which were also recorded in 360-degree videos, but this time the camera followed the street between two of these points. Note that each route was recorded in the two directions, resulting in a total of 44 route videos.

Figure 3.24 shows the distribution of the point videos (represented with marker icons and labeled starting with *p*) and the route videos (represented with thin black lines and labeled starting with *r*). Point videos are usually starting and ending points of route videos, but some are in the middle of a route (e.g.: *p2* in the middle of *r1* or *p4* in the middle of *r2*). Some connections between point videos had to be made (e.g.: between *p10* and *p18*), represented with thin red lines in Figure 3.24. Thick gray lines show the actual GPS data registered by the camera. Finally, the area enclosed in a green dotted polygon contains the subset of videos used to perform the study detailed in Section 3.6.6.

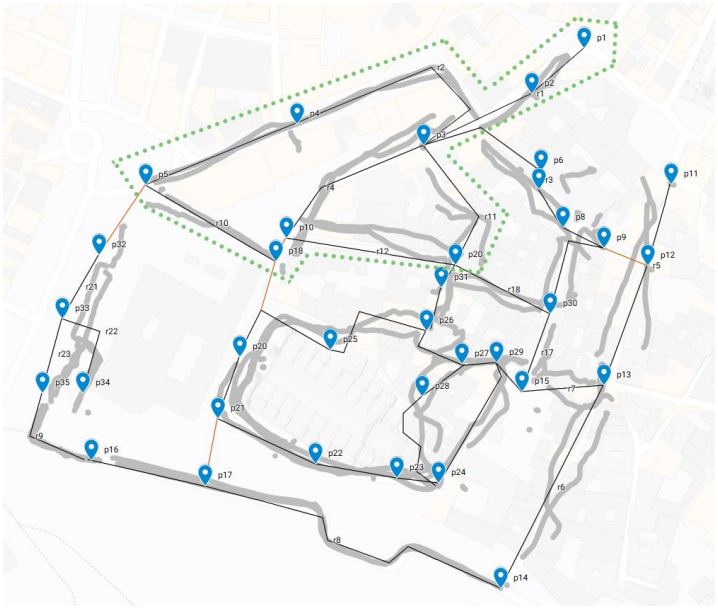

Figure 3.24. Distribution on a map of the 85 omnidirectional videos that were recorded for the multi-device 360-degree hypervideo tour.

These videos were recorded with a Garmin VIRB 360[17], which features two wide lenses and is capable to produce already stitched 360-degree video, with a final resolution of 3840×2160 pixels in the equirectangular format. Furthermore, additional metadata can be incorporated in a GPX (GPS

[17] https://www.garmin.com/virb

exchange format) file, such as the GPS coordinates of the camera. The multimedia content was edited and cut using Garmin's VIRB Edit, which also processes the metadata in the GPX file.

Table 3.5. Resolution and bitrate combinations for the videos in the multi-device 360-degree hypervideo tour.

	240p	**360p**	**540p**	**720p**	**1080p**	**2160p**
Resolution	426 × 240	640 × 360	960 × 540	1280 × 720	1920 × 1080	3840 × 2160
Bitrate (kbps)	300	500	1000	2000	5000	20000

Six representations of each video were generated with FFMPEG and libx264, with the combination of resolution and bitrates detailed in Table 3.5. Key frames were specified every second. These six representations were used to define the adaptation sets in the MPEG DASH manifests, generated with GPAC's MP4Box, with segments of one second.

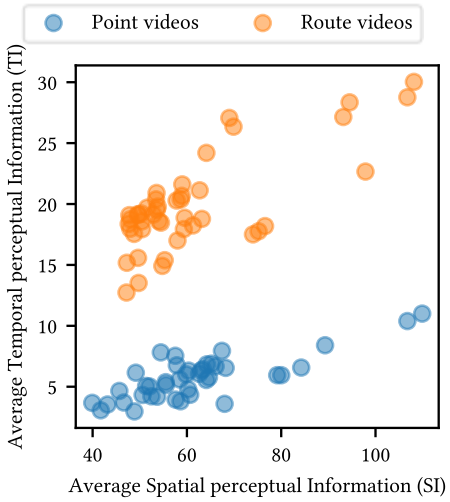

Figure 3.25. Plot of the SI and TI of the videos, grouped by video type, in the multi-device 360-degree hypervideo tour.

The Spatial perceptual Information (SI) and the Temporal perceptual Information (TI) measurements [103] were calculated for each of the 85 videos that were recorded, following the work described in [48]. The Spatial Information measures the amount of information in a given frame and the Temporal Information measures the difference in the image between successive frames. See in Figure 3.25 a plot of the average SI and TI values of all the videos grouped by video type. There appears to be two very distinct clusters of videos according to their video type.

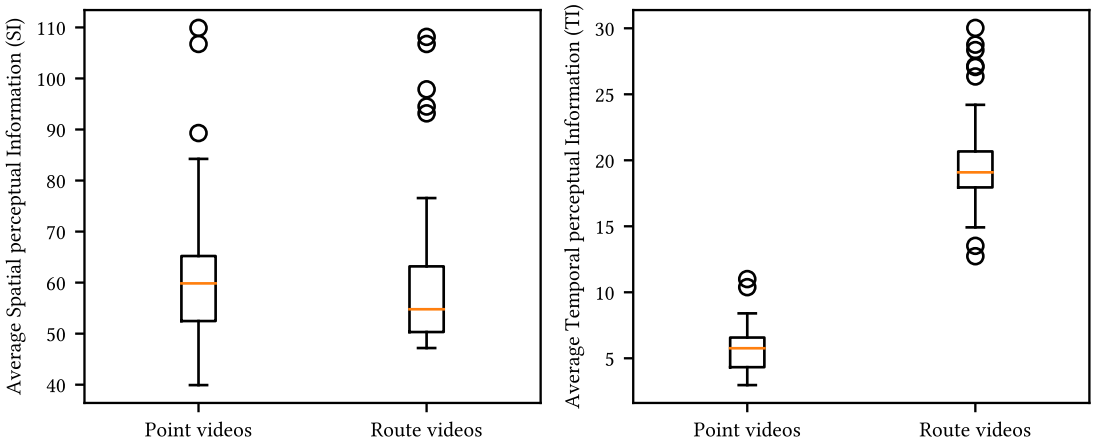

Figure 3.26. Comparison between the SI and TI values of the two types of videos in the multi-device 360-degree hypervideo tour.

The distribution of the average Spatial perceptual Information of both types of videos is similar (see Figure 3.26-left), with their averages almost identical: $\text{avg}(SI_p) = 61.91$, $\text{avg}(SI_r) = 61.11$). The distribution of the average Temporal perceptual Information exhibits a greater difference (see Figure 3.26-right), with a large difference between their averages: $\text{avg}(TI_p) = 5.73$, $\text{avg}(TI_r) = 19.83$. This can be explained because both video types depict similar images (the streets of the city of Palma), which results in a similar SI, but as point videos are static, the images do not change very much in time, whereas they do in route videos, resulting in a much higher TI measure. Given the difference between these two categories we can suspect that user behavior will also be different across point and route videos.

The manifests, the six representations of each video and the metadata in GPX files were uploaded to an HTTP server which makes them available through the Internet.

3.6.2. The hypervideo definition

A hypervideo project is specified by its XML definition (see Figure 3.27): a project p is identified by its URI, defined by its title and description, and contains a set of *videos* V_p. Each video $v_i \in V_p$ is identified by a unique name in p and defined by its src: the URL where the DASH manifest can be found for v_i. The attribute end is used to signal what video should be played once v_i ends, which can be the name of a video $v_j \in V_p$ or the keyword "loop", to specify that v_i should repeat indefinitely.

```
<project title="Title of the project" uri="https://my-project.net">
  <private>
    <owner email="my@email.net"/>
  </private>
  <description>This is an example project.</description>
  <settings>
    <setting key="default-video" value="v1"/>
    <setting key="use-tracker-url" value="https://wsserver.net"/>
    ...
  </settings>
  <videos>
    <video src="https://url.of/manifest.mpd" name="v1" end="v2">
      <markers>
        <marker name="1-to-2" link="v2" texture="texture_url" text="Label">
          <tracking>
            <track t="15" lat="0.00" lon="0.00"/>
            <track t="18" lat="0.00" lon="60.00"/>
          </tracking>
        </marker>
        <marker name="1-to-3" link="v3&t=10&o=90" texture="texture_url" text="Label">
          ...
        </marker>
        ...
      </markers>
    </video>
    <video src="https://url.of/manifest.mpd" name="v2" end="loop">
      ...
    </video>
    ...
  </videos>
</project>
```

Figure 3.27. Example of the XML definition of a hypervideo in the multi-device 360-degree hypervideo.

Each video $v_i \in V_p$ may contain a set of markers M_i, which represents the hyperlinks from video v_i to other videos $v_j \in V_p$. Each marker $m_{ij} \in M_i$ is identified by its name and defines what video $v_j \in V_p$ should be played when following this hyperlink according to the value of the attribute link, which is formatted as a URI query string: after the name of the target video the keys "t" and "o" may be used to indicate the starting point in the spacetime of the omnidirectional video: "t" specifies the starting point in time in seconds from the beginning of the video and "o" specifies the starting point in space in degrees of longitude from the origin; both values default to 0 if missing. Additionally, an URL to an image is specified in its texture property and a label can be displayed near this hyperlink by specifying its attribute text.

The markers, in contrast to hyperlinks in hypertext, are not displayed in static positions inside text, but in a position in 3D space, represented by spherical coordinates, as a function of time. Each marker m_{ij} contains a tracking list T_{ij} of key positions in the space-time of the 360-degree video as triples $t_k \in T_{ij} = \langle time, latitude, longitude \rangle$. The attributes of a track element t_k are time, lat and lon, which specify the elapsed time in seconds since the start of the video v_i when the marker m_{ij} takes the position defined by the spherical coordinates lat and lon in degrees of latitude and longitude from the origin. This origin lies in the center of the equirectangular frame, with latitude values $\phi \in [-90,90]$ from bottom to top and longitude values $\lambda \in (-180, 180]$ from left to right. A minimum of two tracks need to be specified for a given marker, which would define the spatiotemporal positions it takes when it appears and when it disappears. Additional tracks add precision in the tracking of key positions over time for that marker.

Other information present in the definition of the hypervideo is the declaration of the creator of the project, used to enable them to edit the project in the authoring tool; and a list of setting $s_i \in S$ which contain two attributes: key and value, which can be used by the authoring tool and the player application, such as the default video $v_d \in V_p$ that is played when the project is loaded (if none is specified a random $v_r \in V_p$ will play) or to define the WebSocket connection to track the user behavior when playing the project.

3.6.3. The authoring tool

A tool was created to assist the editing of the XML hypervideo definition. Implemented as a web application, it is hosted in the HTTP server (see Figure 3.23) and it provides an interface to create hypervideo projects and edit their definition. Given the set of hypervideo projects P, the user u can edit the hypervideo projects $p_i \in P$ that match $owner(p_i) = email(u)$. The application is a single page web application made with React[18] and React-bootstrap[19], that communicates with the application server to retrieve and store the relevant data (most notably, the project's XML definition) using an HTTP API.

An interactive 360-degree video editor is included in the authoring tool (see Figure 3.28) to assist the placement of hyperlinks in a 360-degree video, based on the same A-Frame implementation as the player application that will be explained next. Through this tool, each key position in the tracking can be defined and visualized at the moment, with final positions between key positions calculated via linear interpolation spherical coordinates.

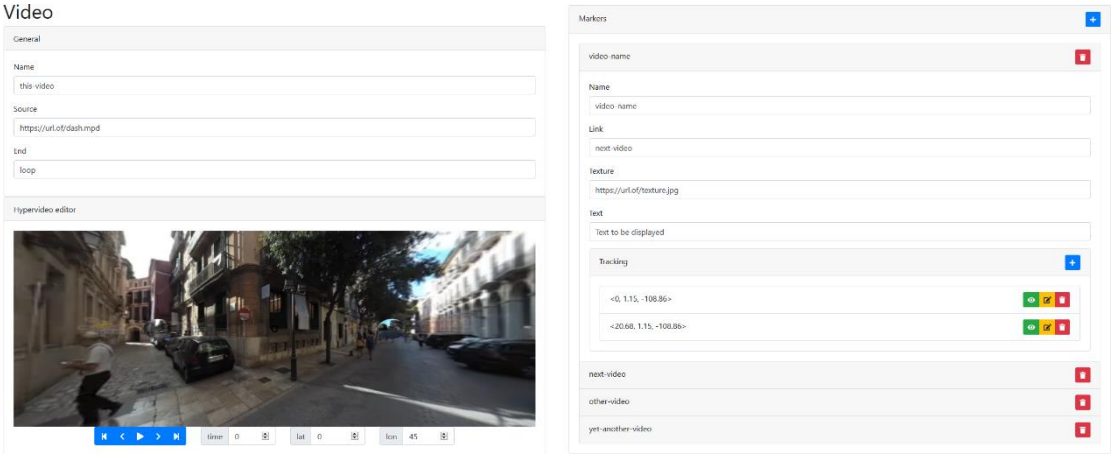

Figure 3.28. The authoring tool in the multi-device 360-degree hypervideo, with the interactive editor on the left and the details of the tracking of a marker on the right.

When the author has finished editing, they can submit the modified XML definition to the server.

[18] https://reactjs.org/
[19] https://react-bootstrap.github.io/

3.6.4. The player application

The player is also a web application that is hosted in the HTTP server. It is accessed by specifying the name of the project in the URL and, optionally, the name of a starting video $v_0 \in V_p$, e.g.: https://ltim.uib.es/360/palma#cort. If a correct video name is specified this way, it will load as the first video; if not, the behavior defined by the hypervideo definition will be followed.

The XML specification is loaded by the web application, which displays the first 360-degree video v_0 in a 3D scene, along with the markers that represent the hyperlinks from to other videos $v_j \in V_p$.

The user can perform a limited set of actions: choose their current viewport, so they can view different parts of the 360-degree video; and follow a hyperlink, represented by a marker in the scene.

The scene can be interacted with multiple devices, which we have grouped in four categories: computers with a mouse (*pc*), smartphone devices (*mb*), VR visors designed to be used with smartphones, such as the *Google Cardboard* (*mv*) and desktop VR headsets, such as the *Oculus Rift*, with hand controllers (*oc*).

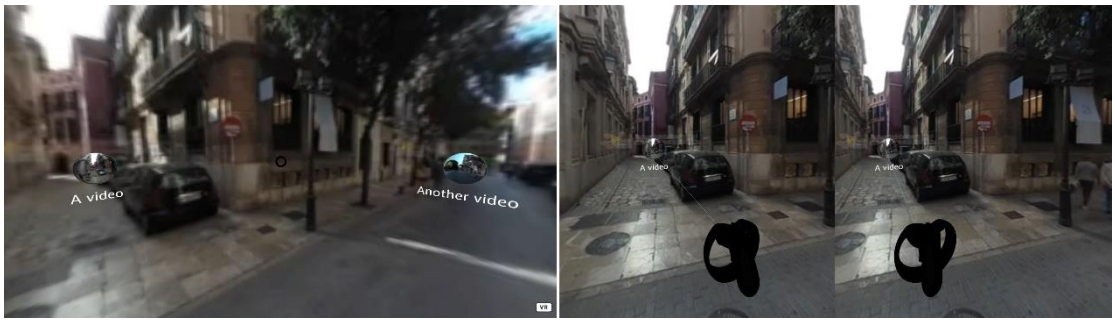

Figure 3.29. The player application of the multi-device 360-degree hypervideo. On the left, the interface as it is shown in the desktop PC interface and on the right as it is shown in the Oculus interface.

When the scene is displayed in a computer with a mouse (*pc*), it is rendered in the computer screen, so the immersion is limited. The user can choose their viewport y moving the mouse (the system pointer is locked, so there is no need to continuously drag the scene) and can follow a hyperlink by clicking a marker in the scene with the aid of a cursor that is shown in the center of the screen.

The interface is similar when a mobile device is used (*mb*): the user can look around by dragging the touch screen and can follow a hyperlink by tapping the screen when the cursor in the middle of the screen is over a marker.

VR headsets introduce immersion to the player application. Before wearing one of these devices, the VR mode has to be enabled by clicking the *enter VR button*. Mobile VR headsets (*mv*) are designed to have the users slide their smartphones inside them and display the scene through a pair of lenses in it. The user can change their viewport by turning their head around, thanks to the sensors in their smartphone. Now that the touch screen of the device is inaccessible, it cannot be tapped in order to follow a hyperlink. In this case, marker selections are performed by centering the cursor on a marker during a dwell time [104] of 2 seconds.

Finally, dedicated VR headsets often include hand controllers. When the application is accessed using one of these (*oc*), the virtual cursor is no longer displayed, and the selection of a marker is performed by aligning a laser line that is drawn from the virtual controllers to that marker and pressing a button on the controller. The viewport can be selected by the user by turning their head, similarly to the previous case.

Figure 3.29-left shows the scene from a *pc* device, with one marker at each side, the cursor (a black circle) in the middle, and the *enter VR* button on the bottom left of the screen; Figure 3.29-right shows the scene as it is seen from an *oc* device, with half of the screen for each eye in the VR headset and a hand controller visible at the bottom of the scene, with the laser line aligned with the marker. To aid the selection of these hyperlinks, when the user is about to do so, the playback of the 360-degree video automatically pauses, so they do not move away from the cursor or the laser line.

The A-Frame WebXR framework follows a modular architecture in the form of components. The hypervideo component was created to display a given 360-degree hypervideo project p both in the player application and also in the authoring tool, which is specified by either the URL of the XML definition of the hypervideo project (when it is used in the player application) or the XML DOM itself (when it is used in the authoring tool). This component plays the required videos $v_i \in V_p$ through the dash component, created to control the reproduction of the MPEG DASH manifests with dash.js[20]; sets up the markers $m_{ij} \in M_i$ using the marker component (which displays a marker m_{ij} according to the tracking defined by $t_k \in T_{ij}$); handles marker selection, enabling navigation through the hyperlinks; plays the next video when v_i has finished (if it does not loop); exposes the functionality required for the authoring tool to play and pause the reproduction of the video and to jump to a specific spatiotemporal position; and finally tracks the users' actions and submits them to the WebSocket server (see Figure 3.23) using the Socket.IO[21] library.

Table 3.6. Summary of the events in the multi-device 360-degree hypervideo and the significant parts of the generated xAPI statements.

Event	Verb	Object	Context	Result
The application loads	*logged-in*	*virtual-reality*	*accept-language, referrer, user-agent*	–
A video is loaded	*initialized*	*video*	*display, session-id*	–
A video is played	*played*	*video*	*latitude, longitude, session-id*	*time*
A video is paused	*paused*	*video*	*latitude, longitude, session-id*	*time*
A video ends	*completed*	*video*	*latitude, longitude, session-id*	*time*
The orientation changes	*interacted*	*video*	*latitude, longitude, time, session-id*	–
The player is closed	*abandoned*	*virtual-reality*	*session-id*	–

The usage tracking is performed through xAPI statements (see Table 3.6 for the details of the events and statements, and Figure 3.30 for the hierarchy of statements according to their *session-id* extension). We used the general xAPI vocabulary registry [96] and ADL's video vocabulary [97], as well as some of the vocabulary we introduced in the past sections. We will now define the new vocabulary and so we can referrer it with the last part of the URI in the rest of this section. To record some of the data from the HTTP headers in the logged-in statement, the context extension *https://ltim.uib.es/xapi/extensions/accept-language* was included together with *referrer* and *user-agent*. The *initialized* statements feature the context extension *https://ltim.uib.es/xapi/extensions/display*, which is an object that contains whether the device is a mobile device, whether a headset has been connected and whether the VR mode is currently enabled, so we can infer which of the four interfaces is being used to play that video.

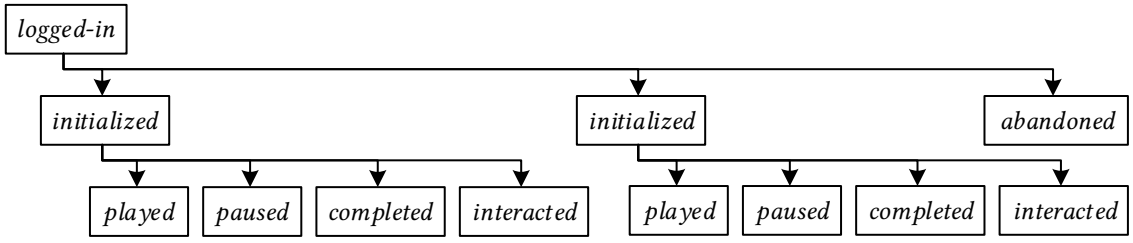

Figure 3.30. Hierarchy of xAPI statements by their verb in the multi-device 360-degree hypervideo.

After a user loads the application (and connects to the WebSocket server) an xAPI statement is stored with the verb *logged-in*, the URI of the hypervideo project as the id of the activity (of type *virtual-reality*) and the contents of the HTTP headers *accept-language*, *referrer,* and *user-agent* as contextual information. When a 360-degree video is loaded, a statement with the verb *initialized* is generated, with the src attribute of that video as the id of the activity (of type *video*), with the following contextual information: which device is being used in the *display* extension and the id of the *logged-in* statement as the session-id. If a video starts playing (or resumes), is paused or ends, *played*, *paused* and *completed*

[20] https://github.com/Dash-Industry-Forum/dash.js/
[21] https://socket.io/

xAPI statements are posted to the LRS. The id of the parent *initialized* statement is used as *session-id* and the *latitude* and *longitude* context extensions contain the current orientation of the viewport of the user. The current *time* of the video is included as a context extension, as recommended by the vocabulary [97]. When the user changes the orientation of its viewport (throttled four times per second), a statement with the *interacted* verb is posted, with the orientation of the new viewport in its *latitude* and *longitude* extensions and the id of the parent *initialized* statement is used as its *session-id*. Finally, after the connection with the server is lost, a statement with the *abandoned* verb is posted to the LRS to signal the end of the session.

3.6.5. The user experiment

To validate the proposed solution, a user experiment was performed in the University of the Balearic Islands. A sample tour was prepared with 20 videos of the 360-degree hypervideo production (those inside the green dotted area in Figure 3.24): 8 point videos and 12 route videos between them (six routes in both directions), using the four interfaces we described before.

30 users were accepted in groups of maximum 4 to be given a brief explanation of how the overall system and each of the interfaces work. Each user ran the experiment with all the interfaces during a slot of time in a random order to counterbalance learning effects while their actions were automatically recorded by the usage tracker. Participants were asked to verbalize their thoughts following the *think-aloud* technique [105] and, once they finished, they were asked to fill the System Usability Scale questionnaire [106] to quantify the usability of the system.

In the first interface, *Desktop computer* (*pc*), a standard PC equipped with a mouse was available for the subjects to open the web application and start browsing the hypervideo. They were instructed to enter in full screen mode and to lock their cursor, so they did not have the need to continuously drag with the mouse to orient their viewport center. In general, this interface was well received, as most users felt comfortable with hardware they already understood for a task they could link it with, even if it was not immersive.

In the second interface, *Smartphone* (*mb*), users were asked if they wanted to try the experiment with their own device, which was well received, although it could only be done if they owned a smartphone compatible with WebGL (if they did not, they were provided a suitable Smartphone for this task); furthermore, some users wrongfully selected the *Enter VR* button, so they had to restart their session, increasing the total number of sessions recorded in Table 3.7. Participants particularly disliked this interface because they found it cumbersome to align the cursor with the hyperlinks to navigate between videos.

A similar course of action was performed in the third interface, *Mobile headset* (*mv*): users who wanted to use their devices could do that or they could use a provided Smartphone. In any case they would slide the device inside a *Google Cardboard* after enabling the VR mode by selecting the *Enter VR* button. Users found this interface more intuitive and a good enough substitute for a fully-fledged VR headset.

Finally, in the fourth interface, *Oculus headset* (*oc*), participants were asked to wear an Oculus Rift headset and the Touch controllers, which were powered by a PC. Although most of them had never used VR before, they learned how to use it and were able to enjoy this experience.

3.6.6. Results

Users observed that the adaptive streaming was not working very well, resulting in deficient image quality that was most evident when using the Oculus headset and least when using the small screen in mobile devices. This makes evident that adaptive streaming technique needs to be adjusted to work with 360-degree videos, or that it may be combined with other dynamic streaming techniques already explored by the literature, such as tile-based streaming.

A higher degree of immersion is achieved using headsets: there is a route video that follows stairs and users expressed their fear to fall when going down the steps. Some users observed that they felt as if they were flying, and this is natural, since when the videos were recorded, the camera was placed on a stick on top of a helmet the camera operator was wearing. Since most markers are placed on the beginning of streets, some users expressed their desire to explore through streets that did not have a

marker. Curiously, most users performed the experiment standing, but others preferred to sit on a wheeled chair (except when using the desktop computer, in which all users sat down).

Although it is the most immersive interface, the use of the Oculus headset needs preparation: the headset itself needs to be adequately placed on the users' head and the user needs to familiarize with the controls. The mobile headset provides a similar experience as the Oculus headset but was considered a poor version of the real thing. Users felt very confident when using the non-immersive interfaces, as they were already used to interact with desktop computers and smartphones and just had to learn how to navigate this new kind of media.

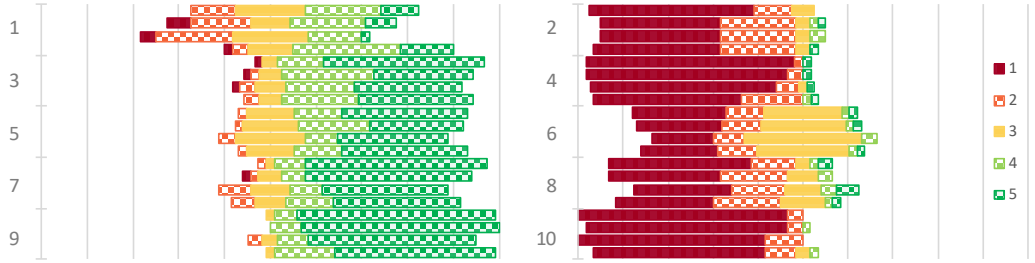

Figure 3.31. Diverging stacked bar chart of the answers given in the SUS questionnaire. Positive questions (odd-numbered) are shown on the left and negative questions (even-numbered) are shown on the right. Answers in each question refer to the use of the four interfaces in this order: desktop computer, smartphone, mobile VR headset and Oculus VR headset.

The SUS scores per interface are as follows: $SUS_{pc} = 85.97$, $SUS_{mb} = 81.76$, $SUS_{mv} = 77.29$ and $SUS_{oc} = 81.39$. Although all these scores are in the acceptable range [107], they suggest that usability was highest when using a desktop computer (*pc*) and lowest when using a mobile VR headset (*mv*). While users liked the novelty in the immersive devices (*oc*, *mv*), the usability in interfaces that were familiar to them (*pc*, *mb*) was superior. The question with lowest score was #1: "I think that I would like to use this system frequently", with $\text{avg}(Q_1) = 3.31$ and $\text{std}(Q_1) = 1.04$, while the question with the highest score was #10: "I needed to learn a lot of things before I could get going with this system", with $\text{avg}(Q_{10}) = 4.81$ and $\text{std}(Q_{10}) = 0.49$, hinting the simplicity of this system.

After the user experiment was performed, the data which was collected could be imported from the LRS through the analysis module and exported into csv files which contained the relevant information to perform further analysis.

Table 3.7. Number of user and video sessions by device in the experiment of the multi-device 360-degree hypervideo.

	pc	*mb*	*mv*	*oc*
User sessions	33	41	36	30
Video sessions	255	205	214	315

A total of 140 user sessions were recorded by the usage tracker, with a total of 928 video sessions, distributed per device as shown in Table 3.7. The video sessions in which $p2$ and $p4$ were played (the two point videos which were not the end of any route but were in the middle of them) were excluded, as they were reproduced very few times.

Table 3.8. Number of samples per video type and device in the experiment of the multi-device 360-degree hypervideo.

Video type	*pc*	*mb*	*mv*	*oc*	**Total**
Point	2556	3242	3589	5290	14677
Route	10057	7689	8405	15456	41607
Total	12613	10931	11994	20746	56284

The distribution of changes in orientation extracted from the LRS was resampled into a distribution of viewport centers by interpolation on the great-circle coordinates (3.2) in fixed intervals of 250 ms

to a) make all the samples be on the same temporal values and b) obtain intermediate orientations when the user did not change it for more than 250 ms. This way, a total of 56284 samples were obtained from the 928 video sessions, distributed by video type and device following Table 3.8.

The distribution of viewport centers is shown as a heatmap in Figure 3.32, with all the devices, grouped by video type: point videos (a) and route videos (b). We can observe that in point videos, the viewport centers are more distributed along the longitude (x axis) than in route videos.

Centroids and covariances of these viewport centers were calculated (longitudes were normalized into the range (−180,180]) for every type (p denotes point videos and r route videos): $\overline{x_p} = (-10.69, 6.25)$, $\overline{x_r} = (-9.38, -3.44)$. We can observe that the variance in latitude is similar in both types of videos, but the variance in longitude is much larger in point videos (a) than in route videos (b).

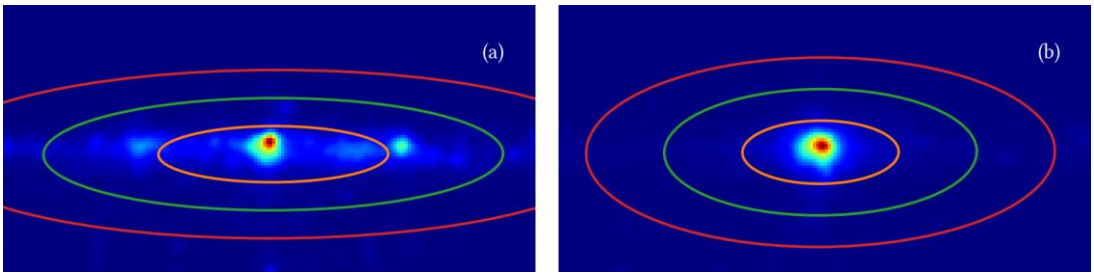

Figure 3.32. Heatmap of the viewport centers of all the reproductions in the experiment of the multi-device 360-degree hypervideo grouped by point videos (a) and route videos (b). Concentric ellipses represent the covariances at 1, 2, 3 σ. Built with 180 × 90 bins and smoothed with a gaussian filter $\sigma = 1.5$.

Two measures of dispersion between the samples and their centroids were calculated: s' based on the Mean Absolute Error (3.4) and s'' based on the standard deviation (3.5). Without differentiating between devices, the first measure reports $s'_p = 62.16$ and $s'_r = 35.76$ for point and route videos, respectively, while the second reports $s''_p = 75.20$ and $s''_r = 52.17$, showing a great difference between video types.

$$s' = \frac{1}{n}\sum_{i=1}^{n}|d(x_i, \bar{x})| \tag{3.4}$$

$$s'' = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d(x_i, \bar{x}))^2} \tag{3.5}$$

When devices are taken into consideration, we can obtain these measures for each combination of video type and device, which are reported in Table 3.9. We can observe how there are small differences between devices, especially in s'.

Table 3.9. Dispersion of the distribution of viewport centers per video type and device in the experiment of the multi-device 360-degree hypervideo.

	s'				s''			
Video type	***pc***	***mb***	***mv***	***oc***	***pc***	***mb***	***mv***	***oc***
point	65.40	67.84	53.74	62.65	76.11	76.32	74.34	75.00
route	333.25	34.58	34.83	37.97	52.50	52.80	52.23	51.93

As observed in prior work, the centroid of the viewports falls near the origin of coordinates. This is influenced by two variables: a) the videos start looking at that direction, b) the camera in route videos follows that direction. From this data, user behavior on point videos (with a static camera and lower TI) show greater dispersion than on route videos (with a moving camera and higher TI), in line with [47], reinforcing the idea that users tend to look around more often in point videos than in route videos. There is not an evident difference by the device used to play the videos, as both measures are very similar across devices; further analysis will be performed in this regard, which will be reported in Chapter 5.

3.7. Conclusion

In this chapter, we detailed we evolution of the interactive multimedia platform that allowed us to play interactive experiences with 360-degree videos and virtual reality and register the user's actions with the Experience API. We evolved from a simple drag interface designed for computers in which 360-degree videos were played to a fully immersive player compatible with VR headsets.

In the first iteration (An interactive 360-degree video activity), we found that the Experience API is a suitable medium to record users' behavior on interactive learning activities and that it may be extended to non-educative activities without much complication. The proposed solution provides the required data to perform analyses on the users' behavior on interactive multimedia, with the benefit of a very modular approach: the entities at either side of the xAPI paradigm (the Learning Record Provider, the player module and the Learning Record Consumer, the analysis module) could be replaced without needing to change how the other works.

The selection process worked well in most cases and served its purpose to test the system, but there were some edge cases that require further improvement. First, the decision to not showing any kind of visual aid (as markers) so as not to make users aware they may be required to select these or not resulted in inaccurate selections if the object appeared larger in the video than the selection window that we considered to register a selection. This could be solved by configurable per-object window sizes or with polygonal-based trajectories, that may be helped by computer vision on the 360-degree video. Second, plane linear interpolation was used to obtain the spherical coordinates in a specific time between two key positions. This is inaccurate and, if the object were to cross the line $\lambda = \pm 180^{o}$, erroneous, as with this approach it would travel all the longitudes from one side to the other, which would be improved in future iterations.

The player application, despite it is a web application, was designed to be interacted with a computer screen and mouse. Although it would launch in a smartphone device, it would not be usable enough, so this would be addressed in future iterations, as mobile browsing of the Internet is leading, and these kinds of devices are also used in some educational centers. Immersive hardware such as virtual reality headsets would be a natural choice to play an immersive experience such as these based on 360-degree videos, with further challenges in the user interface and its usability, which would also be addressed in future iterations.

The second iteration of the interactive multimedia platform (A 360-degree hypervideo documentary) proved useful to display interactive hypervideos. This time, to address video quality constraints, the adaptive technique MPEG DASH was introduced to the main 360-degree video and the associated linear videos. To properly display markers in their actual location, the interpolation algorithm had to be adapted to work with great-circle formulas.

We have demonstrated that the player system and the xAPI model can be used to collect the required data to perform rather complex user analysis, from which we have learnt that markers in the beginning are selected more often than the rest, that associated media is consumed thoroughly, that registered users perform better than unregistered users in main video consumption (time and space-wise) and that the behavior between the selection of markers and exploration in time or space is better correlated in registered users. There appears to have been a difference in engagement between these two user classes, as registered users showed a more active behavior than unregistered users.

Player issues from the first iteration remained in the second, most notably the lack of multi-device support (smartphones and immersive devices), which would be addressed in the next iteration.

The third iteration of the interactive multimedia platform (A multi-device virtual reality museum) featured the support of multiple devices and the introduction of specialized user interfaces for them. We leveraged A-Frame to implement those interfaces and take advantage of the immersive hardware they support. The Experience API protocol we had been using for 360-degree videos could be adapted for a pure virtual reality experience.

We noticed some usability issues during the study: participants showed difficulties to perform the pointing gesture with the Oculus interface (Figure 3.18), as it required to place their thumbs on the

thumbrest (which is a touch sensor) and participants did not understand the need to do so if there was no button there. After the study, we are certain that this gesture should be more flexible.

Slightly related to this, the computers on the scene are placed on the plane $y = 1.5\ m$. Since interfaces with three degrees of freedom (all except desktop VR devices like the Oculus) lack positional sensors, this makes those users see the virtual environment from that height as well, but with six degrees of freedom (like with the Oculus interface), one can move in all directions, also vertically. This was an issue with the youngest participants, because with their reduced height they struggled to reach the virtual objects to touch them. This makes evident that the age of the users is a factor that needs to be considered when designing immersive experiences.

We also noticed that many participants, who were inexperienced with virtual reality applications, were confused by the lack of a precise objective or expressed their concern of not seeing the real world anymore, which distressed them. This complicated the task of selecting a computer, as they were rigid and did not move their limbs as needed. In contrast, participants were more relaxed with the Cardboard interface. As they performed the experiment in group, it became some sort of a social experience, they talked among themselves, and they even tried to find each other in the virtual world (which was impossible). They easily understood they needed to select a computer to move next to it, but they also tried to physically move, but as the Cardboard interface lacks positional sensors, that did not translate into a movement in the virtual world.

Finally, as we have discussed before, we noticed some differences in behavior between interfaces. We observe that differences in the design of the interface produce differences in the users' behavior, which will be further addressed in Chapter 4.

The final iteration of the interactive multimedia platform (A multi-device 360-degree hypervideo tour) consolidates the advancements in the introduction of several devices into the reproduction of 360-degree videos: user behavior in 360-degree videos can now be automatically registered from computers, smartphones, VR headsets and mobile VR visors. The definition of hypervideos with the creation module we introduced proves to be useful in generating interactively linked 360-degree videos, as validated by the experiment we described.

We conducted a formal usability study for each of the four devices, with different results but all in the acceptable range. Users disliked that the video quality changed often, due to adaptation algorithm of MPEG DASH.

We noticed technical differences in the 360-degree videos: point videos, in which the camera does not move, show a lower TI than route videos, in which the camera follows that route. Per the usability study, we learned that not all the devices were experienced in the same way. For these reasons, the users' behavior may be different according to these two variables.

From the data collected during the study, we conducted a preliminary analysis and observed the distribution of the orientation of the users in the 360-degree videos. We noticed an evident difference between both types of videos: users are much more disperse in point videos than in route videos. We observe that the direction in which the camera is moving (following the street in route videos) conditions the preferred orientation of the users. The difference in the dispersion between devices is not so evident. There are small differences, but we cannot arrive at any conclusion in this preliminary study. For this reason, a more in-depth analysis will be carried in Chapter 5.

Furthermore, we will study a number of techniques to predict future viewports in Chapter 6, which are necessary to introduce tile-based adaptive streaming, which addresses the difficulties of streaming high-quality spherical content.

To conclude, we have seen that the users' actions when they play spherical videos and virtual reality are worthy of study. We analyzed the spatiotemporal exploration in the interactive 360-degree video activity; the relationship between the selection of points of interest and the visualization of the spherical video in the 360-degree hypervideo documentary; the difference by device in spatial exploration and selection of objects in the multi-device virtual reality museum; and the difference by video type and device in the dispersion of the users' orientations in the multi-device 360-degree hypervideo tour.

The user data recorded in the experiments of these last two iterations was used to perform more in-depth studies which are reported in the following chapters: an analysis of the user behavior in multi-device virtual reality is reported in Chapter 4; an analysis of the user behavior in multi-device 360-degree hypervideo is reported in Chapter 5; and finally a study of classification and prediction of future orientations based on this last case is reported in Chapter 6.

Chapter 4. Analysis of user behavior in a multi-device virtual museum

In this chapter we study in more detail the user behavior in the multi-device virtual reality museum reported in section 3.4. In section 3.5.3 we detailed a user experiment in which we collected the user behavior in two of the four available devices: a desktop VR headset, the *oculus* interface; and a mobile VR headset, the *cardboard* interface. The non-immersive devices (desktop PC and mobile) are excluded from this analysis as they were not used in the experiment.

As previously introduced in Chapter 2, prior studies on user behavior in virtual reality environments consider a single device in their experiments. In order to advance the actual state of the art, we compare the users' behavior with two immersive devices. To evaluate the difference in behavior between these two devices, we are interested in the differences in interaction with the virtual objects, in the movement and in the orientation. We are also interested to assess whether there is difference in behavior in the *cardboard* interface when different web browsers are used to access it.

For this, we formulate the following research questions, which we will answer in this chapter.

RQ1. Is there a difference in the time between interactions with the virtual objects?
RQ2. Are the virtual objects interacted in the same magnitude?
RQ3. Do users navigate the virtual space in the same way?
RQ4. Do users look around, turning their heads, similarly?
RQ5. When the *cardboard* interface is used, is the behavior uniform across different web browsers?

In Table 4.1, we can grasp a first approach at analyzing the data obtained in the user study: we recorded 244 mobile sessions that used the "Enter VR" button at some point, so the mobile headset (*cardboard* from now on) was enabled. Sessions in which the user did not use the "Enter VR" are not counted in the following graphs as *cardboard* sessions.

Table 4.1. Summary of the operating system and web browser used to access the *cardboard* interface.

OS	Chrome	Firefox	MiuiBrowser	Safari	Samsung Internet	Total
Android	134	1	3	0	10	148
iPhone	32	0	0	64	0	96

244 sessions were registered during the case study that used a mobile headset, of which 175 interacted with at least a computer. Due to the overwhelming success of the *oculus* interface, individual sessions per user could not be recorded and the data is aggregated. In total, computers were selected 967 times from the mobile interface and 708 times from the *oculus* interface. A single user was expected to use only one of the two devices, so no learning effects should not have happened. Note that successive plays of the same computer if its associated video did not completely play are not counted, as some users turned the same computer on and off repeatedly.

4.1. Time between interactions with the virtual objects

In this section, we analyze the results and the difference between devices in the selection and interaction of the virtual objects.

Following Figure 4.1, there were many *cardboard* sessions which were very short or in which the user did not select any computer: 31 (13% of the sessions) ended in less than 1 minute and 69 (28% of the sessions) ended without selecting any computer. The mean duration of a *cardboard* session is :03:39, selecting 3 different computers. It can be observed that the behavior in some sessions is way more intense than others, with session durations over 5 minutes and selecting over 10 computers.

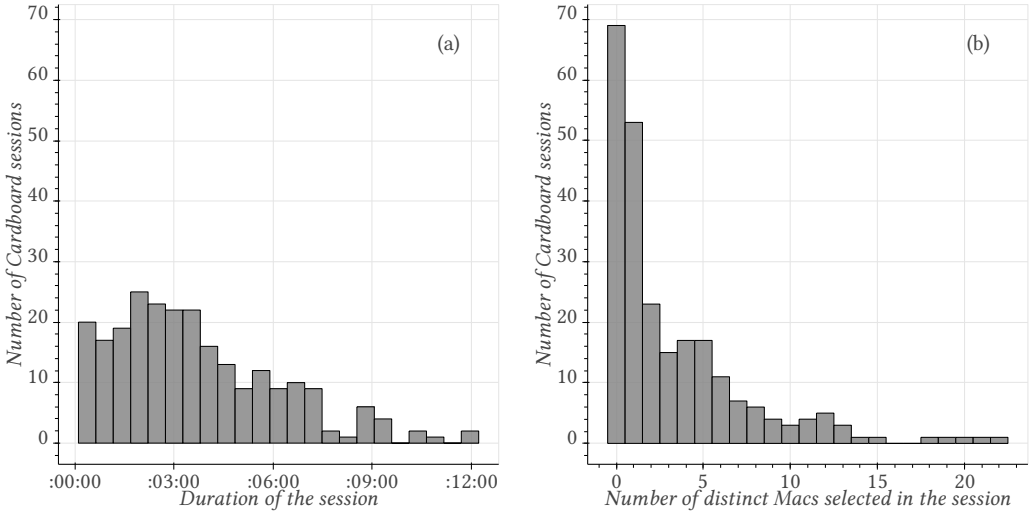

Figure 4.1. Histogram of the duration of the 244 *cardboard* sessions (a) and histogram of the number of distinct computers selected in them (b).

We can observe in Figure 4.2 the amount of time between two consecutive selections (without counting consecutive selections of the same element) in the same session. The trend in both interfaces is similar and the mean time between selections is :00:36 for *cardboard* sessions and :01:04 for *oculus* sessions, showing a rapid drop for greater times.

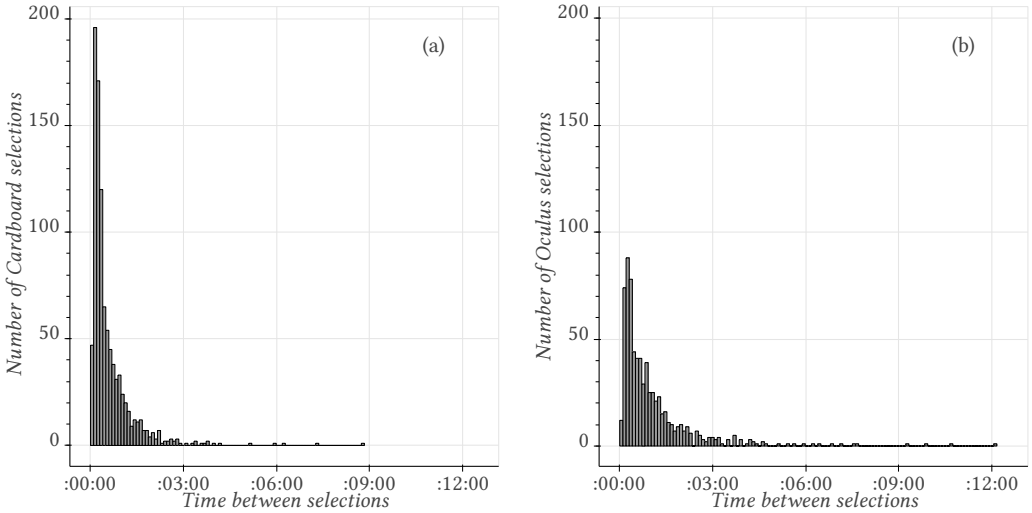

Figure 4.2. Histogram of time between consecutive selections in *cardboard* (a) and *oculus* (b) sessions.

Of the 1675 selections in the dataset, we removed the 5% with less and the 5% with more time between selections to eliminate outliers, so the time between sessions lies between 6 and 155.3 seconds, with a total of $N = 1504$ selections (868 in *cardboard* and 636 in *oculus*).

To answer RQ1 (Is there a difference in the time between interactions with the virtual objects?), we set the null hypothesis H_0 that the mean time between selections in both groups is the same. Levene's test for homogeneity of variances reports unequal variances between groups ($F = 26.81$, $p < 0.001$), so a Welch's t-test was performed on the time between selections in *cardboard* ($N = 868$, $M = 33.720$, $SD = 28.908$) and *oculus* ($N = 636$, $M = 45.993$, $SD = 34.542$). The result, $t(1218) = -7.29$, $p < 0.001$, with a Cohen's $d = -0.391$, rejects the null hypothesis, so we can accept the alternate hypothesis H_1 that the average time between selections is different in both groups for a confidence level $\alpha = 0.05$ with a small effect size.

4.2. Selection of the virtual objects

The interactions with the objects in the virtual museum are plotted in Figure 4.3, according to their position in the virtual environment. We can observe that the computers in the centre of the scene are the most selected in *cardboard*, while the most selected in *oculus* are those in the top of the scene. We can also appreciate a difference in selections between computer families and inside each computer family. In example, of the five central circles in Figure 4.3 (a), the top right one is the most selected, and objects in the middle appear to be more selected than those farther away in Figure 4.3 (a), while the opposite seems to be true in Figure 4.3 (b).

To answer RQ2 (Are the virtual objects interacted in the same magnitude?), we will test three groups of hypotheses. First, we will analyse if the selections are uniformly distributed between families (see Figure 3.19) and if they are comparable between devices; second, we will check if the selections are uniformly distributed per computer inside each family and if these are comparable between devices; third, we will correlate the placement of the object with the number of times it has been selected.

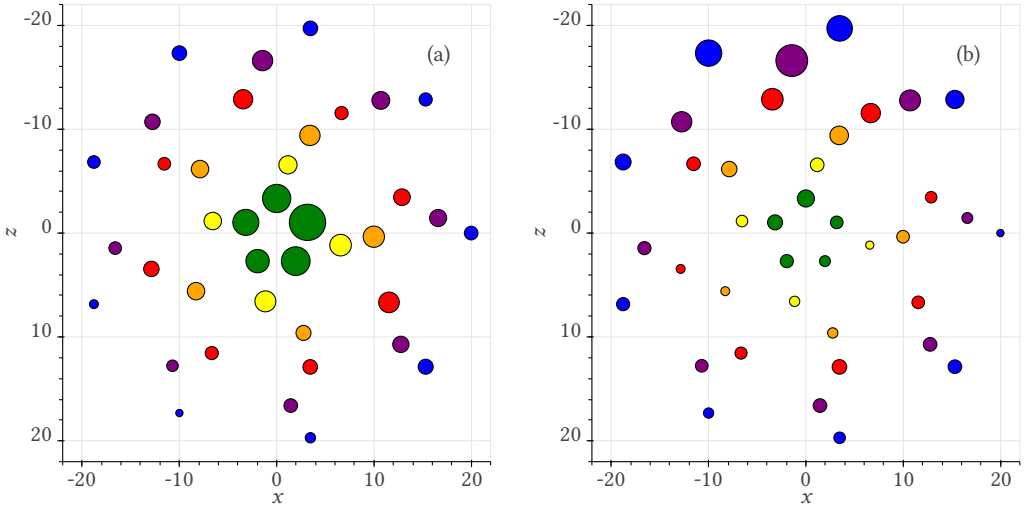

Figure 4.3. Spatial representation of the computer selection frequency in *cardboard* (a) and *oculus* (b) sessions.

4.2.1. Distribution of selections between computer families

In this case, we wanted to know, for each device, if the selections between computer families (see Figure 3.19) follow a uniform distribution and if they come from the same distribution. In this case, the selections were summed per family.

Chi-Square goodness of fit tests were performed for each device to test a) if the selections per family follow a uniform distribution and b) if selections per family in both interfaces follow the same underlying distribution. H_0 is set that the selections a) follow the uniform distribution or b) follow the same distribution. The alternative hypotheses, H_1, are set that the selections a) do not follow the uniform distribution or b) do not follow the same distribution. The degrees of freedom are set to the number of groups minus one; in this case, 5, as there are six families. Cohen's W is used to report effect sizes.

Neither the selections in the *cardboard* interface ($N = 708, \chi^2 = 202.69, p < 0.001, W = 0.434$) nor in the *oculus* interface ($N = 967, \chi^2 = 182.46, p < 0.001, W = 0.535$) follow a uniform distribution, as H_0 is rejected with a confidence level $\alpha = 0.05$ in both cases, with a medium and large effect size, respectively. If we compare both devices, ($N = 1675, \chi^2 = 4497.13, p < 0.001, W = 0.745$), H_0 is also rejected at $\alpha = 0.05$, so we can accept H_1: the selections in both devices are not drawn from the same underlying distribution with a large effect size.

We can begin to answer RQ2: the selections of virtual objects are not uniform across families and the distribution of selections is not the same between devices.

4.2.2. Distribution of selections in each computer family

Finally, we wanted to know if the distribution of selections in each device, in each family of computers, is uniform and if it is the same between devices. The actual number of selections were used this time, as the data was not aggregated.

We followed a similar approach: per each family of computers, Chi-square goodness of fit tests were performed to test a) if the selections per computer follow a uniform distribution and b) if selections per computer in both interfaces follow the same underlying distribution. The null and alternate hypotheses are defined in a similar way than before, so are the degrees of freedom, which will change per family (as the number of computers per family varies).

Table 4.2. Results of the Chi-Squared tests performed on the selection of computers in each family.

Family	***A***	***B***	***N***	***DoF***	χ^2	***p***	***W***
Classic Macs	*oculus*	*uniform*	73	4	6.93	0.14	0.308
Classic Macs	*cardboard*	*uniform*	313	4	30.88	< 0.001	0.314
Classic Macs	*oculus*	*cardboard*	386	4	1422.73	< 0.001	0.459
Macintosh II	*oculus*	*uniform*	37	3	4.62	0.20	0.353
Macintosh II	*cardboard*	*uniform*	116	3	3.59	0.31	0.176
Macintosh II	*oculus*	*cardboard*	153	3	393.32	< 0.001	0.508
Performa	*oculus*	*uniform*	70	4	18.86	< 0.001	0.519
Performa	*cardboard*	*uniform*	128	4	7.31	0.12	0.239
Performa	*oculus*	*cardboard*	198	4	509.72	< 0.001	0.433
PowerBook	*oculus*	*uniform*	132	7	41.45	< 0.001	0.560
PowerBook	*cardboard*	*uniform*	153	7	20.12	0.01	0.363
PowerBook	*oculus*	*cardboard*	285	7	978.71	< 0.001	0.615
iBook...	*oculus*	*uniform*	202	7	139.07	< 0.001	0.830
iBook...	*cardboard*	*uniform*	151	7	17.74	0.01	0.343
iBook...	*oculus*	*cardboard*	353	7	861.03	< 0.001	0.512
MacBook...	*oculus*	*uniform*	194	8	113.86	< 0.001	0.766
MacBook...	*cardboard*	*uniform*	106	8	15.08	0.06	0.377
MacBook...	*oculus*	*cardboard*	300	8	628.60	< 0.001	0.560

The results of these tests are reported in Table 4.2: the first column indicates which computer family is analyzed; columns *A* and *B* represent the groups which are compared: either one interface with the uniform distribution or both interfaces with each other; *N* denotes the number of samples; *DoF* the degrees of freedom used in the test; χ^2 the result of the test; *p* the p-value; and *W* Cohen's *W*.

The behavior per family varies: in some of them, the selections are apparently uniform in one interface or the other (rows in which *B*=*uniform*), so H_0 cannot be rejected, but for every family, the distribution of selections is different between both interfaces (rows in which *A*=*oculus* and *B*=*cardboard*) and H_0 can be rejected at $\alpha = 0.05$ with at least a medium or strong effect size. Notably, the selections in the first family (Classic Macs) may be uniformly distributed in the *oculus* interface, but are not in *cardboard*, but for the next family (Macintosh II), both may be uniformly distributed at a low significance level. From the third family (Performa), the distribution of selections is not uniformly distributed in *oculus* ($p < 0.001$). In the fourth and fifth families (PowerBook and iBook...), it is not uniformly distributed in *cardboard* ($p = 0.01$), while *p* rises to 0.06 in the last family (MacBook...).

We can continue to answer RQ2: the distribution of object selections inside each family is always different between devices, but there are some cases in which it can be uniformly distributed (first two families with *oculus* and second and third families with *cardboard*).

4.2.3. Correlation of the selections and the position of the virtual object

We formulated two sets of hypotheses: one based on the distance from the center of the virtual environment, and another based on the *z* component of the position of the object.

In the first case, the null hypotheses H_0 is set that there is no correlation between the number of selections received by a computer and its distance from the centre of the scene. Using Spearman's rank correlation, we found that we can reject H_0 at a confidence level $\alpha = 0.05$ in selections made in the *cardboard* interface, as they show a medium inverse correlation ($\rho = -0.77$, $p < 0.001$) but we cannot reject it, as they are not significantly correlated, in the *oculus* interface ($\rho = 0.23$, $p = 0.15$).

In the second case, we set the null hypotheses H_0 as that there is no correlation between that number of selections and the value of the z axis in the absolute position of that computer in the virtual environment. In this case, we cannot reject H_0 in selections made in the *cardboard* interface, as the correlation is not significant ($\rho = -0.22$, $p = 0.18$) but we can reject it at a confidence level $\alpha = 0.05$ in the *oculus* interface, as they show a medium inverse correlation in that case ($\rho = -0.64$, $p < 0.001$).

Per this, we can assert with a confidence level $\alpha = 0.05$ that the selections of the virtual objects and their position is correlated in both interfaces: in the *cardboard* interface it is inversely correlated with the value of the z axis of their absolute position, while in the *oculus* interface, it is inversely correlated with their distance from the center.

With this, we will conclude our answer of RQ2: the interaction with virtual objects in *oculus* is inversely correlated with the value of the z axis of the objects, while in *cardboard*, it is inversely correlated with their distance from the center.

4.3. Spatial navigation in the virtual environment

The collected behavior can also be used to plot the position in the virtual environment where this experience happened. We already discussed in section 3.5.4 Figure 3.22, which relates the spatial position of the selection of virtual elements and the position of the users in the virtual environment.

We can observe in Figure 4.4 the dispersion of the users' positions in the virtual museum. While the position of *cardboard* users (Figure 4.4 (b)) have similar dispersion in both axes ($\sigma_x = 6.30$, $\sigma_z = 5.57$, $\text{cov}(x, z) = -0.44$), *oculus* users (Figure 4.4 (a)) are more disperse, especially in the z axis ($\sigma_x = 8.95$, $\sigma_z = 11.38$, $\text{cov}(x, z) = 13.86$), as we can see more in the negative direction.

To answer RQ3 (Do users navigate the virtual space in the same way?) and assess if these two samples of spatial positions are drawn from the same underlying distribution, we performed a two-dimensional Kolmogorov-Smirnov test [108], based on Taillon's python library[22]. We set the null hypothesis H_0 as that both samples are drawn from the same distribution. For $N_{oculus} = 176092$, $N_{cardboard} = 162553$, we obtained a considerable $D = 0.5154$, with $p < 0.001$, so we can reject H_0 and affirm that the spatial positions are significantly different per device used at a confidence level $\alpha = 0.05$.

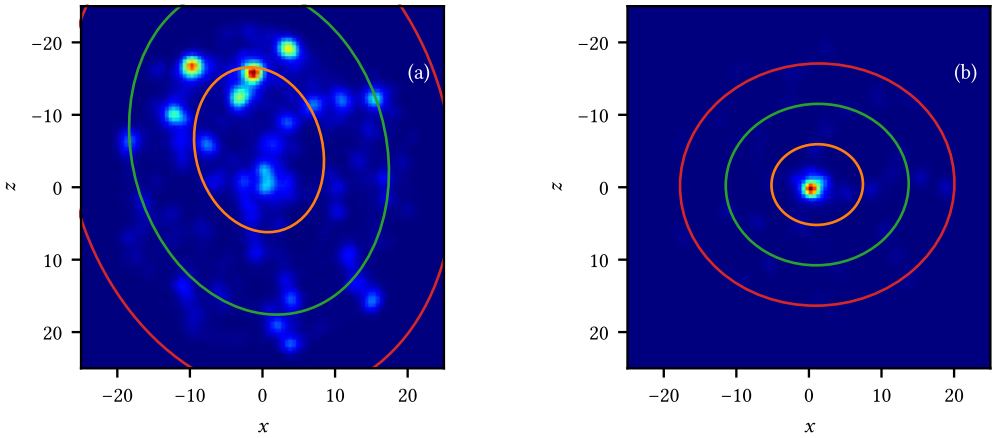

Figure 4.4. Heatmap of the users' positions in the virtual environment when using the *oculus* interface (a) and the *cardboard* interface (b). Concentric ellipses represent the covariances at 1, 2, 3 σ. Built with 100×100 bins and smoothed with a gaussian filter $\sigma = 1.5$.

[22] https://github.com/Gabinou/2DKS

4.4. Head orientation in the virtual environment

We can observe that, in *oculus*, many users stayed in the top part of the graphs in Figure 3.22 and Figure 4.4. To try to understand it better we plot the users' head orientation in Figure 4.5. We can see that, in Figure 4.5 (a), the *oculus* users' longitude is, most of the times, in the $[-90, 90]$ quarter, which corresponds to the direction of the negative z axis, with the centroid in $\bar{x} = (-10.2, -13.41)$; while in Figure 4.5 (b), the *cardboard* users' longitude shows a rise around 70°, with the centroid in $\bar{x} = (-18.91, 56.41)$. The dispersion in longitude is similar, but *cardboard* users showed more variation in latitude: in *oculus*, $\sigma_{lat} = 19.97$, $\sigma_{lon} = 87.92$, $\text{cov}(lat, lon) = -65.00$, while in *cardboard*, $\sigma_{lat} = 31.41$, $\sigma_{lon} = 94.67$, $\text{cov}(lat, lon) = 148.30$.

To answer RQ4 (Do users look around, turning their heads, similarly?) we set the null hypothesis H_0 as that the variance in longitude between groups is equal. Levene's test for homogeneity of variances on the longitude reports unequal variances between groups ($F = 3129.96$, $p < 0.001$), so we can reject the H_0 and assume that the users move their heads differently.

We calculated the dispersion of the users' orientations by adapting the Mean Average Error (3.4), s', and Standard Deviation (3.5), s'', formulas for spherical coordinates, with d as the spherical distance between each observation and the previously reported centroids (3.1). These values are similar: in *oculus*, $s' = 69.3$ and $s'' = 83.90$, while in *cardboard*, $s' = 73.82$ and $s'' = 84.19$.

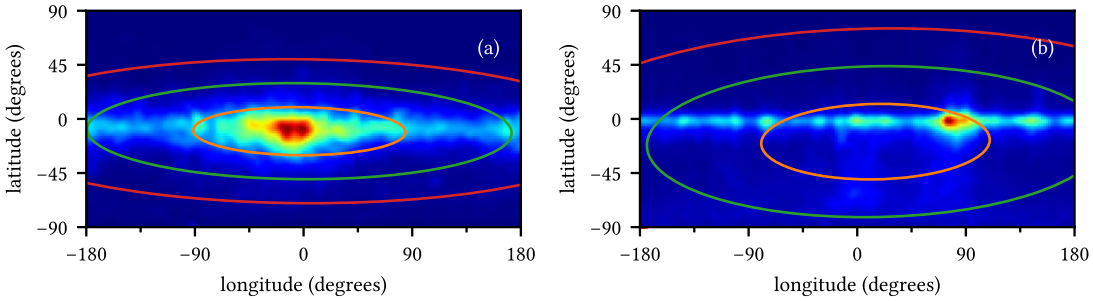

Figure 4.5. Heatmap of the users' head orientation in the virtual environment when using the *oculus* interface (a) and the *cardboard* interface (b). Concentric ellipses represent the covariances at 1, 2, 3 σ. Built with 180×90 bins and smoothed with a gaussian filter $\sigma = 1.5$.

We observed the distribution of the distance between the centroid and the orientations per device type (its average is s'). Levene's test for homogeneity of variances reports unequal variances between groups ($F = 5656.38$, $p < 0.001$), so a Welch's t-test was performed on the great-circle distance between the orientations and the centroid per device type. The null hypothesis H_0 is set that the mean of that distance is equal between groups. The result, $t(338645) = -29.76$, $p < 0.001$, rejects the null hypothesis, so we can accept the alternate hypothesis H_1 that the average distance between the orientation of the observations and their centroid is different per device with a confidence level $\alpha = 0.05$. A Cohen's $d = 0.103$ was reported by the test, a very small effect size.

With this last knowledge, we can add to the answer to RQ4: the orientation of the users of the *cardboard* interface was significantly more disperse than those of the *oculus* interface, although the size of this effect seems to be limited.

4.5. Behavior across web browsers

Regarding RQ5 (When the *cardboard* interface is used, is the behavior uniform across different web browsers?), we observed the difference in behavior between the *oculus* and *cardboard interfaces* described in the past section and wondered if there would be a difference between the web browsers used to play the latter interface, as we found surprising that most of the behavior in *cardbard* happened around 90° in longitude.

As we can see in the first row in Figure 4.6, the orientation of the users' heads differs when they are using different browsers: in Figure 4.6 (a), users of Chrome in Android ($N = 95$) tend to look at around 70° in longitude, but in Figure 4.6 (b), users of Chrome in iOS ($N = 19$) tend to look at around 0° in

longitude, while users of Safari in iOS ($N = 47$), as reported in Figure 4.6 (c), tend to look at both around 0° and around 70° in longitude.

In further analysis, shown in the second row in Figure 4.6, we obtained the orientation in which the first selection was done in each session. We can notice three things, which we will test next: first, the orientation per browser in the first row seems different; second, the orientation of the first selections per browser in the second row seems different (at least between (d) and (e)); third, there appears to be a relationship between the orientation in which the first selection was performed (second row) and the most viewed orientations (first row) per browser.

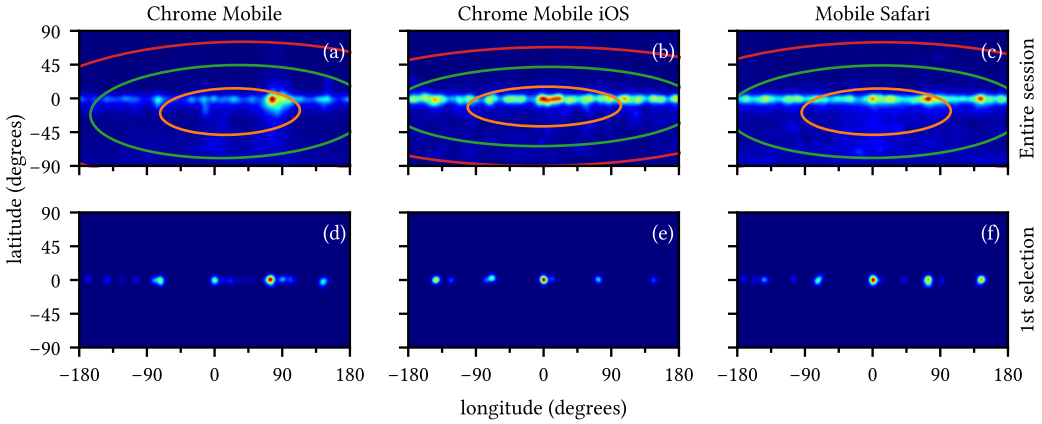

Figure 4.6. Heatmap of the users' head orientation in the virtual environment when they use three specific browsers in the *cardboard* interface: Chrome in Android (first column), Chrome in iOS (second column) and Safari in iOS (third column). The first row contains the data from all the session, with concentric ellipses that represent the covariances at 1, 2, 3 σ, the second row contains the data of only the first computer selection in each session. Built with 180×90 bins and smoothed with a gaussian filter $\sigma = 1.5$.

To test the difference between browsers, we design H_0 as that samples from two given browsers come from the same distribution. We employ a Kolmogorov-Smirnov test on the normalized longitude (as objects are always in latitude 0°) to assess that hypothesis. Here we could reject H_0 in all the cases at $\alpha = 0.05$: between Chrome Mobile and Chrome Mobile iOS ($t = 0.1249$, $p < 0.001$), between Chrome Mobile and Mobile Safari ($t = 0.1145$, $p < 0.001$) and between Chrome Mobile iOS and Mobile Safari ($t = 0.0264$, $p < 0.001$).

The same test was performed on the orientations of the first selections, with mixed results: between Chrome Mobile and Chrome Mobile iOS, we obtained $t = 0.4316$ with $p < 0.01$; between Chrome Mobile and Mobile Safari, $t = 0.1424$ with $p = 0.49$; between Chrome Mobile iOS and Mobile Safari, $t = 0.3427$ with $p = 0.06$. With a 5% confidence level, we can only reject H_0 in the first case: orientation of the first selection in Chrome Mobile and Chrome Mobile iOS are not drawn from the same distribution.

Finally, we calculated the dispersion in longitude and the two spherical measures of dispersion, which we described in section 4.4, between the orientation of each user and the orientation of their first selection. The results per session were averaged to obtain a single value. The average standard deviation of the longitude is $\sigma_{lon} = 80.04$; while the average of the spherical measures based on the MAE and standard deviation are $s' = 67.81$ and $s'' = 79.17$, respectively. All these values of dispersion are smaller than when the overall centroid is considered (as we reported in section 4.4).

The average per browser of these values of dispersion between the orientation in a session and the orientation of the first selection are as follows: in Chrome Mobile, $\sigma_{lon} = 72.45$, $s' = 62.00$, $s'' = 72.97$; in Chrome Mobile iOS, $\sigma_{lon} = 86.77$, $s' = 72.41$, $s'' = 85.63$; in Mobile Safari, $\sigma_{lon} = 91.86$, $s' = 77.58$, $s'' = 88.73$.

The dispersion between the orientation and the centroid per browser (similarly as what we reported in section 4.4) are the following: in Chrome Mobile, the centroid is $\bar{x} = (-17.58, 63.39)$, $\sigma_{lon} = 89.55$,

$s' = 71.22$, $s'' = 82.39$; in Chrome Mobile iOS, $\bar{x} = (-10.85, 36.71)$, $\sigma_{lon} = 101.59$, $s' = 84.59$, $s'' = 96.29$; in Mobile Safari, $\bar{x} = (-17.72, 31.31)$, $\sigma_{lon} = 98.74$, $s' = 80.40$, $s'' = 90.52$.

The distribution of the distance between the first selection and the other orientations was calculated with the great-circle distance formula (3.1). We compared this distribution between browsers. Levene's test for homogeneity of variances reports unequal variances in all the cases: between Chrome Mobile and Chrome Mobile iOS ($F = 226.89$, $p < 0.001$); between Chrome Mobile and Mobile Safari ($F = 65.62$, $p < 0.001$); and between Chrome Mobile iOS and Mobile Safari ($F = 349.85$, $p < 0.001$). A set of Welch's t-tests were performed on the distance between the orientation of that first selection and the rest per browser, so H_0 is set that both distributions have equal means. Effect size is also reported with Cohen's d. The results are as follows: between Chrome Mobile and Chrome iOS ($t(104969) = -40.21$, $p < 0.001$, $d = -0.363$); between Chrome Mobile and Mobile Safari ($t(129957) = -61.03$, $p < 0.001$, $d = -0.363$); and between Chrome Mobile iOS and Mobile Safari ($t(56926) = 0.53$, $p = 0.60$, $d = 0.005$). We can reject H_0 and accept that the distribution of the distance between the orientations and the orientation in the first selection has different means between Chrome Mobile and Chrome Mobile iOS and between Chrome Mobile and Mobile Safari at a confidence level $\alpha = 0.05$ with small effect sizes, but we cannot prove that between Chrome Mobile iOS and Mobile Safari, with no effect size.

All these observations help us answer RQ5 (When the *cardboard* interface is used, is the behavior uniform across different web browsers?): there is a significant difference in the values of longitude, while the distribution of the distance between the first selection in a session and their orientations is statistically significant between the Android and either iOS browser (Android users tended to be less disperse), while it is not significant between both iOS browsers.

4.6. Discussion and conclusion

We studied the user behavior with two different devices in a virtual environment, guided by the research questions we formulated in the beginning of the chapter.

As observed in Figure 4.1, there were many *cardboard* sessions which were either very short or in which the user did not select any computer. Since this was the first time most of the users tried virtual reality and they did it using the simplest interface, we can assume that some of them had difficulties understanding the interface. Another factor that could have had an impact is the technical capacity of the smartphones used by the participants in the experiment: although the VR player does not require a high-end smartphone to work correctly, it does need to have some sensors that may be absent in lower-end devices, accounting for some of these sessions with poor performance.

In contrast, we detected several mobile sessions that lasted more than 5 minutes or selected more than 10 different computers. Our point of view on this behavior is that as users could play with their own devices, some of them chose to extend their experience as much as possible. We planned that the users would play for approximately 5 minutes and interact with half a dozen computers. This was the behavior of most of the mobile sessions, most surely because of the time slots, but some of these sessions, as we are discussing, were longer and more active than expected.

We found a significant difference in the average time between interactions with the virtual objects between devices: 34 seconds in *cardboard* and 46 seconds in *oculus*, on average. The most logical explanation lies in the difference of the interaction: mobile users only had to look at a computer and hold their gaze at the desired computer for two seconds; while *oculus* users needed to walk near to the desired computer using the joystick and then touch it with the virtual hands using the specific gesture.

Virtual objects are not interacted in the same way between devices: selection patterns are never based on the same distribution between devices, although the selection in some families may be uniform with some devices. Furthermore, we found that there there is a significant inverse correlation in *cardboard* with the distance from the center and in *oculus* with the z axis. In any case, computer selections are related to the users' spatial position in the virtual environment (Figure 3.22).

We observed that *oculus* users were more spatially disperse than *cardboard* users. The users of the first device tended to explore the northern part of the scene, while users of the second usually

remained in the middle of the environment. We found a statistically significant difference between the spatial positions of both groups.

This difference in spatial distribution can be attributed to the difference in interaction between interfaces: *cardboard* users started their experience in the middle of the scene and could only move around by selecting objects in the virtual world, whereas *oculus* users could move freely and started their experience where the previous user was. This explains the behaviour in *cardboard*, but to answer why most of the times *oculus* users remained in that area, we must analyse the orientation of the users' heads.

The first difference between users' orientation we can observe is that most of the times, *oculus* users looked forward (in the negative direction of the z axis), while in *cardboard*, in which users tended to look more towards the right of the scene. The second difference is the variance in longitude of these orientations: $\sigma_{lon} = 87.92$ in *oculus* and $\sigma_{lon} = 94.67$ in *cardboard*. Despite the small difference in standard deviation, variance in longitude was reported to be significantly different. We obtained spherical values of dispersion and we compared the distribution of one of them between devices: the average spherical distance with the centroid of the observations was found to be significantly different, with the orientation of *cardboard* users more disperse than that of *oculus* users.

When we took under consideration the mapping of the virtual world with the real world, we noticed that the table which contained the monitors and sensors for the *oculus* interface (see section 3.5.3) was in the negative direction of the z axis in the virtual environment so users, even when they had 360 degrees to explore, tended to choose that direction most of the times. This also influenced the spatial distribution of *oculus* users, as they were already at the limit of the environment, they usually did not turn around to advance in the opposite direction.

Conversely, users of the *cardboard* interface were not aware of any apparatus in the physical world that could condition their orientation. Despite this, Figure 4.5 (b) shows a bias around longitude 90º, which could be related to the default position of the web browsers used to play the *cardboard* interface in smartphones.

We noticed that the browser used to play the virtual experience can play a role in the users' behavior. We observed how different web browsers show different highly played orientations. Significant differences on the normalized longitude were found between browsers.

We observed how these most frequent orientations are very close to the orientation of the first selection in each web browser. Precisely, different browsers showed different distributions of first selections: the difference between Chrome Mobile (Android) and Chrome Mobile iOS was significant, while between the other pairs it was not significant.

Those first selections, when used to calculate the dispersion in longitude and the spherical dispersion, resulted in lower values of dispersion than when the mean longitude or the centroid of observations was considered. Breaking down by browser, we observed the same behavior: the average dispersion is lower when calculated with the orientation of the first selection than when it is calculated with the mean longitude or the centroid of the observations, effectively linking the orientation of the users' heads to the orientation in which that first selection was performed.

We observe that users of Google Chrome have very strong biases in the orientation of their first selection: in Android, the most common first selection was in longitude $\lambda = 72º$, while in iOS it was in longitude $\lambda = 0º$. Users of Safari iOS did not show such strong biases.

The initial orientation of the users when they start their sessions may have played a role in that first selection: the sensor used to track the orientation in the virtual scene may be based on the Earth's magnetic field[23], acting like a compass and aligning the virtual scene and the virtual world. The exact

[23] Stated by one of the creators of AFRAME in https://stackoverflow.com/a/47667912/3579466: "*look-controls* takes the orientation from the sensors of the phone that is absolute and not relative to where you position the camera initially in the scene".

implementation was apparently different by web browsers, and apparently Google Chrome (Android) initialized orientation to longitude $\lambda = 90^{o}$ and Google Chrome iOS defaulted to $\lambda = 0^{o}$.

We used the orientation of those first selections to calculate the distribution of the spherical distance of the orientations. We could prove how the mean distance between the Android browser and either iOS browser is significantly different, but it is not significant between both iOS browsers.

To conclude, we have found several differences between both devices. These differences, although significant, are not vastly different between each other. This is because, although the interface differs, they are both immersive devices and are subject to the same constraints of the human body, especially in terms of orientation.

Differences in the interface, like the trigger to select a virtual object or the initial values in the experience, such as the initial position or the initial orientation in the virtual environment are likely to cause differences in user behavior: the selection of virtual objects was easier with the *cardboard* interface than with the *oculus* interface; the fact that users in *cardboard* started in the middle of the scene while users of *oculus* did not, affected both their positions in the virtual environment and their selections of virtual objects; finally, users of *oculus* tended to look towards the physical table which was found in the experimental setup, while users of *cardboard* showed different distributions of first selections per web browser, which conditioned the general orientation in those browsers.

These differences are important and must be considered if a uniform experience is to be provided across different devices. Otherwise, when it is not possible to make the interaction similar between devices (due to, e.g., limitations on the device), part of the virtual environment is going to be underexplored with a device or a browser.

We have observed the following that can introduce differences between these two interfaces. First, relying in specific gestures to select an object can make the process more difficult. Second, the positioning of the objects in the scene, in relation to the starting position of the player, makes an important role on which are interacted. A layered design worsened the situation, as to arrive to the farthest objects, one had to pass through all the others first. Third, and related to the previous, the initial position of the player in the system is important. Fourth, the use of a monitor could have been a good reclaim to make users interested in our stand in the science fair, but it made participants look at that direction most of the times. Finally, and related to the third point, the initial orientation is also important. An effort to make users look consistently into a desired direction may be desirable.

Differences in behavior between these immersive devices and non-immersive devices (pc and smartphone) must have been more important but were not collected in this study. The following chapter studies the users' behavior with these four devices in 360-degree videos.

Chapter 5. Analysis of user behavior in a multi-device 360-degree hypervideo

In this chapter, we perform a study of user behavior when playing the 360-degree videos in the multi-device 360-degree hypervideo reported in section 3.6. A user experiment was detailed in section 3.6.5, which resulted in the collection of a dataset of user behavior when these videos were played with four different devices: a desktop computer, a smartphone, a mobile-based VR headset (a Google Cardboard) and a desktop VR headset with controllers (an Oculus Rift). The first two are interfaces based on non-immersive devices, while the other two are based on immersive devices.

Two types of videos were identified in section 3.6.1: point videos, in which the camera is static, and route videos, in which the camera follows a path. All the videos feature similar imagery: they were recorded in the streets of the city of Palma, and they together make an interactive tour.

Although prior studies consider a single device in their experiments, as previously discussed in Chapter 2, we base our grouping of video types on the use in [48] of Temporal perceptual Information [103], which did not analyze the difference in behavior between different video types. The following parts of this chapter are based on prior work, extended to compare the differences between video types and devices: our exploratory analysis is based on that work, with a clear distinction by video type and device used; the work of Corbillon *et al.* [47] serves as a basis for our movement analysis; finally, we base our analysis of the trajectory of the users' orientations in [109].

To assess the differences in behavior between the four devices and the two types of videos, we formulate the following research questions.

RQ1. Is spatiotemporal exploration the same across devices and video types?
RQ2. Is the dispersion of observed viewport centers similar across devices and video types?
RQ3. Is movement performed with the same magnitude across devices and video types?
RQ4. Are trajectories followed in the same way across devices and video types?

In the following sections, we report the structure of the dataset, perform an exploratory analysis of the collected data, and will answer these questions with the analysis of the data in multiple ways, ultimately comparing devices in the same video type and video types with the same device. We will end this chapter with some conclusions obtained from this analysis.

5.1. The dataset

A total of 140 user sessions were registered during the experiment. Each of these user sessions US consists of a series of linked video sessions VS played by a user u with a device d.

$$us \in US = \{VS, u, d\}$$

Each of these video sessions VS contains the behavior of playing a video v_s using a device d_s in the form of a distribution of viewport centers during its playback (F_s).

$$s \in VS = \{v_s, d_s, F_s\}$$

These videos of V are identified by an id, categorized in a $type$ (point or route videos), have a $length$ in seconds and define the src of the multimedia file.

$$v \in V = \{id, type, length, src\}, type \in \{p, r\}$$

The devices used to play them are $d \in \{pc, mb, mv, oc\}$ (PC, Mobile, Mobile visor and Oculus, respectively).

The distribution of viewport centers F during the playback of the video (from *start* to *end*) is represented as follows:

$$F = \{f_t\}_{t=start}^{end}, 0 \leq start \leq end \leq length$$
$$f_t = \{\phi_t, \lambda_t\}, \phi_t \in [-90,90], \lambda_t \in \mathbb{R}$$

A video session can *start* at a point in time after the beginning of the video ($t = 0$) and can *end* before the end of the video ($t = length$).

The angles ϕ_t and λ_t represent, respectively, the latitude and longitude of the center of the viewport of the session at consecutive frames each 250 ms, where (0,0) represents the center of the video, which is 180 degrees tall and 360 degrees wide. Longitude is not normalized to $[-180,180)$, as the user may turn around (although it will often be normalized in this study).

To obtain F, the data extracted from the LRS had to be preprocessed to make the sequence of frames aligned to multiples of 250 milliseconds because, while the source data is recorded roughly every 250 milliseconds, the recorded temporal sequence may not match the video timeline. This operation was achieved by linear interpolation of these points on a great circle following (3.2): given two frames, $f_{t_0} = \{\phi_0, \lambda_0\}$, $f_{t_1} = \{\phi_1, \lambda_1\}$, and a time $t, t_0 < t < t_1$, the great circle distance between them, d, and the interpolated frame $f_t = \{\phi_t, \lambda_t\}$ are calculated with (3.1) and (3.2), respectively.

As point videos are played in loop (in contrast to route videos which are skipped when they end), frame data belonging to a looped reproduction of a point video is discarded, and only the first loop-worth of data is kept.

5.2. Exploratory analysis

A total of 20 different videos were played in the experiment. 8 of these were point videos, in which the camera does not move, and the other 12 were route videos, in which the camera follows a route through a street. Two of these 8 point videos were excluded from the analysis, as they were reproduced very few times (see section 3.6.6).

Table 5.1. Count of video sessions per video type and device.

	pc	***mb***	***mv***	***oc***
Point videos	120	102	108	165
Route videos	105	94	95	139

The number of reproductions per video type and device can be found in Table 5.1. It can be appreciated that not all the user sessions played all the videos in all the devices, so there is more data available for some videos than the others.

Table 5.2. Count of frames per video type and device.

	pc	***mb***	***mv***	***oc***
Point videos	2433	3181	3566	5211
Route videos	9673	7567	8381	15374

The total amount of data samples can be observed in Table 5.2: here, again, not all video sessions played the same video for the same amount of time, as some video sessions ended early through the use of a hyperlink or started late due to the effect of the incoming hyperlink.

The orientation of the frames per video type and device is drawn in the following figures, following the study reported in [48]. We can observe the distribution of latitude in Figure 5.1. There is not an evident difference between video types, but values of latitude are lower in immersive than in non-immersive interfaces: the mean latitude in point videos for the oculus interface is $\text{avg}(\phi_{p,oc}) = -11.71$, while in route videos it is $\text{avg}(\phi_{r,oc}) = -12.04$; for the mobile visor interface, these values are $\text{avg}(\phi_{p,mv}) = -15.49$ and $\text{avg}(\phi_{r,mv}) = -11.53$; for the pc interface, they are $\text{avg}(\phi_{p,pc}) = -6.65$ and $\text{avg}(\phi_{r,pc}) = -6.82$; and for the mobile interface, $\text{avg}(\phi_{p,mb}) = -6.91$ and $\text{avg}(\phi_{r,mb}) = -5.01$.

The non-normalized values of longitude are plotted in Figure 5.2, limited to the range $[-720,720)$. Non-normalized values are not going to be used throughout this section extensively, but they serve to

highlight these points: first, users tend to keep watching at the longitude they start (around 0°); second, values of longitude that are normalized to around 0° are the most viewed, especially in route videos (Figure 5.2 (b)), but also in point videos (Figure 5.2 (a)), to a lesser extent; finally, users of immersive interfaces tend to turn around less than users of non-immersive interfaces: values of standard deviation of non-normalized longitude are as follows for each video type and device: std($\lambda_{p,oc}$) = 116.18, std($\lambda_{r,oc}$) = 88.96, std($\lambda_{p,mv}$) = 99.82, std($\lambda_{r,mv}$) = 85.52, std($\lambda_{p,pc}$) = 559.22, std($\lambda_{r,pc}$) = 695.80, std($\lambda_{p,mb}$) = 274.82, std($\lambda_{r,mb}$) = 330.70.

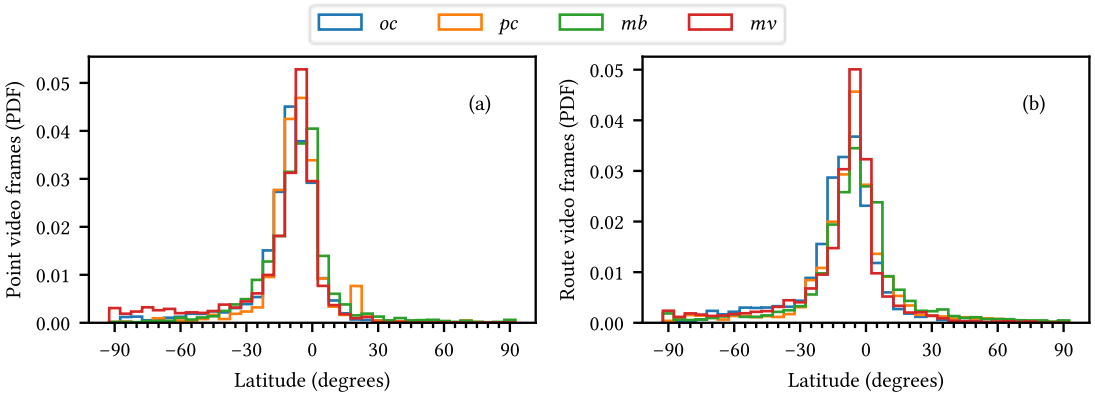

Figure 5.1. PDF of latitude per video type and device used to play them. Bins of 5 degrees of latitude.

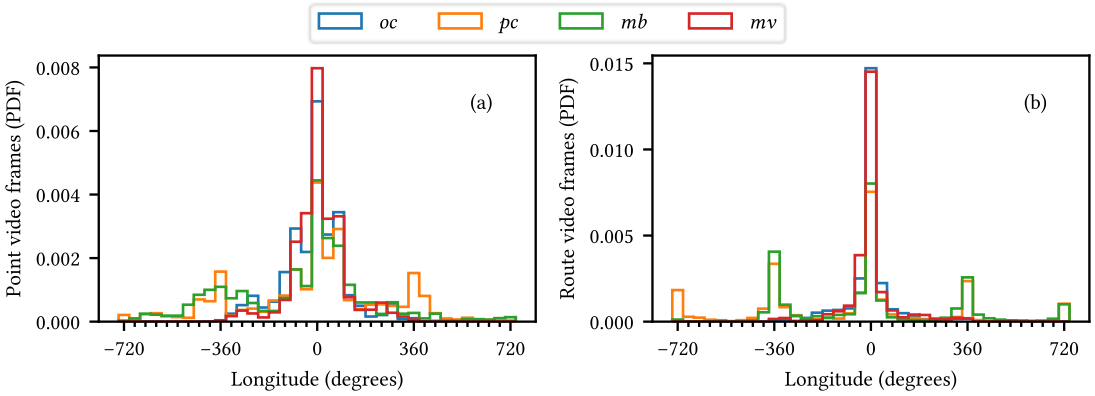

Figure 5.2. PDF of longitude per video type and device used to play them. Bins of 40 degrees of longitude.

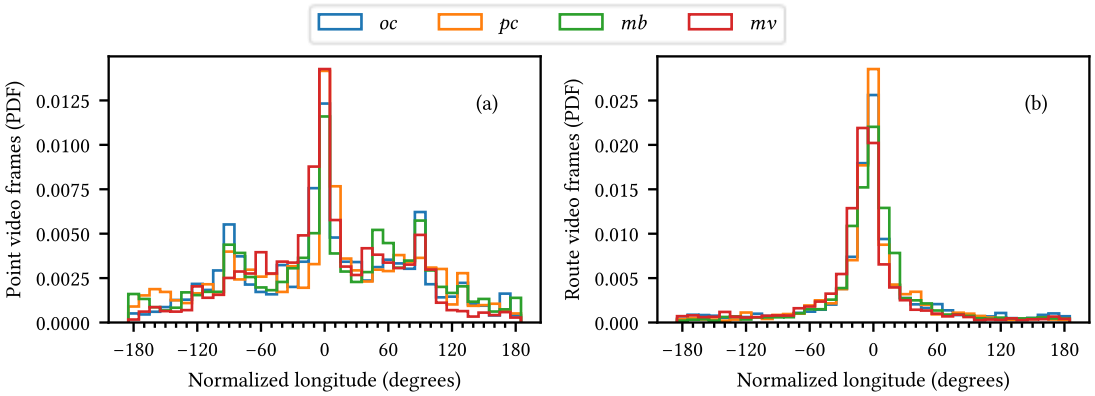

Figure 5.3. PDF of normalized longitude per video type and device used to play them. Bins of 10 degrees of longitude.

Those values of longitude are normalized and plotted in Figure 5.3. Here we cannot appreciate much difference between devices, but the difference between video types is very evident: although in both video types the most frequent values of longitude are around 0°, they are much more disperse in point

videos (Figure 5.3 (a)) than in route videos (Figure 5.3 (b)). In this case, the standard deviation of the normalized longitude per video type and device were found to be $\text{std}(\lambda_{p,oc}) = 78.19$, $\text{std}(\lambda_{r,oc}) = 57.60$, $\text{std}(\lambda_{p,mv}) = 67.64$, $\text{std}(\lambda_{r,mv}) = 52.14$, $\text{std}(\lambda_{p,pc}) = 82.50$, $\text{std}(\lambda_{r,pc}) = 48.24$, $\text{std}(\lambda_{p,mb}) = 83.05$, $\text{std}(\lambda_{r,mb}) = 48.15$. We can observe lower values of deviation in route videos than in point videos. Deviation with immersive devices is lower in point videos than with non-immersive interfaces, while the opposite seems to be true in route videos.

These results are consistent with [48]: areas of interest are shown around $\phi = 0°$ and $\lambda = 0°$, with a more uniform distribution in longitude than in latitude.

5.3. Spatiotemporal exploration

In this section, to answer RQ1 (Is spatiotemporal exploration the same across devices and video types?), we analyze the spatiotemporal exploration of the videos in a video session: how much of the time of the video was consumed, how much of the 360-degree space was consumed, and whether these values are correlated or not.

5.3.1. Temporal exploration

We define the temporal exploration of a session is a measure of how much of a video was explored timewise in a session. It is calculated as the difference between *end* and *start* normalized to the video *length* (5.1).

$$e_t = \frac{end - start}{length} \tag{5.1}$$

It should be noted that point videos are played in loop, so $end = length$ if the entire video was played, even if it stopped being played halfway during a further loop.

The distribution of the temporal exploration by video type and interface device is shown in Figure 5.4 in the form of a Cumulative Distribution Function. There is an initial peak in $e_t \approx 0$, specially in the *mb* interface, which probably are erroneous sessions in which the user could not launch the system in the device and tried again later; 82 sessions show $e_t < 0.01$ and will be excluded from further analysis. It can be appreciated that $e_t \leq 0.5$ in 80% of point video sessions that are played with the *pc* interface and in 60% using the other interfaces, while this value drops to 30-50% in route video sessions.

Apparently, point videos played with the *pc* interface are less explored temporally than in the other interfaces and route videos are more explored temporally than point videos.

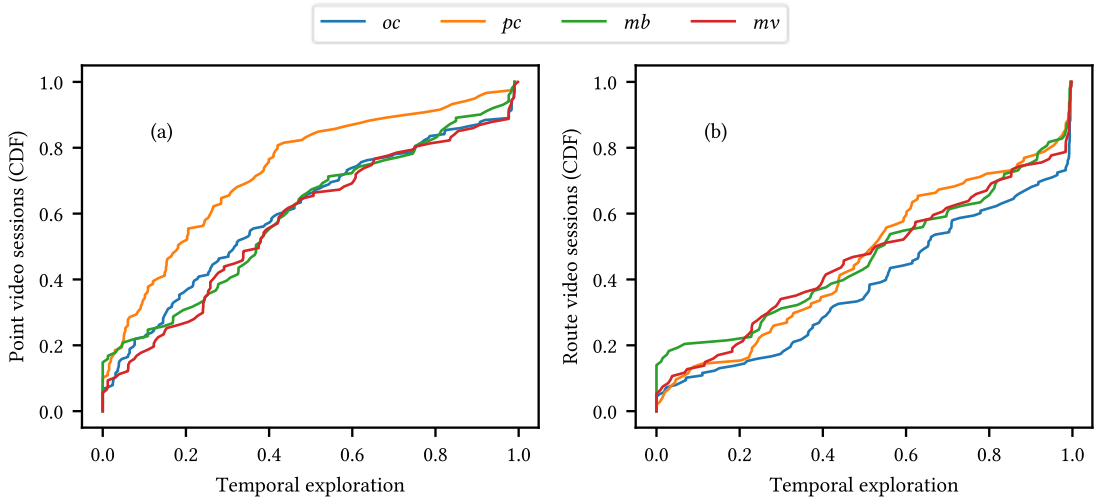

Figure 5.4. Cumulative Distribution Function of temporal exploration of the 946 sessions by video type and interface used.

To assess the difference between video types and devices, we will test if the e_t is different between groups with Mann Whitney's U test, as the Shapiro-Wilk test for normality was rejected ($p < 0.001$).

The rank-biserial correlation (r) is reported as a measure of effect size. Under the null hypotheses H_0, the distribution of temporal exploration between two categories of video sessions are drawn from the same underlying distribution. The alternative hypotheses H_1 sets that they are drawn from different distributions.

The null hypotheses could only be rejected at a significance level $\alpha = 0.05$ in point videos between *pc* and the other devices: between *pc* and *oc* ($N_{pc} = 107, N_{oc} = 153, U = 10102.0, p = 0.001, d = -0.234$), between *pc* and *mb* ($N_{pc} = 107, N_{mb} = 86, U = 2868.5, p < 0.001, r = 0.377$), and between *pc* and *mv* ($N_{pc} = 107, N_{mv} = 100, U = 3684.0, p < 0.001, r = 0.311$). In route videos, the difference is not so clear, and the null hypotheses could only be rejected at $\alpha = 0.05$ between *pc* and *oc* ($N_{pc} = 101, N_{oc} = 131, U = 7903.0, p = 0.011, r = -0.195$). These results are consistent with the visual analysis performed of Figure 5.4: the difference between *pc* and the rest of the interfaces in Figure 5.4 (a) is significant, while the difference between the others is not; and the difference between devices in Figure 5.4 (b) is only significant between the two extreme positions: *oc* and *pc*. We should consider that the repetition of the experiments increases the chance of error type I, so under Bonferroni's correction with 16 repetitions, this last difference may not be significant, at $\alpha = 0.003$.

The difference in e_t per device between video types is statistically significant in all the cases at $\alpha = 0.05$: in *oc* ($N_p = 165, N_r = 139, U = 6973.0, p < 0.001, r = 0.392$), in *pc* ($N_p = 120, N_r = 105, U = 3259.5, p < 0.001, r = 0.483$), in *mb* ($N_p = 102, N_r = 94, U = 3732.0, p = 0.007, r = 0.222$), and in *mv* ($N_p = 108, N_r = 95, U = 4098.0, p = 0.014, r = 0.201$). Again, under Bonferroni's correction with 16 repetitions, the differences in *mb* and *mv* may not be significant, at $\alpha = 0.003$.

With these first results, we can begin to answer RQ1 (Is spatiotemporal exploration the same across devices and video types?): temporal exploration is similar between devices (except in *pc*) and it is significantly different between video types.

5.3.2. Spatial exploration

The spatial exploration of a session is a measure of how much of a video was explored spatially in a session. We have chosen to simplify the measure to only account for the horizontal exploration (in longitude), as we have previously observed that most of the behavior takes place in the same latitude. It is calculated as the difference between the highest and lowest latitudes present in the session (5.2).

$$e_s = \min\left(360, \max_t \lambda_t - \min_t \lambda_t\right) \tag{5.2}$$

This calculation is possible because the recorded latitudes account for full rotations, so a user that starts at $\lambda_0 = 0$ and ends at $\lambda_1 = 365$, has explored 365 degrees and not only the 5 degrees that we would have found if full rotations were not accounted for. For this reason, this result is minimized to 360 (a full rotation) as, following the previous example, an exploration of 365 degrees means that the user explored the full space (360 degrees).

A histogram of the spatial exploration of the sessions with $e_t \geq 0.01$ (Figure 5.5) makes evident that most of the times, 360-degree videos are fully spatially explored, with spatial exploration in the last bin. Without accounting for full exploration sessions ($e_s \geq 350$), sessions with low spatial exploration, in the range $e_s \in [0,180)$, have been observed more than sessions with high spatial exploration, in the range $e_s \in [180,350)$ in both video types, but a difference between point and route videos can be observed: sessions with spatial exploration values $e_s \in [0,140)$ have been observed more when playing point videos than when playing route videos. The most likely explanation lies in the nature of the hypervideo design: point videos act as a gateway between videos and display hyperlinks that may be clicked without needing much exploration. In both kinds of videos, sessions played using non-immersive interfaces as the *pc* and *mb* interfaces seem to show greater spatial exploration than other devices and will be further discussed next.

A distinction per interface device is difficult to be observed in a histogram; Figure 5.6 contains the Cumulative Distribution Function of the previous data, and the differences between interface devices are more evident: spatial exploration in point video sessions with immersive interfaces (*oc*, *mv*) is lower than with non-immersive interfaces (*pc*, *mb*) but follows a similar trend, whereas there is a much larger difference between immersive and non-immersive interfaces in route video sessions.

Between video types, spatial exploration using immersive devices is similar in the two types of videos, but compared to route videos, it grows in route videos with non-immersive devices.

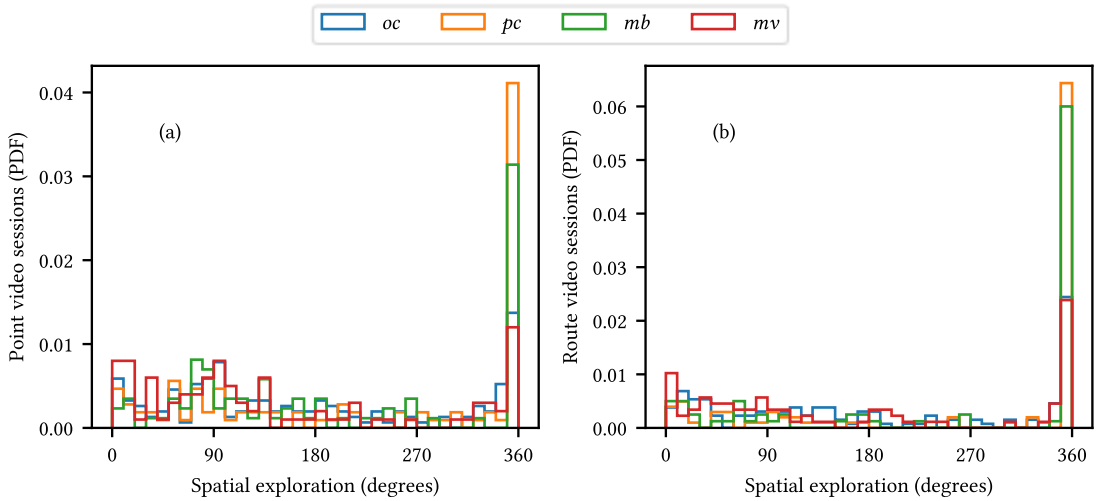

Figure 5.5. Histogram of spatial exploration of the 846 sessions with $e_t \geq 0.01$ by video type and interface used, relative to the number of sessions per interface, with bins of size 10 degrees.

We tested the difference in spatial exploration between groups with a series of Mann-Whitney's U tests, as the Shapiro-Wilk test for normality was rejected ($p < 0.001$). The rank-biserial correlation (r) is reported as a measure of effect size. Under the null hypotheses H_0, the distribution of e_s between two categories of video sessions are drawn from the same underlying distribution, while under the alternative hypotheses H_1, they are drawn from different distributions.

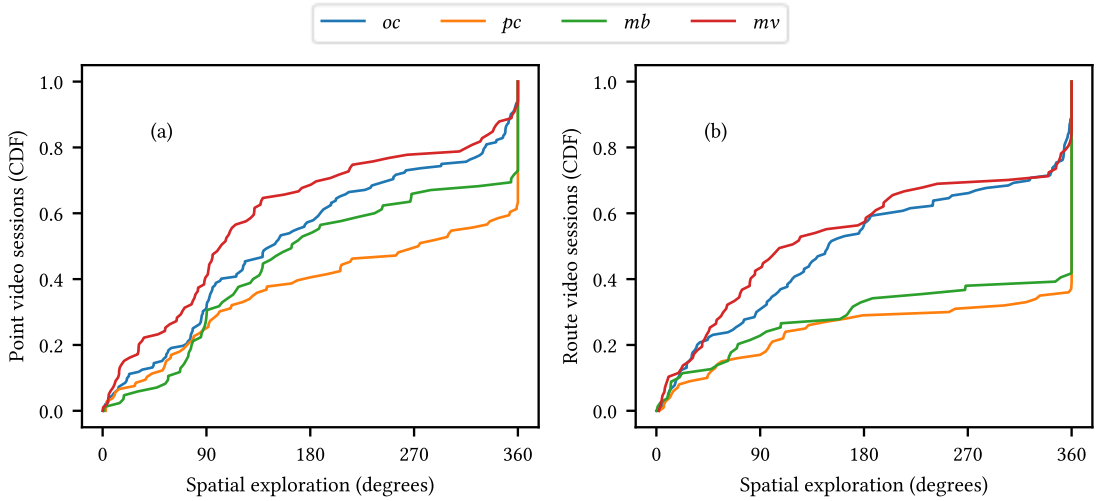

Figure 5.6. Cumulative Distribution Function of spatial exploration of the 846 sessions with $e_t \geq 0.01$ by video type and interface used.

In point videos, we could reject H_0 at $\alpha = 0.05$ between *pc* and *oc* ($N_{pc} = 107$, $N_{oc} = 153$, $U = 6050.0$, $p < 0.001$, $r = -0.261$), between *pc* and *mv* ($N_{pc} = 107$, $N_{mv} = 100$, $U = 7232.0$, $p < 0.001$, $r = 0.352$) and between *mb* and *mv* ($N_{mb} = 86$, $N_{mv} = 100$, $U = 5494.0$, $p = 0.001$, $r = 0.278$). In route videos, the differences between immersive and non-immersive devices are very evident and significant: between *pc* and *oc* ($N_{pc} = 101$, $N_{oc} = 131$, $U = 3708.0$, $p < 0.001$, , $r = -0.439$), between *oc* and *mb* ($N_{oc} = 131$, $N_{mb} = 80$, $U = 3291.0$, $p < 0.001$, $r = -0.372$), between *mb* and *mv* ($N_{mb} = 80$, $N_{mv} = 88$, $U = 4880.0$, $p < 0.001$, $r = 0.386$), and between *pc* and *mv* ($N_{pc} = 101$, $N_{oc} = 88$, $U = 2453.0$, $p < 0.001$, $r = -0.448$). All of these results would still be significant under Bonferroni's correction with 16 repetitions, at $\alpha = 0.003$.

Between video types, there is not a significant difference in e_t with immersive devices, but it is significant with non-immersive devices: with *pc* ($N_p = 107$, $N_r = 101$, $U = 4257.0$, $p = 0.005$, $r = 0.212$) and with *mb* ($N_p = 86$, $N_r = 80$, $U = 2645.0$, $p = 0.008$, $r = 0.231$). This differences in *mv* and *mb* may not be significant under Bonferroni's correction with 16 repetitions, at $\alpha = 0.003$.

We can observe that sessions with immersive devices explore the 360-degree space approximately equally in both kinds of video and that video sessions made with immersive interfaces explore less space than those made with non-immersive devices. We understand that users of immersive devices have to perform a physical effort to turn around, so this action is less likely to be performed in great extents than with a non-immersive interface, while the opposite may be true for short movements, this does offer an explanation concerning the full exploration of the space of a 360-degree video.

We can now continue our answer of RQ1 (Is spatiotemporal exploration the same across devices and video types?): spatial exploration is significantly different in point videos between some devices and it is not between some others, while in route videos, it is significantly different between immersive and non-immersive devices. Between video types, it is not significant with immersive devices, but it is with non-immersive devices.

5.3.3. Relationship between temporal and spatial exploration

Once spatial and temporal exploration have been analyzed separately, they should be analyzed together.

Figure 5.7 contains a scatter plot of both measures per video type and device. It is interesting to note the maximum exploration values in both dimensions in the top and right sides of each plot, as expected per previous analysis, but a cluster in the lower-left corner in both video types, especially in point video sessions, with $e_t \in [0,0.5)$ and $e_s \in [0,140)$ can also be observed. This cluster, although more visible now, can be explained by previous analysis: point videos act as gateways to quickly navigate to other videos, in many cases with low spatiotemporal exploration.

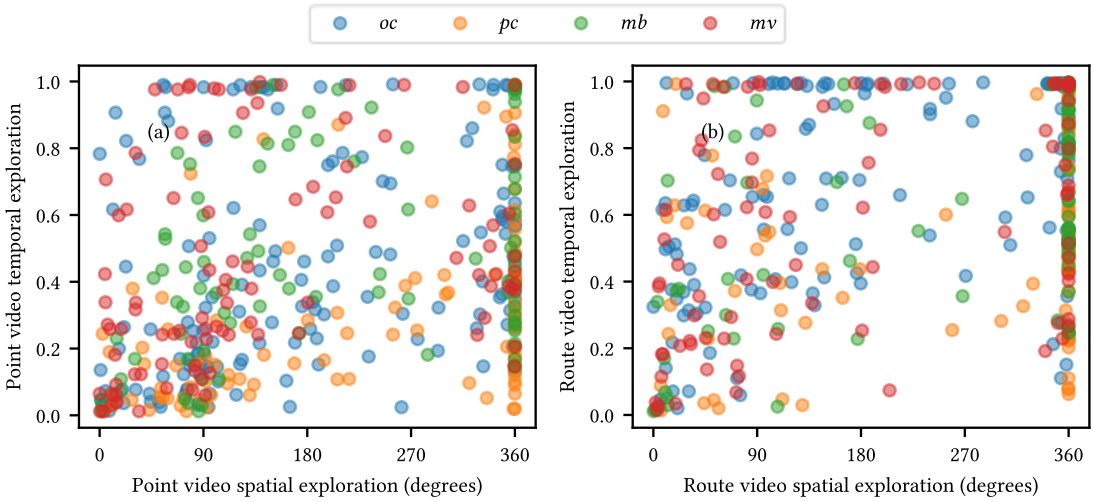

Figure 5.7. Scatter plot of spatial exploration and temporal exploration of the 846 sessions with $e_t \geq 0.01$ by video type and interface used.

The Spearman's rank correlation coefficient has been calculated between spatial and temporal exploration for each video type and device (see Table 5.3). Correlation coefficients are reported between 0.35 and 0.54, with $p < 0.001$ in all the cases. This suggests there is a statistically significant moderate correlation between temporal and spatial exploration. Correlation using immersive interfaces is similar between video types: $\rho_{p,oc} = 0.43$, $\rho_{r,oc} = 0.43$ and $\rho_{p,mv} = 0.43$, $\rho_{r,mv} = 0.44$, whereas a difference has been noted in non-immersive interfaces: $\rho_{p,pc} = 0.54$, $\rho_{r,pc} = 0.39$ and $\rho_{p,mb} = 0.35$, $\rho_{r,mb} = 0.39$; in these last cases, the *pc* interface has a considerably greater correlation in point videos than in route videos, but the *mb* interface has a greater correlation in route videos than in point

videos. We attribute the similar behavior in immersive devices to the similarity in the interface and the physical constraints on the human body movements required to perform the exploration.

Table 5.3. Correlation between spatial exploration and temporal exploration of the 846 sessions with $e_t \geq 0.01$ by video type and interface used.

Video type	Device	Spearman's ρ	p-value
point	*oc*	0.4259	< 0.001
point	*pc*	0.5381	< 0.001
point	*mb*	0.3538	< 0.001
point	*mv*	0.4087	< 0.001
route	*oc*	0.4255	< 0.001
route	*pc*	0.3932	< 0.001
route	*mb*	0.4539	< 0.001
route	*mv*	0.4421	< 0.001

Finally, we can complete our answer on RQ1 (Is spatiotemporal exploration the same across devices and video types?): there is a statistically significant moderate correlation between temporal and spatial exploration in all the cases. We observed that this correlation is similar between video types with immersive devices, while with non-immersive devices, there is variation in correlation between video types.

5.4. Dispersion of the orientation

In this section, to answer RQ2 (Is the dispersion of observed viewport centers similar across devices and video types?), we continue the analysis we began in section 3.6.6 to measure the dispersion of the orientations recorded by video type and device used. We already explained in section 3.6.6 how we obtained two measures of dispersion and that there were evident differences between video types, which were not so evident between devices (Table 3.9).

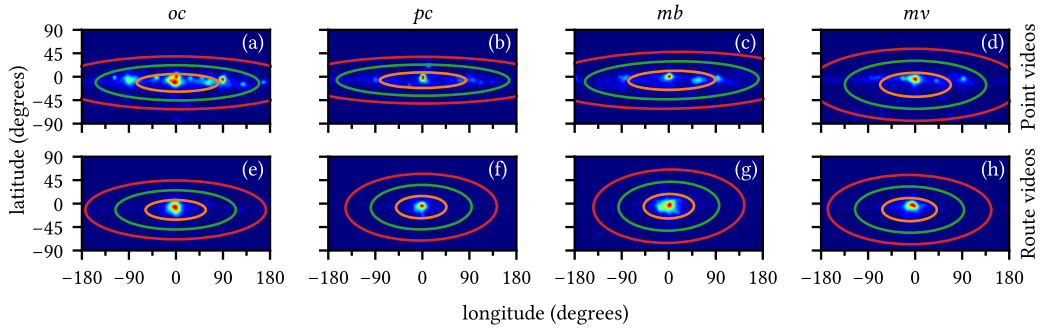

Figure 5.8. Distribution of the orientation in all the frames per video type and device used to play it.

The orientation of these frames per video type and device is plotted in Figure 5.8. The longitude is normalized in the range $[-180, 180)$. While in all the cases the orientation around (0,0) is the most common, we can notice an evident difference between point and route videos, already discussed in section 3.6.6: orientation in route videos is more concentrated than in point videos.

To test the significance of the difference in dispersion between samples and the centroids (see Table 5.4), we calculated the distribution of the distance between these values (in which s' is based (3.4)) and ran a series of Welch's t-tests between devices in the same video type and between video types with the same device, which are reported in Table 5.5. Cohen's d is reported as a measure of effect size. The null hypotheses H_0 are set that there is no difference in mean distance between the samples and their centroid between two sets of data a and b in a same group (e.g.: between *oc* and *mb* in point videos). The alternate hypotheses H_1 are set that there is a significant difference between that mean distance.

We can observe how in almost all the cases, the difference in the mean distance is significant between sets. It is only non-significant in route videos between *mb* and *mv* ($p = 0.66$). In all the other cases, the null hypotheses can be rejected at $\alpha = 0.05$. As we expected, the difference in a device between

video types (the last four rows in Table 5.5) is significant. We can observe that the effect sizes are medium or large when we compare video types, and that they are only cases in which they are medium when devices are compared are in point videos between *oc* and *mv*, *mb* and *mv*, and *pv* and *mv*.

Table 5.4. Centroids of the orientations and average distance between the orientations and these centroids grouped by video type and device.

Video type	Device	Centroid	Avg. distance
point	*oc*	(−11.71,4.02)	62.65
point	*pc*	(−6.65,9.85)	65.40
point	*mb*	(−6.91,14.71)	67.84
point	*mv*	(−15.49,2.16)	53.74
route	*oc*	(−12.04, −1.68)	37.97
route	*pc*	(−6.82, −2.57)	33.25
route	*mb*	(−5.01, −1.02)	34.58
route	*mv*	(−11.53, −9.85)	34.83

These results show us how the differences in average distance in Table 5.4 are significant in all the cases except in route videos between *mb* and *mv*, so the considerable difference between video types is significant and between interfaces it is also significant except in that case: we can observe how in point videos orientation with non-immersive interfaces are more disperse than with immersive devices, while in route videos the opposite appears to be true. These results are consistent with the difference in standard deviation of the longitude reported in section 5.2.

We should consider that the repetition of the experiments increases the chance of error type I, so if we apply a Bonferroni correction with 16 repetitions, we should consider $\alpha = 0.003$, we may no longer reject the differences in point videos between *oc* and *pc*, *mb* and *pc*, and in route videos between *mb* and *pc*.

Table 5.5. Results of the Welch's t-tests applied to the distribution of the distance between observations and their centroid, grouped by video type and device.

Group	*a*	*b*	N_a	N_b	*t*	*p*	*d*
point	*oc*	*mb*	5290	3242	-5.4421	< 0.001	-0.122
point	*oc*	*pc*	5290	2556	-2.5371	0.011	-0.063
point	*oc*	*mv*	5290	3589	10.5655	< 0.001	0.223
point	*mb*	*pc*	3242	2556	2.0499	0.040	0.055
point	*mb*	*mv*	3242	3589	14.3919	< 0.001	0.352
point	*pc*	*mv*	2556	3589	10.5447	< 0.001	0.283
route	*oc*	*mb*	15456	7689	6.5404	< 0.001	0.087
route	*oc*	*pc*	15456	10057	9.7276	< 0.001	0.121
route	*oc*	*mv*	15456	8405	6.1477	< 0.001	0.080
route	*mb*	*pc*	7689	10057	2.4703	0.014	0.037
route	*mb*	*mv*	7689	8405	-0.4401	0.660	-0.007
route	*pc*	*mv*	10057	8405	-2.9676	0.003	-0.044
oc	point	route	5290	15456	37.3012	< 0.001	0.602
pc	point	route	2556	10057	32.6584	< 0.001	0.842
mb	point	route	3242	7689	38.6724	< 0.001	0.881
mv	point	route	3589	8405	25.8534	< 0.001	0.522

With this knowledge, we can answer RQ2 (Is the dispersion of observed viewport centers similar across devices and video types?): the average dispersion between video types is significantly different in a given device (greater in point videos than in route videos) and the dispersion between devices is also significantly different except in route videos between *mb* and *mv*.

5.5. Angular movement in segments

One of the key aspects in adaptive streaming is the length of the segments of the video in which quality changes can happen. For this reason, in this section we calculate, in a segment, the distance

between the orientations in that segment and the orientation at the start of the segment, which would give us a measure of how much the user moved during that segment. This will help us answer RQ3 (Is movement performed with the same magnitude across devices and video types?).

The viewport movement of a session is defined with an arbitrary segment length. Segment lengths of $K = \{4,8,12,20\}$ frames (1, 2, 3, 5 seconds) have been used in [47]. Given a segment length $k \in K$, segments of a session s are defined as $E_s^k = \{e_i\}_{i=0}^{N-1}, N = s_{length} \cdot 4 - k + 1$ (as there are four frames per second) and each segment $e_i \in E_s^k$ contains the following frames: $e_i = \{f_{i+j}\}_{j=0}^{k-1} \in s$.

$$m(e_i) = \max_{0<j<k} d(f_0, f_j) \tag{5.3}$$

The movement in a segment (5.3) is defined as the maximum great circle distance d (3.1) between viewport centers between the first frame in the segment and any others. Note that movement in opposite directions in a segment cancels out: e.g., a user starts at an initial orientation (0,0), moves to (0,15) in the second frame and then to (0,0) again, although the total movement is 20 degrees, the maximum distance is 15 degrees.

Table 5.6. Distribution across video type and device of the 775 video sessions.

	oc	***pc***	***mb***	***mv***
point	131	91	80	90
route	127	96	76	84

A total of 775 sessions were considered in this study, as the sessions with less than 8 frames were filtered out for being too short. They are distributed across video type and device as described in Table 5.6.

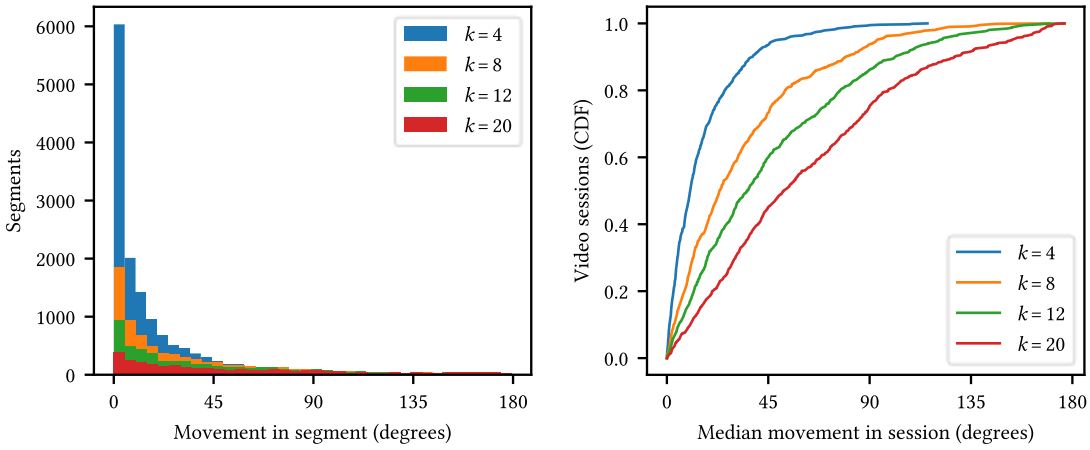

Figure 5.9. Histogram of the observed movement in segments, bins of 5 degrees (left) and CDF of the median movement across segments in each session (right).

Segments with low movement are the most observed, especially when using a short segment length, as shown in Figure 5.9 (left): it is expected that a user does not move constantly, but alternates segments in which they move and segments in which they do not. Larger segment lengths yield segments with greater movement. First, it is understandable that, while users are not perfectly still (at least when they use immersive devices, this may indeed happen when they use non-immersive devices), during most of their experience they perform little movement; second, as segment length grows, there is more room to perform movement in that window and obtain higher movement.

To perform an analysis on the sessions themselves, the movement of their segments is aggregated to obtain the median movement in a session (5.4).

$$m(s) = \text{median}\big(m(e_i)\big), e_i \in s \tag{5.4}$$

We can observe the effect of the segment length in the median movement in a session in Figure 5.9 (right): for segments of lengths $k \in \{4,8,12,20\}$ frames, 80% of the sessions showed a median movement

of less than {26,53,77,98} degrees, respectively. The increase in movement as segment length grows is expected as more movement is accounted for in each segment. This result is consistent with [47].

It can be observed that there is an almost perfect linear relationship in the first three segment lengths: in segments of length $k = 8$ frames, users move twice as much as in segments of length $k = 4$ frames and, similarly, in segments of length $k = 12$ frames, the extent of movement is three times the observed in segments of length $k = 4$. Segments of length $k = 20$ frames do not follow this rule, with a median movement of 98 degrees, less than four times the movement in segments of length $k = 4$ frames.

We can further analyze the median movement in video sessions grouping it by video type and device used to play the video, as shown in Figure 5.10. It can be observed that median movement in route video sessions is lower than in point video sessions and this difference increases as segment length grows (subfigures a-d). Across devices, median movement is similar in a given segment size in route video sessions, whereas median movement in point video sessions is more dispersed across devices.

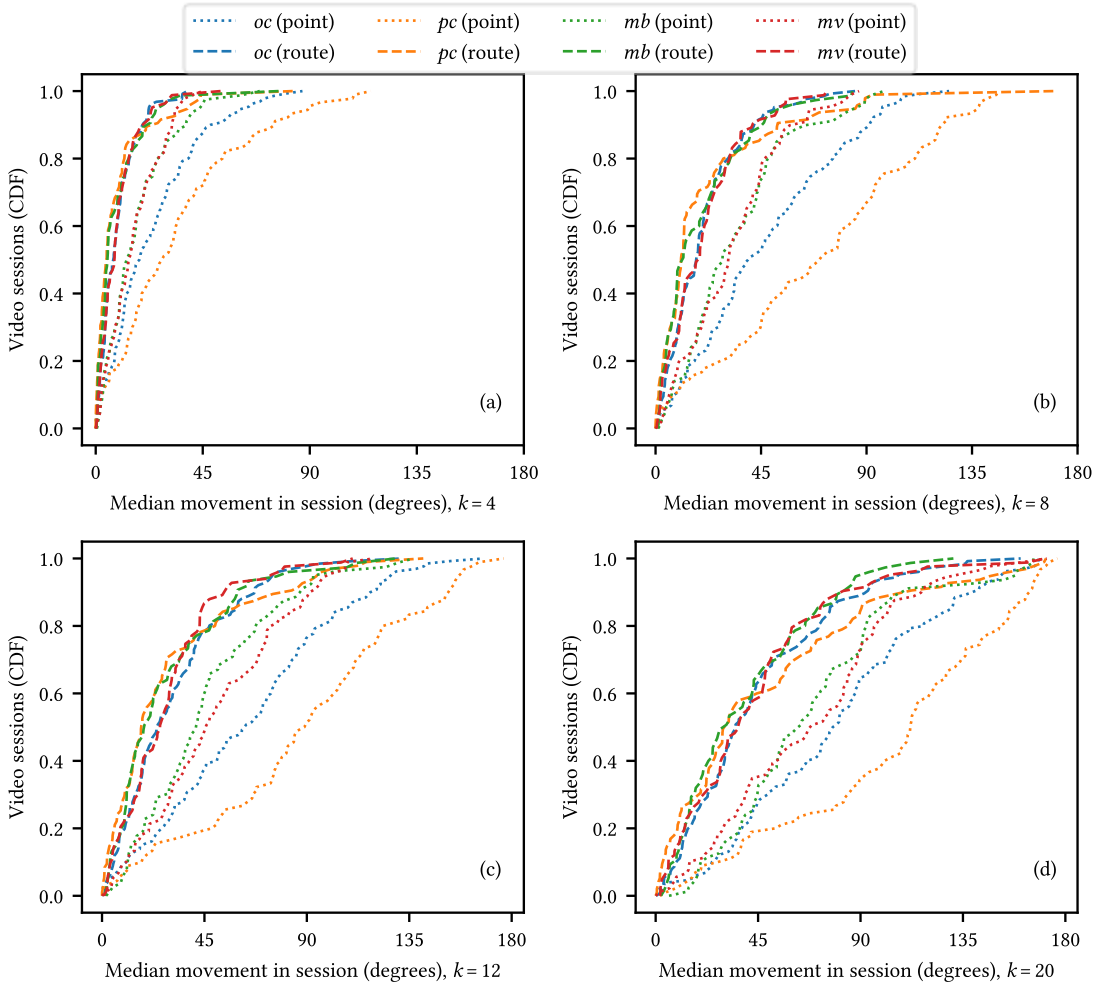

Figure 5.10. Cumulative Distribution Function of the observed median movement in video sessions, grouped by video type and device used, for segment lengths $k \in \{4,8,12,20\}$ frames in (a, b, c, d), respectively.

We can observe the median movement averaged between sessions per segment length, on videos of the same type and made with the same device, in Table 5.7. We can first notice the difference between video types: average movement in point videos is almost double than in route videos. Then, the average of the median movement is consistent across devices in route videos but, in point videos, the *pc* interface shows the greatest average movement, followed by the *oc* interface and then by both mobile interfaces (*mb* and *mv*). From this, we can extract that, when a route video is played, users tend to let themselves be carried by the camera movement and typically perform a more passive role, while

they tend to move more in point videos, in which the content does not change for them, and the kind of device used to play these point videos influences the degree of that movement.

Table 5.7. Summary of the average of the median movement in a session per segment length, video type and device used.

Video type	Device	$avg_{k=4}$	$avg_{k=8}$	$avg_{k=12}$	$avg_{k=20}$
point	*oc*	23.2680	45.6933	62.8396	78.3473
point	*pc*	33.5255	68.1666	86.7000	103.1732
point	*mb*	16.8941	33.4955	45.0474	67.5932
point	*mv*	15.6284	32.2099	49.1514	66.9930
route	*oc*	9.0558	20.1493	30.7367	43.6829
route	*pc*	8.9392	20.7932	28.6206	47.4430
route	*mb*	8.8716	18.9942	27.9672	38.7422
route	*mv*	9.1904	20.0761	28.0457	42.1717

The *pc* interface shows a much greater average degree of movement than the rest. This can be explained by the ease of performing such movement through the mouse of a desktop computer, which is not bound by the physical constraints of a device that is worn on the physical body, which is the case of the immersive interfaces. The degree of that movement can be influenced by several things: the kind of interaction (drag or cursor fusing), the mouse sensitivity and the space available on the table. Since, in this experiment, the cursor fusing technique was used with moderately sensitive mouse and a reasonable amount of space was available on the table, it is understandable why these values are higher than the rest.

The *oc* interface is next, with greater movement than the mobile interfaces in lower segment lengths but a similar degree of movement in greater segment lengths. It should be noted that although immersive interfaces (oculus and mobile visor) are constrained to the physical limitations of the human body and are expected to perform similarly, the movement described by this interface falls between that of the *pc* and mobile interfaces for lower segment lengths but approximates the movement of *mv* in higher segments lengths. A possible explanation for this can be that movement is quicker in *oc* than in the *mv* interface and that may be caused because users needed to hold their smartphone in a cardboard contraption with their own hands and, for this reason, they performed slower turns than in the *oc* interface, which is strapped on their heads and is held on place for them.

Both mobile interfaces performed similarly: it was expected that the movement in *mb* to be similar to *oc*, and has been discussed in the previous paragraph, but it was not expected that the *mb* device, a drag interface similar to *pc* performed so differently. While similar, there are key differences in the user interface of the two non-immersive devices (*pc* and *mb*): first, the *pc* interface uses the cursor fusing technique, which allows for continuous movement without the need to repeatedly drag the cursor, whereas this is not an option in *mb* and the finger needs to be repeatedly dragged on the screen; second, the small size of the screen of the smartphone plays a role on the magnitude of the movement per action, as the finger needs a surface to drag on to register that movement on the *mv* interface, which was tuned for handheld devices. For these reasons, movement in *mb* is very different than in *pc*.

A pairwise Kolmogorov-Smirnov test was performed on the sampled data from the navigation on the videos, refer Table 5.8 for the results of the test and Table 5.6 for the distribution of the 775 sessions by video type and device. By grouping the sessions by video type (column G), we can test whether the observed median movement from two devices (columns A and B) come from different underlying distributions. The null hypotheses H_0 are set that both distributions of median movement come from the same underlying distribution.

It can be observed that, for point videos, we can reject the null hypotheses at $\alpha = 0.05$ and conclude that, the movement performed using different interfaces came from different underlying distributions in many cases: between *pc* and *oc*, *mb* and *pc*, *mv* and *pc*. We can find it is not different between *mb* and *mv*. Other combinations may be significant at some values of *k*. We can infer from this that, in many cases, there is a difference in the movement between devices in point videos.

When we compare behavior in route videos, it is more difficult to reach a conclusion. We can reject H_0 between *pc* and *oc* at $k \in \{4,8\}$ and between *mv* and *pc* at $k = 8$, but in the other cases, we cannot. What we can notice is how the difference (the D statistic) is the smallest when devices of the same type of immersivity are compared (*pc* with *mb* and *oc* with *mv*).

Table 5.8. Results of the pairwise Kolmogorov-Smirnov test on the median movement in a session by segment size, video type and device used. The D statistic and the p-values are reported.

G	A	B	$D_{k=4}$	$p_{k=4}$	$D_{k=8}$	$p_{k=8}$	$D_{k=12}$	$p_{k=12}$	$D_{k=20}$	$p_{k=20}$
p	*pc*	*oc*	0.209	0.015*	0.287	0.000*	0.267	0.000*	0.353	0.000*
p	*mb*	*oc*	0.188	0.051	0.255	0.002*	0.278	0.000*	0.184	0.058
p	*mb*	*pc*	0.339	0.000*	0.467	0.000*	0.508	0.000*	0.476	0.000*
p	*mv*	*oc*	0.212	0.014*	0.262	0.001*	0.205	0.019*	0.142	0.209
p	*mv*	*pc*	0.373	0.000*	0.483	0.000*	0.470	0.000*	0.460	0.000*
p	*mv*	*mb*	0.090	0.844	0.103	0.717	0.140	0.338	0.119	0.535
r	*pc*	*oc*	0.206	0.016*	0.221	0.008*	0.172	0.068	0.132	0.267
r	*mb*	*oc*	0.167	0.122	0.182	0.073	0.146	0.238	0.138	0.294
r	*mb*	*pc*	0.104	0.697	0.116	0.566	0.094	0.807	0.125	0.480
r	*mv*	*oc*	0.057	0.998	0.075	0.912	0.090	0.762	0.069	0.949
r	*mv*	*pc*	0.192	0.063	0.210	0.033*	0.162	0.167	0.121	0.491
r	*mv*	*mb*	0.167	0.186	0.201	0.067	0.163	0.210	0.143	0.347
oc	r	p	0.435	0.000*	0.451	0.000*	0.428	0.000*	0.388	0.000*
pc	r	p	0.614	0.000*	0.593	0.000*	0.604	0.000*	0.529	0.000*
mb	r	p	0.372	0.000*	0.391	0.000*	0.359	0.000*	0.370	0.000*
mv	r	p	0.334	0.000*	0.325	0.000*	0.391	0.000*	0.358	0.000*

Finally, if we group the data by device (last four rows in Table 5.6), we can see how H_0 can be rejected in all the cases: in a given device, movement is always statistically different between video types.

We can now answer RQ3 (Is movement performed with the same magnitude across devices and video types?): the movement is not performed with the same magnitude between video types with a given device, arriving to the same conclusion as Corbillon *et al.* in [47], while between devices it is different in many cases in point videos, but it is similar in almost all cases between devices in route videos. In route videos, the smallest differences are when immersive and non-immersive devices are compared (*pc* with *mb* and *oc* with *mv*).

5.6. Tracking of trajectories

To answer RQ4 (Are trajectories followed in the same way across devices and video types?) we chose to apply a clustering algorithm to our dataset and compare its results (mainly, the average size of the clusters and the size of the largest cluster, in relative amounts) between video types and devices. We are also interested in comparing these results when devices are considered individually and all together.

Rossi *et al.* [109] proposed a method for the clustering of users navigating 360º video content to quantitatively measure how many groups of users consistently share the same behavior over time. We implemented this method, based on the Bron-Kerbosch (BK) clique-finding algorithm [110], and analyzed our dataset with it.

This algorithm identifies clusters of users that have a common viewport overlap over a time window. A distance between viewport centers $G_{th} = \pi/10 = 18°$ is reported in overlaps of at least an 80% [109] and was found suitable as a threshold for this method. As we were more interested in measuring the differences in our data than in the results of different clustering algorithms, we considered only this method.

5.6.1. Frame-based clustering

For each frame in a given video, an undirected graph, G, is built between all the visualizations available in that frame. An edge between two visualizations exists when the spherical distance between them is in the defined threshold, G_{th} (5.5), which are represented in the binary adjacency matrix, W_t.

$$w_t(s_i, s_j) = \begin{cases} 1, & \text{if } d(f_i^t, f_j^t) \leq G_{th} \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

This graph is then fed to the clique-based clustering algorithm, which finds the list of clusters, C, ordered by descending size. Each cluster $c \in C$ is a set that contains, iteratively, the observations which were cliques as defined by W_t in what remains of the graph G, after removing the observations in each successive clique [109].

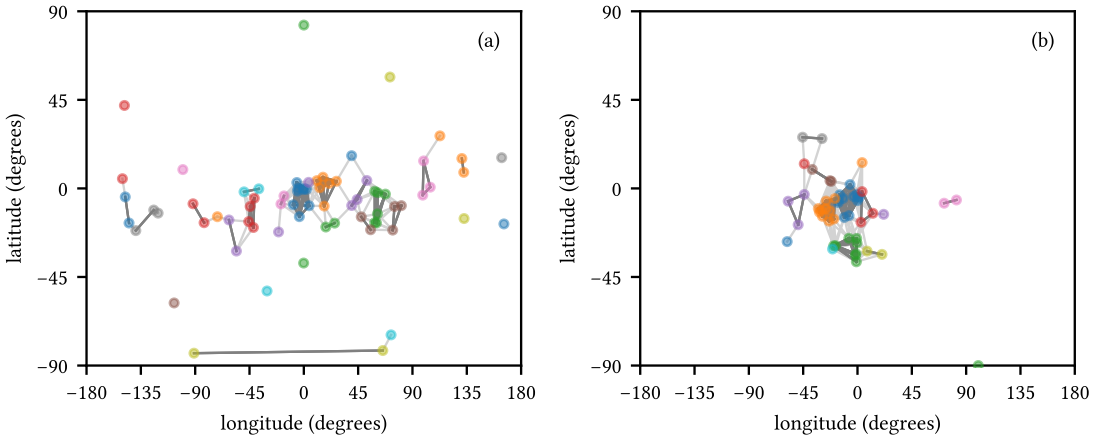

Figure 5.11. Result of the clustering in the frame 15 of the reproductions of two videos: (a) in *p1*, a point video and (b) in *r1a*, a route video, using a window of $T = 1$ frame. Markers denote individual visualizations, and their colors indicate the calculated clusters. Lines between markers denote if their distance is less than the threshold G_{th} (lighter gray) and if both markers belong to the same cluster (darker gray).

A sample of the result of the clustering method can be seen in Figure 5.11, where it is applied frame by frame to the visualizations of each video and a specific frame of two videos (a point video (a) and a route video (b)) and a specific frame of each is represented in the graph. It can be observed that observations are much more disperse in the point video than in the route video, but it is difficult to quantify the number of clusters, the average observations per cluster or the ratio of observations that lie in the main cluster.

We define the number of observations (N) in a specific point in time as the total amount of observations of which we have recorded their position for the last τ frames (in the case of a frame window of $T = 1$ frame, $\tau = 1$ frame).

With these observations, we obtain the number of clusters (K) in a specific point in time (5.6).

$$K = |C| \tag{5.6}$$

The size of the clusters (S) is the number of observations in each of the K clusters. We obtain its average (5.7) and standard deviation (5.8) values expressed as a fraction of N per each cluster.

$$S_{avg} = \frac{\sum_{i=0}^{K} \frac{|c_i|}{N}}{K} = \frac{1}{K} \tag{5.7}$$

$$S_{std} = \sqrt{\frac{\sum_{i=0}^{K} \left(\frac{|c_i|}{N} - S_{avg} \right)^2}{K - 1}} \tag{5.8}$$

Finally, the main cluster (c_0) is the one that has the biggest size. We consider its size (M) a particular case to be discussed independently, also expressed as a fraction of N. As the algorithm lists the clusters in decreasing size, this is effectively the size of the first cluster (5.9).

$$M = \frac{|c_0|}{N} \tag{5.9}$$

Figure 5.12 and Figure 5.13 show these values for the two videos we previously discussed. It can be observed that the amount of clusters (K) quickly rises during the first 4 seconds, as users spread from their initial position, and then it decreases as the number of observations (N) decreases, more noticeably in the case of the point video (Figure 5.12 (a)) than in the route video (Figure 5.12 (b)). The average size of the clusters (S) quickly drops as the number of clusters (K) grows during the first seconds and slowly rises as both N and K decrease, but its standard deviation slowly grows after the first seconds in both videos (Figure 5.13). The size of the main cluster (M) also describes a quick descent in the few first seconds until it settles around 0.2 in the point video (Figure 5.13 (a)), while it varies more in the route video, between 0.2 and 0.4 (Figure 5.13 (b)).

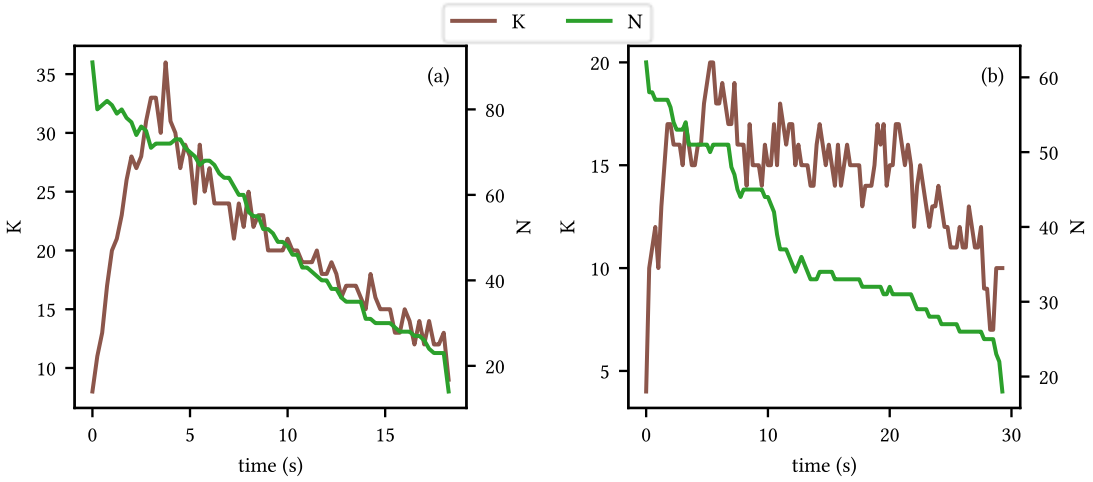

Figure 5.12. Plot of the number of clusters (K) and the number of observations (N) over time of the reproductions of a point video, *p1* (a), and a route video, *r1a* (b), using a window of $T = 1$ frame.

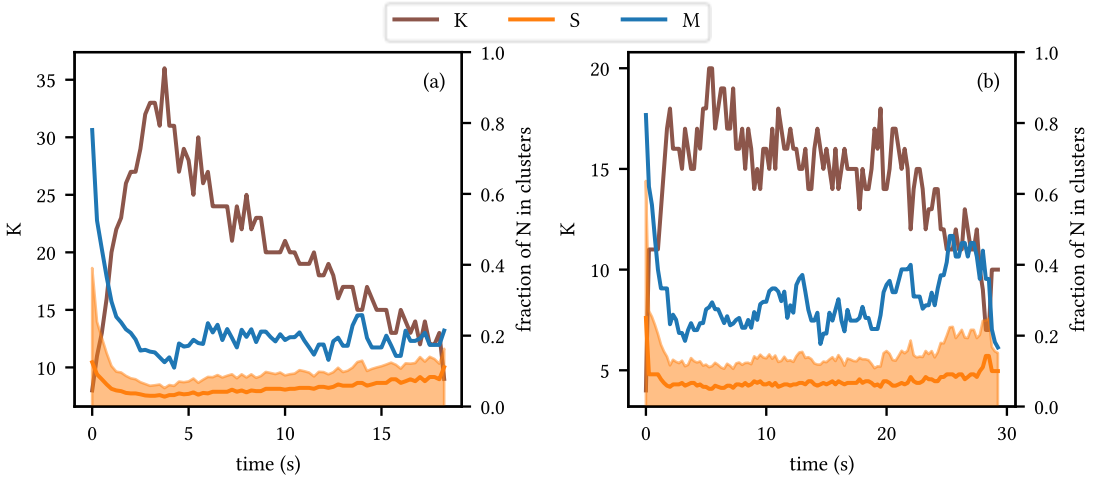

Figure 5.13. Plot of the number of clusters (K), average and standard deviation of the size of the clusters as a fraction of N (S) and size of the main cluster as a fraction of N (M) over time of the reproductions of a point video, *p1* (a), and a route video, *r1a* (b), using a window of $T = 1$ frame.

Keeping in mind that this is the clustering of the visualizations of two specific videos, we analyze and discuss these figures: first, it is expected that the number of clusters (K) quickly grows in the first few seconds, as users usually start their visualizations from the same orientation; second, it is also expected

that the amount of clusters (K) decreases as the number of available visualizations (N) decreases, as less visualizations mean less possible clusters to make, overall; third, it is expected that the average size of the clusters (S_{avg}) and the amount of size of the main cluster (M) are independent to the number of visualizations (N), as it is expressed as a fraction of that value, so it should remain constant; fourth, the average size of the clusters (S_{avg}) and the size of the main cluster (M) start in a high value for the same reason as the first point.

Furthermore, we observe that there is a difference between the visualizations of both videos: while the amount of visualizations (N) follow a similar, descendent, pattern, the amount of clusters (K) in the point video drops following the descent of N, whereas in the route video, K has remained stable during most of the drop in N, so although the average size of the clusters (and thus the average number of clusters per user) is similar and even the size of the main cluster is larger in the route video than in the point video, when part of the users stopped watching the video, those that kept watching it maintained the number of clusters, so comparatively, visualizations in the point video are disperse and the number of clusters (K) tends to follow the evolution of the number of visualizations (K), whereas in the route video, visualizations are fixated around the central region of interest, which it alone warrants for a minimum amount of clusters, as seen in Figure 5.11 (b).

Additionally, the average size of the clusters (S_{avg}) is similar between both videos, but its standard deviation is slightly larger in the route video, so clusters in the route video varied more in size. This is reasonable when we observe the size of the main cluster (M): with a median of 0.1875 in the point video and of 0.2835 in the route video, we can see that more visualizations were in the same place in the route video, resulting in a highly populated area, with larger clusters and those visualizations that rarely leave this area formed smaller clusters, whereas in the point video, the strength of this central area is not as strong, so visualizations navigate the space more freely. These results are consistent with [109]: they found between 12.10% and 25.90% of users belonged to the main cluster in the *timelapse* video (analogous to our *p1*, a point video) and between 30.50% and 57.70% belonged to the main cluster in the *rollercoaster* video (analogous to our *r1a*, a route video).

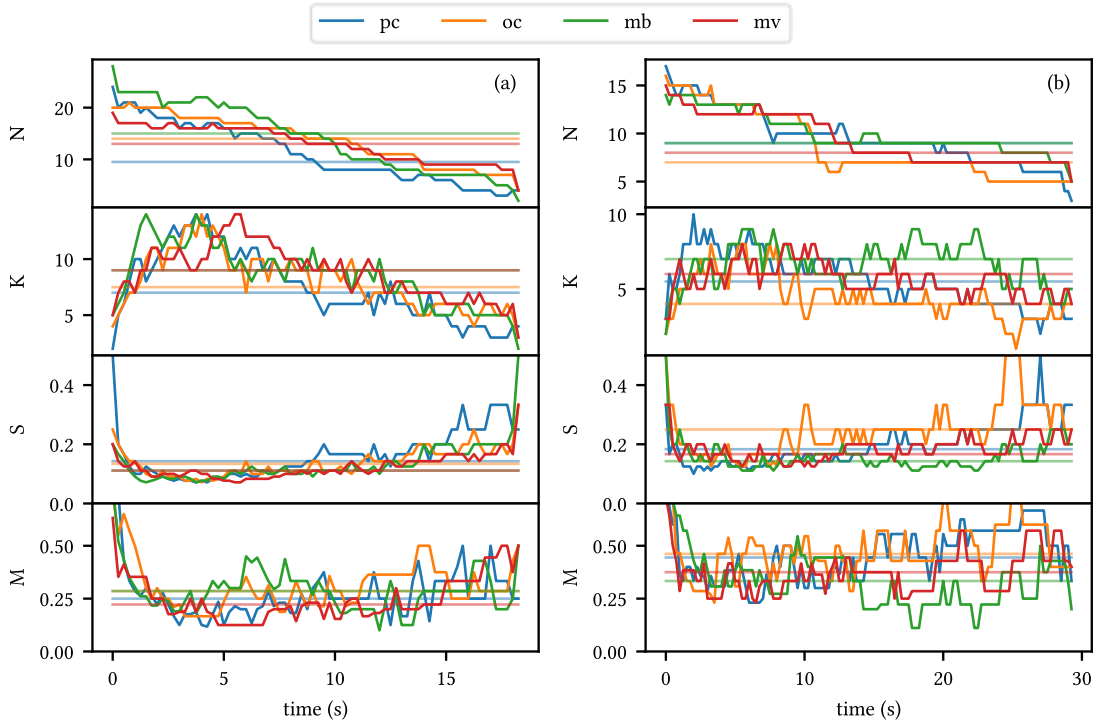

Figure 5.14. Result of the application of the clustering method in the reproduction data of two videos, using sessions grouped per device that played that video, with a trajectory window of $T = 1$ frame: (a) in *p1*, a point video and (b) in *r1a*, a route video. The number of visualizations (N), number of clusters (K), average size of clusters (S) and ratio of users in the main cluster (M) are reported over time, as well as their median value across time (horizontal lines).

If we group the visualizations per device that played each video, as in Figure 5.14, we first observe that values of N are effectively divided between the four different device interfaces. Despite this, we can observe a similar behavior to that of when the four devices were grouped together in the evolution of K over time, which was discussed before: in the point video (Figure 5.14 (a)), the descent follows the descent in N whereas that relationship is not as strong in the route video (Figure 5.14 (b)): in the point video, all interfaces follow a similar evolution of K over time, but in the route video, the *mb* interface even grows in K during the latter part of the video. Regarding the average size of the clusters (S_{avg}), we can observe that in the point video (Figure 5.14 (a)), its evolution is consistent across interfaces, with very similar median values, while in the route video (Figure 5.14 (b)), we can see that the *mb* interface shows less S_{avg} than the others, as expected by what we just previously commented and *oc* shows more S_{avg}, possibly due to the drops in N, so the users that remained playing the video were looking at the region of interest. In Table 5.9 we can observe that median S_{avg} is fairly consistent per video in all devices except in the route video played with the *oc* device, while S_{std} is higher in both cases when played with the *oc* device. Finally, the size of the main cluster (M) shows a greater variance but a similar evolution in the point video (Figure 5.14 (a)) for all the interfaces and in the route video (Figure 5.14 (b)) except for *mb*, which has lower M in the latter part, as expected. The median M is greater in the point video for *oc* and *mb*, while it is greater for *pc* and *oc* in the route video. In any case, M is greater in the route video for any interface than any interface in the point video.

Table 5.9. Median values in Figure 5.14: number of visualizations (N), number of clusters (K), average size of clusters (S_{avg}), standard deviation of the size of clusters (S_{std}) and ratio of users in the main cluster (M), for different devices in the reproductions of *p1*, a point video and *r1a*, a route video using a trajectory window of $T = 1$ frame.

Video	Type	Device	N	K	S_{avg}	S_{std}	M
p1	point	*pc*	8.5	8.0	0.1250	0.0274	0.2000
p1	point	*oc*	14.0	9.5	0.1056	0.0638	0.2500
p1	point	*mb*	14.5	10.0	0.1000	0.0583	0.2554
p1	point	*mv*	13.0	10.0	0.1000	0.0475	0.2000
r1a	route	*pc*	9.0	6.0	0.1667	0.1286	0.4222
r1a	route	*oc*	7.0	4.0	0.2500	0.1368	0.4286
r1a	route	*mb*	9.0	7.0	0.1429	0.0819	0.3077
r1a	route	*mv*	8.0	6.0	0.1667	0.0927	0.3333

The clustering method was applied to all the videos and its results are reported in Table 5.10 in those cases in which $N \geq 10$. The aggregated values were obtained from all the individual samples through all the videos, not aggregated per video and then again across videos. We can observe how the average size of the clusters (S_{avg}) tends to be higher in average in route videos than in point videos, and the same can be said of the size of the main cluster (M).

Table 5.10. Average and standard deviation across all the videos of the results of the results of the frame-based clustering method with $T = 1$ when $N \geq 10$, grouped by video type and device.

Video			Average				Standard deviation			
type	Device	L	N	S_{avg}	S_{std}	M	N	S_{avg}	S_{std}	M
p	*all*	393	36.05	0.07	0.05	0.21	22.15	0.03	0.04	0.1
p	*oc*	242	16.6	0.11	0.07	0.26	5.78	0.04	0.06	0.12
p	*pc*	88	14.56	0.15	0.1	0.3	4.6	0.12	0.12	0.19
p	*mb*	107	16.37	0.11	0.08	0.27	4.69	0.03	0.05	0.12
p	*mv*	172	14.61	0.12	0.08	0.27	4.14	0.06	0.07	0.15
r	*all*	1690	22.1	0.11	0.11	0.38	9.18	0.06	0.06	0.13
r	*oc*	474	13.38	0.17	0.15	0.45	2.43	0.09	0.08	0.15
r	*pc*	69	11.65	0.17	0.14	0.42	2.01	0.07	0.1	0.15
r	*mb*	68	11.56	0.16	0.13	0.4	1.53	0.05	0.07	0.13
r	*mv*	105	11.5	0.19	0.17	0.45	1.08	0.06	0.1	0.17

We have identified some differences in the results of the clustering method between video types and devices. We leveraged Welch's t-tests to evaluate the significance of these differences. Null hypotheses

H_0 are set as there is no difference in mean S_{avg} or M between the results of applying the clustering method to reproductions of two different devices in the same video type or in two different video types with the same device. Note that here we only considered frames in which $N \geq 10$, so as not to take into account frames in which there were too few visualizations available. We can observe these results in Table 5.11, with p-values lower than $\alpha = 0.05$ are marked with an asterisk (*).

Table 5.11. Results of the Welch's t-tests on S_{avg} and M between devices (A, B) grouped by video type (G) and between video types (A, B) grouped by device (G) for $T = 1$, including the effect sizes (d) and the length of the samples (L).

G	A	B	L_A	L_B	t_{Savg}	p_{Savg}	d_{Savg}	t_M	p_M	d_M
p	*all*	*oc*	393	242	-13.4693	0.0*	-1.1754	-6.0319	0.0*	-0.5164
p	*all*	*pc*	393	88	-5.6454	0.0*	-1.2553	-4.1325	0.0001*	-0.7167
p	*all*	*mb*	393	107	-10.6344	0.0*	-1.2279	-5.2475	0.0*	-0.6286
p	*all*	*mv*	393	172	-11.6677	0.0*	-1.316	-4.8755	0.0*	-0.5153
p	*oc*	*pc*	242	88	-2.4953	0.0143*	-0.4608	-1.4271	0.1563	-0.218
p	*oc*	*mb*	242	107	0.5746	0.5661	0.0622	-0.6647	0.507	-0.0762
p	*oc*	*mv*	242	172	-2.2922	0.0226*	-0.2403	-0.2713	0.7863	-0.0279
p	*pc*	*mb*	88	107	2.6475	0.0095*	0.4145	0.9456	0.346	0.1423
p	*pc*	*mv*	88	172	1.5932	0.1141	0.2597	1.1831	0.2388	0.1691
p	*mb*	*mv*	107	172	-2.5815	0.0104*	-0.2859	0.343	0.7319	0.0401
r	*all*	*oc*	1690	474	-12.1422	0.0*	-0.813	-9.0262	0.0*	-0.5235
r	*all*	*pc*	1690	69	-5.9732	0.0*	-0.9132	-2.0193	0.0472*	-0.2918
r	*all*	*mb*	1690	68	-7.1978	0.0*	-0.8387	-1.4175	0.1606	-0.1766
r	*all*	*mv*	1690	105	-13.8989	0.0*	-1.3868	-4.5011	0.0*	-0.5891
r	*oc*	*pc*	474	69	0.1124	0.9107	0.0122	1.6619	0.1	0.211
r	*oc*	*mb*	474	68	0.7832	0.4349	0.069	2.7796	0.0065*	0.3131
r	*oc*	*mv*	474	105	-3.6439	0.0003*	-0.2928	-0.3674	0.7139	-0.0423
r	*pc*	*mb*	69	68	0.4573	0.6483	0.078	0.6229	0.5344	0.1063
r	*pc*	*mv*	69	105	-2.5569	0.0118*	-0.4168	-1.5841	0.1152	-0.2393
r	*mb*	*mv*	68	105	-3.6804	0.0003*	-0.5661	-2.3718	0.0188*	-0.3473
all	p	r	393	1690	-20.2978	0.0*	-0.7971	-28.948	0.0*	-1.4138
oc	p	r	242	474	-11.0646	0.0*	-0.7017	-17.427	0.0*	-1.2832
pc	p	r	88	69	-1.2971	0.1967	-0.1966	-4.3725	0.0*	-0.6816
mb	p	r	107	68	-7.0493	0.0*	-1.202	-6.6702	0.0*	-1.0496
mv	p	r	172	105	-9.8883	0.0*	-1.23	-9.3339	0.0*	-1.1946

In point videos, we can reject H_0 when we compare all the devices are considered together with any of them with a large effect size on S_{avg} and a medium effect size on M: all devices behave differently to the group of all the devices together; in the other cases, we can reject H_0 on S_{avg} in select combinations: between *oc* and *pc*, *oc* and *mv*, *pc* and *mb*, *mb* and *mv*; we cannot reject H_0 on S_{avg} in the rest of combinations nor in any combination on M; this means that the average relative size of the main cluster (M) is not significantly different between devices: approximately the same relative amount of viewers were on the main cluster.

In route videos, something similar happens when all the devices are considered, except when they are compared with *mb*, in which the difference in M is not significant; in the other cases, we can reject H_0 on S_{avg} between *oc* and *mv*, *pc* and *mv*, *mb* and *mv*, and we can reject H_0 on M between *oc* and *mb*, *mb* and *mv*. In this case, there were some differences between devices in the size of the main cluster which were not significant in point videos.

When we compare video types grouped by device, we can observe how we can reject all H_0 except on S_{avg} in *pc*. This means that the average size of clusters is different between video types with all the devices with at least medium effect sizes, except in *pc*, which shows a similar average size of clusters. In any case, the size of the main cluster is significantly different in all the cases, so the relative number of viewers in the main area of interest is larger in route videos than in point videos.

As these t-tests were repeated 25 times for these combinations of variables, there is an increased chance of type I errors. Under Bonferroni's correction, a significance level of $\alpha = 0.002$ should be

considered. Notably, the previously significant differences between one device and all of them are still significant, and the differences between point and route videos are still significant.

5.6.2. Trajectory-based clustering

To properly analyze trajectories, a time window is defined. During this time T, the adjacency between two nodes is compared to a minimum threshold value τ. Two nodes will be connected if they have been adjacent, at least, during τ of the last T frames, creating the affinity matrix that is then used in the clique-based algorithm (5.10).

$$a_t(s_i, s_j) = \begin{cases} 1, & \text{if } \sum_{u=\min(0,t-T+1)}^{t} w_t(s_i, s_j) \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{5.10}$$

This introduces two special considerations: first, it is impossible that there are connected nodes in the first $\tau - 1$ frames, as enough data to pass the threshold is only available at frame τ and, when it does, this condition is so strong (as nodes need to be connected in all τ of τ frames) in the beginning, but it quickly relaxes as T frames are available; and second, as sessions can start and end in any point in the video timeline, we had to be careful to only consider adjacencies between nodes in which both of them existed in at least τ frames.

Table 5.12. Relation between the size of the trajectory window, T, and the threshold, τ, applied to the clustering method.

T	1	4	8	12	20
τ	1	3	6	9	15

In [109], a trajectory window of $T = 3s$ and threshold $\tau = 1.8s$ is used. To maintain consistency with previous analysis and because our data is sampled four times per second, we express our windows in discrete number of frames, according to Table 5.12. In this case, we consider five window sizes, including one with $T = \tau = 1$ frame (this is the previously discussed frame-based clustering) and threshold values are calculated as $0.75 \cdot T$, so the window with $T = 12$ frames (3 seconds) is considered with threshold of $\tau = 9$ frames (2.25 seconds).

Table 5.13. Median values in Figure 5.15: number of visualizations (N), number of clusters (K), average size of clusters (S_{avg}), standard deviation of the size of clusters (S_{std}) and ratio of users in the main cluster (M), for different trajectory windows (1, 4, 8, 12, 20 frames) in the reproductions of *p1*, a point video and *r1a*, a route video.

Video	Type	T	N	K	S_{avg}	S_{std}	M
p1	point	1	51.5	20.0	0.0500	0.0431	0.1875
p1	point	4	50.0	26.0	0.0384	0.0327	0.1642
p1	point	8	49.0	32.0	0.0313	0.0230	0.1304
p1	point	12	48.0	32.0	0.0313	0.0234	0.1273
p1	point	20	46.0	35.0	0.0286	0.0200	0.1154
r1a	route	1	33.5	15.0	0.0667	0.0697	0.2835
r1a	route	4	33.0	18.0	0.0556	0.0581	0.2500
r1a	route	8	33.0	20.0	0.0500	0.0455	0.2121
r1a	route	12	33.0	20.0	0.0500	0.0433	0.2075
r1a	route	20	32.0	21.0	0.0476	0.0391	0.1875

After running the clustering method on these two videos for the time windows as specified in Table 5.12, we can observe in Figure 5.15 the effect of the size of the time window in K, S and M in the visualizations of the previously discussed videos, while median values, plotted horizontally in Figure 5.15, are specified in Table 5.13

First, N is plotted in the first row: we can observe the descent of visualizations over time in the two videos and how N gets slightly lower as T grows: this is because for increasing values of T, the first τ frames worth of visualizations cannot be considered, which have greater values of N and also because of visualizations that start late or end early, with less than τ frames available at a certain point. In any case, values of N are very similar across window lengths.

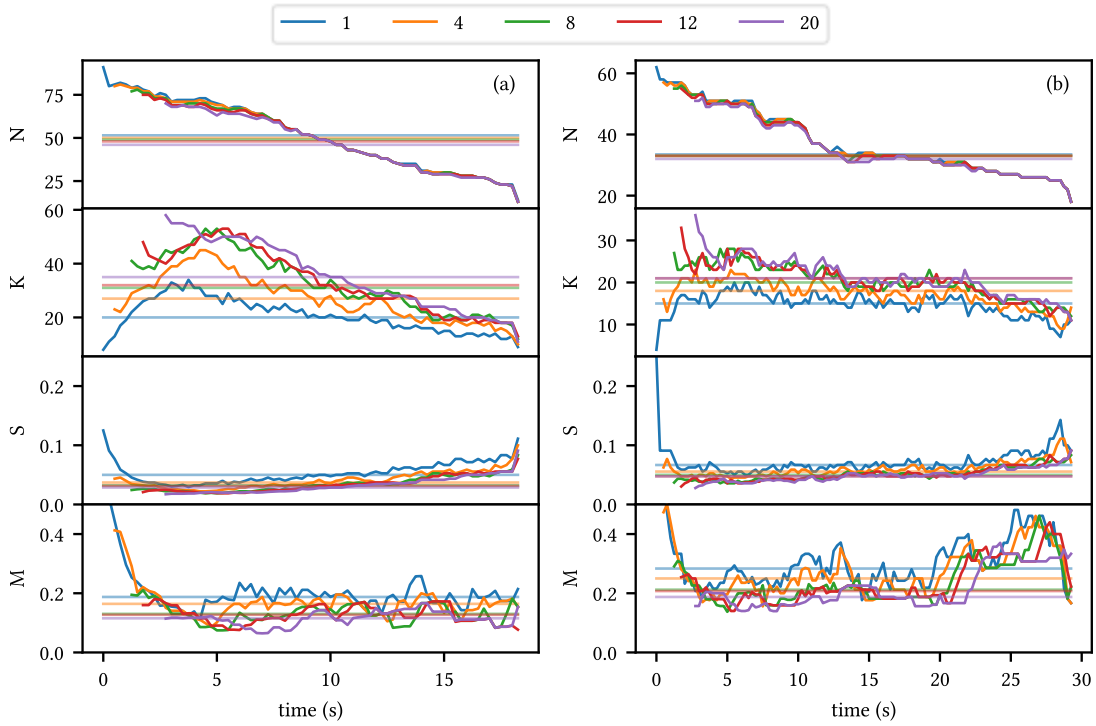

Figure 5.15. Result of the application of the clustering method in the reproduction data of two videos, using sessions with all devices that played that video, for different trajectory windows (1, 4, 8, 12, 20 frames): (a) in *p1*, a point video and (b) in *r1a*, a route video. The number of visualizations (N), number of clusters (K), average size of clusters (S) and ratio of users in the main cluster (M) are reported over time, as well as their median value across time (horizontal lines).

The absolute number of clusters (K), in the second row, varies more across window sizes: we can observe that, for smaller values of T, it grows from a minimum value to the maximum in the first few seconds, as we discussed previously, but for larger values of T, it already begins at the maximum value. This is explained by the skipping of the first τ worth of data, in which visualizations are less disperse and, thus, less clusters are formed. In any case, for all the duration of the videos, the number of clusters is greater when T is larger. This is because a stronger trajectory (forced by τ) is necessary to create a relation of adjacency.

The average size of the clusters (S_{avg}) is plotted in the third row: $T = 1$ shows the quick descent from its maximum (caused by the minimum K discussed before), but other window sizes perform similarly, with a slight grow towards the end of the video. As the number of visualizations (N) is relatively constant across window sizes and the number of clusters (K) grows with the size of the trajectory (T), the average size of the clusters (S_{avg}) turns smaller. Between video types, values of S_{avg} are similar.

The standard deviation of the size of the clusters (S_{std}), see Table 5.13, also turns smaller as the size of the trajectory (T) grows, but this time there is a difference between video types: cluster sizes are more disperse in the route video than in the point video.

Finally, the size of the main cluster (M) is plotted in the fourth row and shows a similar behavior in the first portion of the videos, dropping from its maximum value in smaller window sizes, and also smaller values as the window size grows, but there are two key differences between both video types: first, M is larger in the route video, as most of the visualizations are in the same area of the 360-degree video, while in the point video, the visualizations are more disperse and will not belong to the main cluster. This also helps understand the difference in S_{std}. Second, the relative size of the main cluster remains more or less constant for all the duration of the point video, while it shows a rise in the latter part of the route video, getting close to the 40% of the visualizations in the main cluster, as the number of remaining visualizations drops and most of them are in the area of interest.

We chose $T = 12$ frames (3 seconds), with $\tau = 9$ frames, to break down the results of the clustering method across devices. We can observe in Figure 5.16 the evolution of N, K, S_{avg} and M over time and in Table 5.13 their median values.

Table 5.14. Median values in Figure 5.16: number of visualizations (N), number of clusters (K), average size of clusters (S_{avg}), standard deviation of the size of clusters (S_{std}) and ratio of users in the main cluster (M), for different devices in the reproductions of *p1*, a point video and *r1a*, a route video using a trajectory window of $T = 12$ frames (3 seconds).

Video	Type	Device	N	K	S_{avg}	S_{std}	M
p1	point	pc	8.0	8.0	0.1250	0.0000	0.1818
p1	point	oc	14.0	10.0	0.1000	0.0499	0.2143
p1	point	mb	13.0	10.0	0.1000	0.0392	0.2000
p1	point	mv	13.0	11.0	0.0909	0.0333	0.1667
r1a	route	pc	9.0	6.0	0.1667	0.0840	0.3333
r1a	route	oc	7.0	5.0	0.2000	0.1155	0.4000
r1a	route	mb	9.0	8.0	0.1250	0.0589	0.2500
r1a	route	mv	8.0	7.0	0.1429	0.0630	0.2857

The evolution of N is very similar to that of when a time window with $T = 1$ frame is applied. The number of clusters (K) is also very similar, but instead of starting with very low values, as in $T = 1$, it already starts at high values, as the first three seconds worth of exploration, in which the users leave their starting position, cannot be considered in $T = 12$.

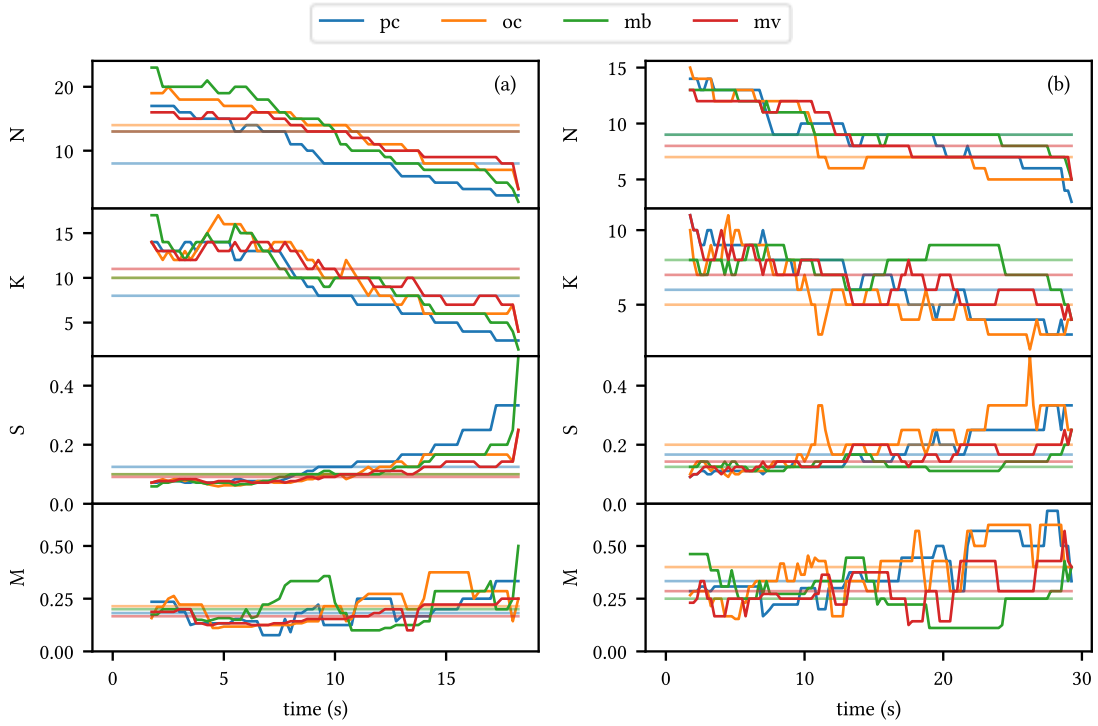

Figure 5.16. Result of the application of the clustering method in the reproduction data of two videos, using sessions grouped per device that played that video, with a trajectory window of $T = 12$ frames (3 seconds): (a) in *p1*, a point video and (b) in *r1a*, a route video. The number of visualizations (N), number of clusters (K), average size of clusters (S) and ratio of users in the main cluster (M) are reported over time, as well as their median value across time (horizontal lines).

We can also appreciate, in the point video (Figure 5.16 (a)), the correlated descent of K with N, which is not that evident in the route video (Figure 5.16 (b)), as discussed previously. This time we can observe less spikes in the evolution of K, as the trajectory window contributes to smooth the formation of clusters.

Regarding S_{avg}, we can observe it is closely related to the evolution of K in both cases, and is very evident when the route video (Figure 5.16 (b)) is played with the *oc* device: two drops in K, one in the middle of the reproduction and another in the end, cause a large growth in S_{avg} at these instants. As we can observe in Table 5.14, S_{avg} is consistent between devices in each video, except when the route video is played with *oc*, as what happened before. With $T = 12$, clusters are more uniform, as S_{std} is lower than with $T = 1$, although the *oc* device has a high deviation, especially when the route video was played, probably due to the drop in N, as many of the observers remained in the main cluster, resulting in a great difference between cluster sizes.

Finally, we can observe the evolution of the size of the main cluster (M), which is also smoother when $T = 12$ that it was when $T = 1$. We can appreciate a growth in the latter part of the route video, as N decreases, as it is more likely that the remaining observers are in the area of interest, except in the case of the *mb* device, as discussed previously.

The clustering method was applied to all the videos and its results are reported in Table 5.15 in those cases in which $N \geq 10$. The aggregated values were obtained from all the individual samples through all the videos, not aggregated per video and then again across videos. We can observe how the average size of the clusters (S_{avg}) tends to be higher in average in route videos than in point videos, and the same can be said of the size of the main cluster (M), very similar to the results of the frame-based method ($T = 1$).

Table 5.15. Average and standard deviation across all the videos of the results of the results of the trajectory-based clustering method with $T = 12$ when $N \geq 10$, grouped by video type and device. The number of samples is also reported (L).

Video			**Average**				**Standard deviation**			
type	**Device**	**L**	***N***	S_{avg}	S_{std}	***M***	***N***	S_{avg}	S_{std}	***M***
p	*all*	351	31.84	0.05	0.02	0.13	18.44	0.03	0.01	0.05
p	*oc*	199	15.08	0.08	0.02	0.15	4.34	0.02	0.02	0.06
p	*pc*	45	13.73	0.08	0.02	0.15	2.46	0.01	0.01	0.05
p	*mb*	83	15.37	0.09	0.04	0.19	3.7	0.02	0.02	0.06
p	*mv*	122	14.22	0.09	0.04	0.18	3.56	0.03	0.03	0.08
r	*all*	1589	21.67	0.09	0.08	0.31	8.66	0.03	0.03	0.1
r	*oc*	432	13.33	0.13	0.11	0.37	2.37	0.04	0.06	0.15
r	*pc*	41	11.73	0.12	0.07	0.29	1.63	0.01	0.02	0.05
r	*mb*	49	11.43	0.13	0.09	0.34	1.21	0.02	0.03	0.07
r	*mv*	84	11.29	0.15	0.12	0.38	0.82	0.04	0.08	0.17

To evaluate the significance of the differences in the results of the trajectory-based clustering method between devices and video types, we used a series of Welch's t-tests. Null hypotheses H_0 are set as there is no difference in mean S_{avg} or M between the results of applying the clustering method to reproductions of two different devices in the same video type or in two different video types with the same device. A trajectory window of $T = 12$ frames (3 seconds) was chosen and as with the analysis on the frame-based approach, we only considered frames in which $N \geq 10$, so as not to consider frames in which there were too few visualizations available. We can observe these results in Table 5.16, with p-values lower than $\alpha = 0.05$ are marked with an asterisk (*).

In point videos, we can reject H_0 when we compare the case when all the devices are considered together with any of them: all devices behave differently to the group of all the devices together, with large effect sizes on S_{avg} and mixed effect sizes on M; in the other cases, we can reject H_0 on S_{avg} between *oc* and *mb*, *oc* and *mv*, *pc* and *mv*; we cannot reject H_0 on S_{avg} in the rest of combinations, while on M we can reject H_0 in these cases plus between *pc* and *mb*. While in a frame-based clustering, differences in M were not significant in point videos, they were with a window of $T = 12$ frames.

In route videos, H_0 can also be rejected in all the cases when all the devices are compared with any of them, with large effect sizes on S_{avg} and mixed effect sizes on M; in the other cases, we can reject H_0 on S_{avg} in all the cases but between *oc* and *mb* and H_0 can be rejected on M in all the cases except between *oc* and *mv*, *mb* and *mv*. In this case, there are some differences between devices in the size of the main cluster which were not significant in point videos.

Table 5.16. Results of the Welch's t-tests on S_{avg} and M between devices (A, B) grouped by video type (G) and between video types (A, B) grouped by device (G) for $T = 12$, including the effect sizes (d) and the length of the samples (L).

G	A	B	L_A	L_B	t_{Savg}	p_{Savg}	d_{Savg}	t_M	p_M	d_M
p	*all*	*oc*	351	199	-15.2401	0.0*	-1.2754	-3.8394	0.0001*	-0.3561
p	*all*	*pc*	351	45	-12.1377	0.0*	-1.2721	-3.4609	0.001*	-0.5425
p	*all*	*mb*	351	83	-16.763	0.0*	-1.5244	-8.7102	0.0*	-1.2558
p	*all*	*mv*	351	122	-14.1958	0.0*	-1.5	-5.9237	0.0*	-0.8013
p	*oc*	*pc*	199	45	-0.3029	0.7627	-0.0405	-0.9582	0.3411	-0.1418
p	*oc*	*mb*	199	83	-2.7086	0.0073*	-0.3154	-5.7634	0.0*	-0.7887
p	*oc*	*mv*	199	122	-2.8901	0.0042*	-0.3506	-3.4161	0.0008*	-0.4306
p	*pc*	*mb*	45	83	-1.906	0.0597	-0.3471	-3.8722	0.0002*	-0.6594
p	*pc*	*mv*	45	122	-2.2417	0.0266*	-0.3069	-2.0806	0.0393*	-0.284
p	*mb*	*mv*	83	122	-0.6564	0.5123	-0.0853	1.6201	0.1068	0.2185
r	*all*	*oc*	1589	432	-20.0478	0.0*	-1.3077	-8.1773	0.0*	-0.5391
r	*all*	*pc*	1589	41	-15.8176	0.0*	-1.0747	2.1324	0.0379*	0.1633
r	*all*	*mb*	1589	49	-18.7918	0.0*	-1.6331	-3.3116	0.0017*	-0.3452
r	*all*	*mv*	1589	84	-15.5419	0.0*	-2.2924	-3.7136	0.0004*	-0.6447
r	*oc*	*pc*	432	41	4.0272	0.0001*	0.276	7.6479	0.0*	0.5547
r	*oc*	*mb*	432	49	-1.7457	0.0833	-0.1391	2.0748	0.0406*	0.1847
r	*oc*	*mv*	432	84	-5.5789	0.0*	-0.6558	-0.4012	0.689	-0.0528
r	*pc*	*mb*	41	49	-5.3586	0.0*	-1.099	-4.0912	0.0001*	-0.8351
r	*pc*	*mv*	41	84	-7.9278	0.0*	-1.1192	-4.3207	0.0*	-0.6067
r	*mb*	*mv*	49	84	-4.2609	0.0*	-0.6357	-1.6015	0.1118	-0.2393
all	p	r	351	1589	-22.899	0.0*	-1.2632	-49.2256	0.0*	-1.8701
oc	p	r	199	432	-18.9533	0.0*	-1.3106	-27.4594	0.0*	-1.7738
pc	p	r	45	41	-12.1444	0.0*	-2.5911	-13.3703	0.0*	-2.8871
mb	p	r	83	49	-15.43	0.0*	-2.8191	-12.0777	0.0*	-2.2589
mv	p	r	122	84	-13.1158	0.0*	-1.9962	-10.0519	0.0*	-1.5959

When we compare video types grouped by device, we can observe how we can reject all H_0 with large effect sizes. This means that the average size of clusters is different between video types with all the devices and that the size of the main cluster is significantly different in all the cases, so the relative number of viewers in the main area of interest is larger in route videos than in point videos.

Again, under Bonferroni's correction with 25 repetitions, a significance level $\alpha = 0.002$ should be considered. Some previously significant differences between devices in point and route videos may not be significant under this correction, but the differences between all the devices and a given device in both point and route videos are still significant (except on M in route videos between all the devices and *pc*) and the differences between video types are still significant in each of the device groups considered.

We have observed some differences in the results between the frame-based and the trajectory-based approaches. For example, we obtained significant differences in M in point videos in route videos which could not be obtained in the frame-based approach, while some similitudes in route videos were kept.

After this analysis, we can answer RQ4 (Are trajectories followed in the same way across devices and video types?): we can strongly affirm that trajectories are not followed in the same way between video types, arriving to the same conclusion as Rossi *et al.* in [109]. Between devices, while the differences between individual devices are significant in some cases and non-significant in other cases, the results when all the devices are considered together are worse than when they are considered individually: the average size of the clusters and the size of the main cluster are smaller when the devices are mixed than when they are processed individually.

5.7. Conclusion

An initial exploratory analysis reported that users of immersive devices tend to look around five degrees of latitude lower than users of non-immersive interfaces. We reported how users of non-immersive devices tend make complete turns in longitude more than those of immersive interfaces. We showed how deviation in longitude was higher in point videos than in route videos and that in point videos, deviation in longitude with non-immersive interfaces was higher than with immersive devices, while the opposite was true in route videos.

We have found that the temporal exploration of video sessions of the same type may not be equal with all the devices: users of the *pc* interface viewed point videos during less time than with the other devices and users of the *pc* interface viewed route videos during less time than with the *oc* interface; the difference between the rest of the video types was not significant. Between video types, route videos were significantly more explored than point videos with all the devices.

We understand that the familiarity of the users with the *pc* interface made the participants more eager to select hyperlinks (and it was easier) which made them skip parts of the 360-degree videos. On the other hand, users of the *oc* interface tended to consume 360-degree videos during more time, possibly because of the immersion that the device provides. The lower temporal exploration of point videos can be attributed to their function to navigate to other videos, so they could be skipped quickly by some users.

Spatial exploration in immersive devices tends to be lower than when non-immersive devices are used. In point videos, this difference between immersive and non-immersive devices is statistically significant except between *oc* and *mb*, while in route videos, it is always significant.

While changes of orientation short in distance may be performed very naturally with immersive devices, we understand that non-immersive interfaces may introduce an advantage in spatial exploration, as it may be easier to simply move the mouse or drag a touchscreen than to turn around one's body and head.

We found that temporal and spatial exploration are significantly moderately correlated. We observed that these values of correlation differ by device used to play the videos and that with immersive devices, the correlation is similar between video types, while it is dissimilar when non-immersive devices are used. The design of the interaction must play a role in this, as interaction with immersive devices is restricted by physical movement of the users' body, while in non-immersive devices, interaction is performed through peripherals like a touchscreen or a computer mouse, which are not as natural. VR fatigue [111] is also likely to play a role in the eagerness of exploration and should be studied.

We analyzed the dispersion of the orientation of the users, which resulted in significantly greater dispersion in point videos than in route videos per device. In point videos, orientation with non-immersive interfaces was significantly more disperse than with immersive devices, while the opposite was true in route videos (except between *mb* and *mv*).

After calculating the users' movement in segments of lengths between one and five seconds, we found it to be significantly larger in point videos than in route videos. We found, in point videos, significant difference in movement between most devices (except between *mb* and *mv*), while in route videos the differences were much slimmer, and significant in only a few cases, but we found how, in route videos, there is very little difference between devices of the same immersivity (between *pc* and *mb* and between *oc* and *mv*). This shows us how in some cases, there are differences in behavior between devices and how, in other cases, this behavior is similar. Particularly, movement in *pc* was found to be more erratic than with the other devices and is thought to be caused by the ease to make large changes of orientation with a simple movement, compared with the other devices.

A clustering method was applied on the reproductions of each video, considering each device by its own and all of them together. The results, the distribution over the duration of the video of the average size of the clusters relative to the number of viewers and the size of the main cluster, were obtained per different trajectory lengths. We observed significant differences between video types in these values with a given device and also between some combinations of devices in a given video type. We

observed that immersive interfaces (*oc* and *mv*) perform similarly in route videos, while they do not in point videos. We finally observed how the results got significantly worse when all the devices were considered together, so while behavior may be similar between devices, they definitely do not follow the same trajectories in the same way.

In the end, we have shown how user behavior is different in 360-degree videos according to the context of the session. We confirmed that the type of video plays a major role in that difference, with behavior more disperse in point videos than in route videos. The behavior per device differs, but in general users of the *pc* interface were able to make quicker, larger movements with the mouse, which were more limited to perform on the touchscreen of *mb*. Behavior in immersive devices was usually similar, due to the similarity of the interfaces and the physical limitations that they impose on the wearer of the headset.

We have obtained important insight from this study, as follows. As viewers are almost always centered in the middle latitudes, points of interest should be avoided in other places. Similarly, we have seen that users do not typically explore the sides of route videos, but look forward, so points of interest should be placed in that direction. We observed that users of immersive devices are more eager to spatially explore the content, so users of non-immersive interfaces may need some sort of encouragement to direct their attention to other parts of the video if that is wanted. We observed the average movement in segments of a given size, which gives us an idea of how much a user's orientation moves in a given window. This information, and the fact that varies by video type and device, is an important finding for the prediction of future orientations, as the magnitude of the change over time of that measure will differ according to what we found. Finally, as we analyzed the clustering of the users, we found that users of different devices do not always behave in the same way, which is a concern for saliency-based prediction techniques [43].

The differences we found will be used in the next chapter, in which we will classify the user behavior into the context variables (video type and device), and we will use them to improve a technique to predict future orientations.

Chapter 6. Improvement of viewport prediction in 360-degree videos

In this chapter, we study the effect that can have the knowledge of the context of a trajectory in the prediction of future viewports in a 360-degree video reproduction.

As we have already analyzed in Chapter 5, we noticed that users behave differently across two contextual variables: the video type (point or route) and the device used to play it (Oculus, Mobile, Mobile visor or PC).

Based on the data reported in section 5.1, we set two experiments. First, we will obtain what we already know: given a trajectory of orientations in a 360-degree video, can we determine the context of that trajectory? Second, can we improve a technique to predict future orientations in the trajectory leveraging the context of that trajectory?

As we previously introduced in section 2.7, Deep Learning techniques are used to predict future values in time series like the trajectory of the orientations of users in 360-degree videos. Deep Learning can also be used to classify data into labels. For these reasons, it seems that these techniques may be useful to pursue our objectives in this chapter.

We will first try to classify the trajectories into the context, with a goal of an accuracy of 90%. Then, to determine if we can improve a prediction technique if we leverage the context of the trajectory, we formulate these research questions:

RQ1. Can we predict future orientations with neural network models?
RQ2. Can we achieve better results with more data (higher values of L)?
RQ3. Does a neural network model improve its results if only samples with the same video type and device are used?
RQ4. Does a neural network model improve its results when the information about which video type and device were used are included?
RQ5. Do these results differ by video type and device used to play the videos?

In the following section, we will report the study on the classification problem and in the next section, we will expose the analysis on the prediction of future orientations. We end this chapter with a conclusion of both issues.

6.1. Classification of the reproductions

The objective of this study is to classify the trajectories of the orientations obtained in the reproductions of 360-degree videos into the contextual variables we have recorded (video type and device used in the reproduction of the spherical video).

We performed three experiments, with deep learning models, which are reported in the following subsections.

6.1.1. Densely connected model on the movement in a segment

We perform our first experiment with the same data used to analyze the angular distance of the sessions in 5.5: each session is divided in segments of $L \in \{4, 8, 12, 20\}$ frames (1, 2, 3, 5 seconds respectively) and the maximum angular distance described by the viewport center from the start of the segment is annotated (5.3). In this first experiment we consider $L = 4$. The 14210 segments are distributed as described in Table 6.1.

Table 6.1. Number of samples per category in the classification by movement in a segment.

	pc	***oc***	***mb***	***mv***	**total**
point	687	1345	821	911	3764
route	2524	3884	1927	2111	10446
total	3211	5229	2748	3022	14210

Since there is an imbalance between groups, we draw an equal number of samples of all groups, limited by the size of the smallest one. So, to classify the video type, 3764 samples of both types were used, while to classify the device used to play the videos, 2748 samples of each device were considered. A 80-20 split between training and testing was used, following [76]: 20% of these samples are randomly reserved to evaluate the model, another 16% (20% of the remaining 80%) are destined for the validation process and the remaining 64% are used to train the model.

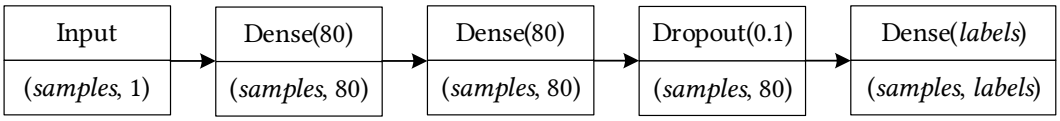

Figure 6.1. Summary of the neural network model used in the classification by movement in a segment.

Each of these segments constitutes a sample. The single feature, the angular movement in the segment, is a real value in the range [0,180]. Each sample is labeled with the kind of video which was played (two values) or the device used to play it (four values). These categorical labels are one-hot encoded to be able to be fed into the network. This way, the shape of the features is (*samples*,) and the shape of the labels is (*samples*, *labels*).

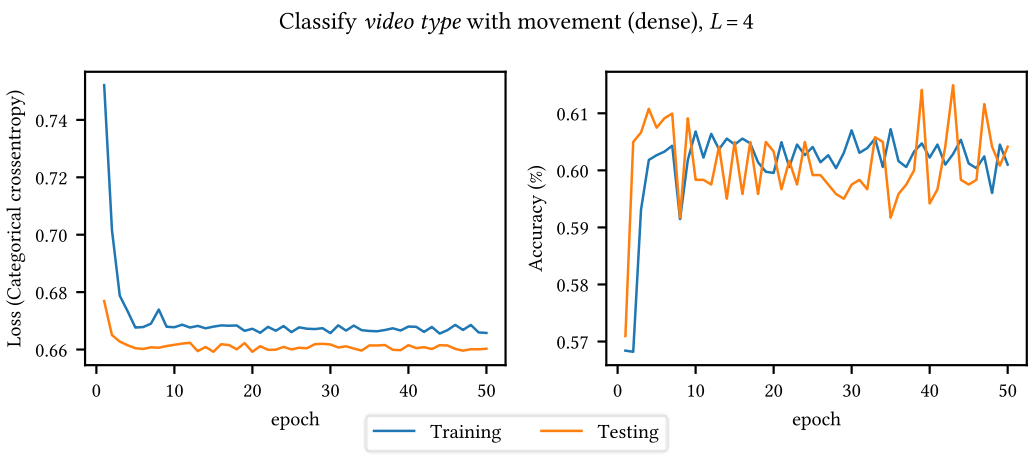

Figure 6.2. Evolution of the training and testing processes of the densely connected model to classify the video type from the movement in segments of $L = 4$.

A sequential model is built in Keras[24] as follows (see Figure 6.1): the samples are fed in batches of 32 samples into two successive densely-connected layers of 80 nodes activated by the *relu* funcion, followed by a dropout layer at 10%, with a final dense layer of *labels* nodes, as much as labels are considered in the experiment (two to classify the video type, four to classify the device used to play the video).

The *adam* optimizer is chosen and categorical crossentropy is used as the loss function. We measure the accuracy of the model as our metric, as the groups are balanced.

After 50 epochs, this model achieved an accuracy of 65.76% to classify the video type and 31.61% to classify the device used to play the video. With longer segment lengths ($L \in \{8, 12, 20\}$), the results were almost identical. We can observe in Figure 6.2 and Figure 6.3 that this model does not learn much from the data when it classifies the video type nor the device used to play the videos.

[24] https://keras.io/

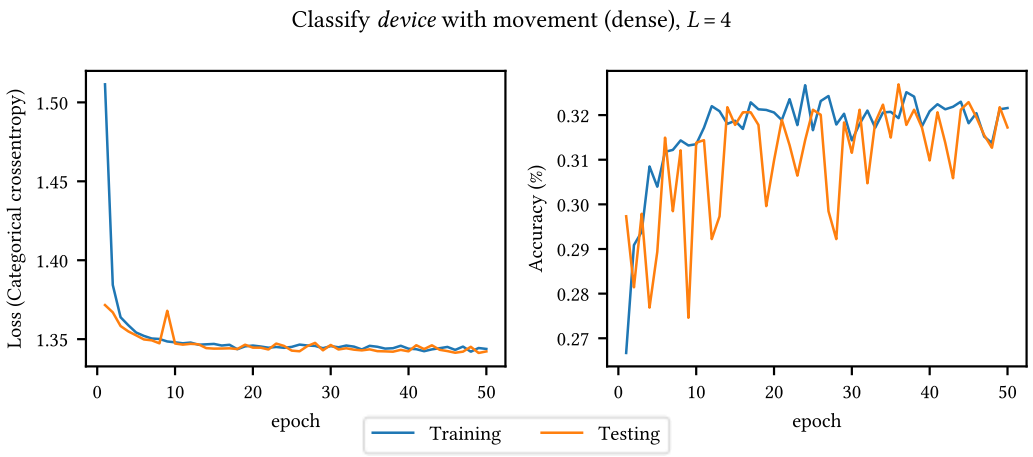

Figure 6.3. Evolution of the training and testing processes of the densely connected model to classify the device from the movement in segments of $L = 4$.

These initial results are not very promising, which is reasonable, as the classification process is based on a single data point.

6.1.2. Recurrent model on the movement in a session

Since our dataset contains trajectories, one way to leverage this kind of data is to use a Recurrent Neural Network (RNN), which are used to process time series. We used the Long Short-Term Memory (LSTM) layer, a kind of RNN layer, to feed trajectories into a neural network, as used in the literature in, e.g. [71], [76].

We chose to use the same dataset as in the previous experiment, but this time instead of considering each of the segments independently, we grouped them according to their session, so each sample is now an entire session, with the movements in each of its segments as features, with shape (*segments*,).

We must provide a 3D tensor to an LSTM layer, so each of the elements in the array of movement in segments of each session was converted to a one-element array. This way, a single sample is converted to shape (*segments*, 1).

Table 6.2. Number of samples per category in the classification by movement in a session.

	pc	***oc***	***mb***	***mv***	**total**
point	91	131	80	90	392
route	96	127	76	84	383
total	187	258	156	174	775

Since the shape must be regular, the samples were padded to the length of the longest session, which we call *timesteps*. The value 360 was chosen as the mask as it is not possible to perform as much movement (it is limited to 180 degrees, as it is the greatest angular distance between two points). This makes each sample to have the shape (*timesteps*, 1).

We followed the same procedure to draw the samples from the 775 sessions (see Table 6.2) which are available and distributed them in evaluation, validation and testing sets in the same way as in the previous experiment.

In the end, the shape of the features is (*samples*, *timesteps*, 1) and the shape of the labels is (*samples*, *labels*), with two or four labels depending on what we want to classify.

A sequential model was built in Keras as follows (refer Figure 6.4): the samples are fed into a Masking layer that marks the timesteps with value 360 to be skipped, followed by an LSTM layer of 128 nodes, followed by a dropout layer at 20%, then a dense layer of *labels* nodes, as much as labels are considered in the experiment, with a final activation layer with the *softmax* function.

Again, the *adam* optimizer was chosen and categorical crossentropy is used as the loss function. We measured the accuracy of the model as our metric.

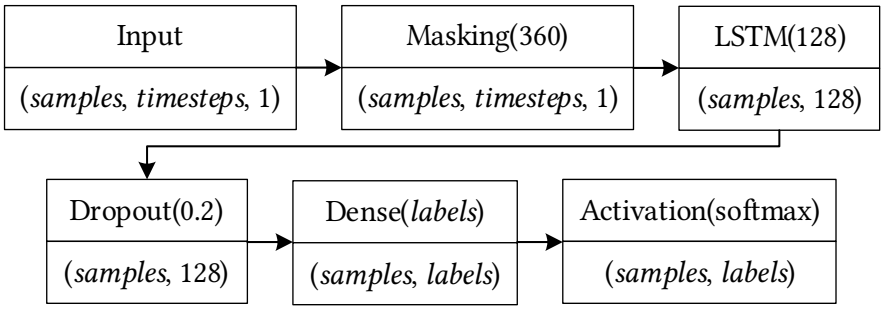

Figure 6.4. Summary of the neural network model used in the classification by movement in a session.

After 50 epochs, this model achieves an accuracy of 81.17% to classify the video type and 34.60% to classify the device used to play the video. With longer segment lengths ($L \in \{8, 12, 20\}$), the results were almost identical. We can observe in Figure 6.5 that this model learns when classifying the video type until epoch 10, after which it starts to overfit, while when it classifies the device used to play the videos (Figure 6.6), it does not perform as well and it overfits heavily.

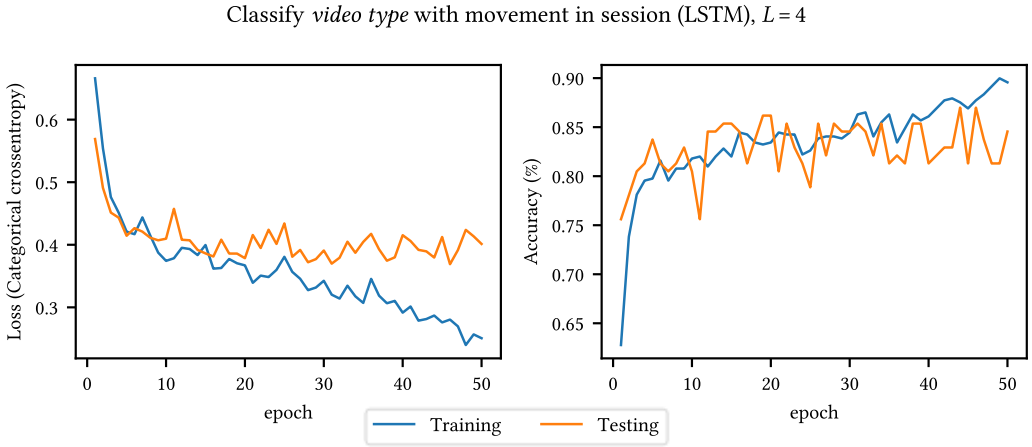

Figure 6.5. Evolution of the training and testing processes of the LSTM model to classify the video type from the movement in a session in segments of $L = 4$.

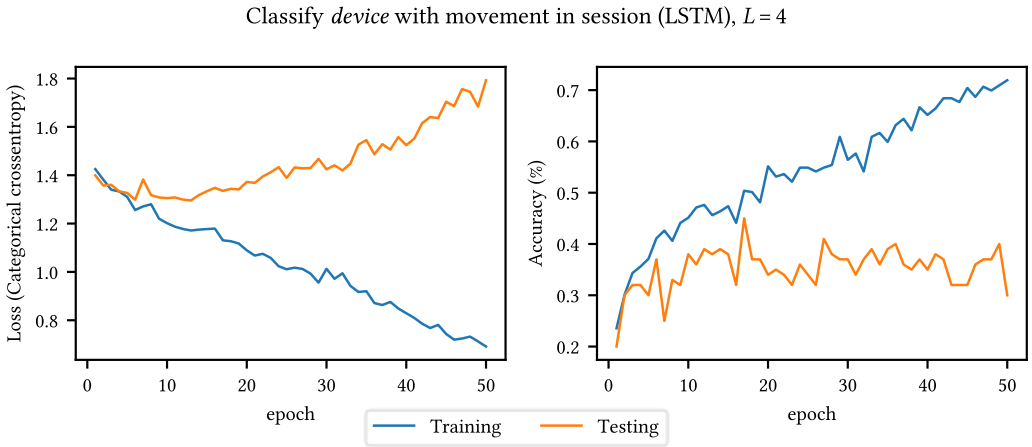

Figure 6.6. Evolution of the training and testing processes of the LSTM model to classify the video type from the movement in a session in segments of $L = 4$.

While these results are slightly better than in the previous experiment, there is still plenty of room for improvement. We have seen that with a Recurrent Neural Network model can be considered instead of individual samples with single values.

6.1.3. Recurrent model on the values of orientation in a window

In this last experiment, we change the representation of the data: instead of using a processed version of the trajectory of the orientations in the reproduction of 360-degree videos, we will use these values of orientation directly. Also, instead of using the values of an entire session as a sample, we will create sliding windows of L frames with an overlap of 1 frame from each session. This results in a high number of samples with the same length, compatible with LSTM networks.

Table 6.3. Number of samples per category in the classification with a sliding window of $L = 4$, for the spherical and cartesian representations.

	pc	***oc***	***mb***	***mv***	**total**
point	2456	4958	3019	3354	13787
route	9749	15054	7435	8133	40371
total	12205	20012	10454	11487	54158

Table 6.4. Number of samples per category in the classification with a sliding window of $L = 4$, for the distance representation.

	pc	***oc***	***mb***	***mv***	**total**
point	2346	4810	2935	3256	13347
route	9649	14924	7355	8046	39974
total	11995	19734	10290	11302	53321

A total of three representations of these orientations were considered, and are compared in this section: first, the raw spherical coordinates, in which we did not normalize the longitude values; second, the cartesian projection of the spherical coordinates in a unit sphere; third, the spherical distance (3.1) between the orientation of a frame and the previous one.

This way, a session of length N frames contributes $N - L + 1$ samples of L timesteps to our dataset (one less in the case of the distance between frames, as it cannot be calculated for the first frame).

Since we based our model in the LSTM layer, we need to provide the network a 3D tensor as its input, with shape (*samples*, *timesteps*, *features*).The first dimension, *samples*, is the total amount of samples that are generated with the sliding window technique; the second dimension, *timesteps*, is the size of that window (L); the third dimension, *features*, depends on the number of features that are fed into the network: two for the spherical coordinates, three in the cartesian representation, and one when the spherical distance is considered. See Figure 6.7 for an example of an input tensor with shape (5,4,3), which could be used for the cartesian representation. Refer Table 6.3 and Table 6.4 for the distribution of samples across categories. A uniform number of samples was sampled from each category, limited by the size of the smallest one, which were divided into training, validation and testing sets as in previous experiments.

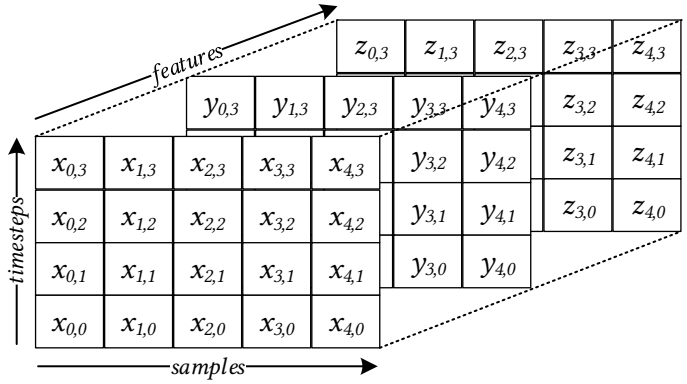

Figure 6.7. Representation of the 3D tensor of features in the LSTM model for three features (such as in the case of the cartesian representation).

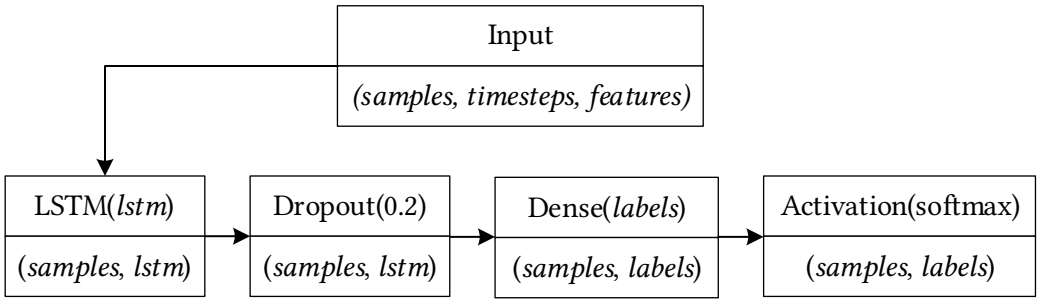

Figure 6.8. Summary of the neural network model to classify a window of a trajectory with a single LSTM layer.

Two sequential models were built, one with one LSTM layer and the other with two (refer Figure 6.8 and Figure 6.9, respectively): the samples are fed, when two LSTM layers are used, into a first LSTM layer of *lstm* nodes which returns the entire sequence and is followed by a dropout layer at 20%. Then, for both models, either the output of that last layer (when two LSTM layers are used) or directly the input (when a single LSTM layer is used) into an LSTM layer of *lstm* nodes, which is also followed by a dropout layer at 20%. Its results are collected by a dense layer of *labels* nodes, as much as labels are considered in the experiment, with a final activation layer with the *softmax* function.

Again, the *adam* optimizer was chosen and categorical crossentropy is used as the loss function. We measured the accuracy of the model as our metric. We performed training processes of 150 epochs with batches of 32 samples, as before.

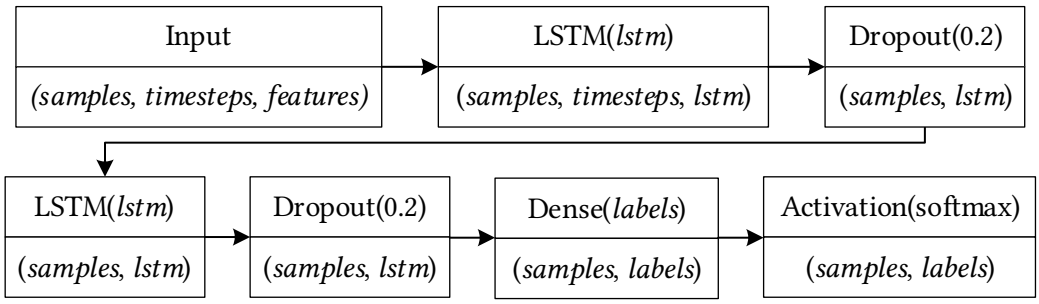

Figure 6.9. Summary of the neural network model to classify a window of a trajectory with a pair of LSTM layers.

We trained both models, the first configured with *lstm* = 128 (a single LSTM layer of 128 nodes) and the second with *lstm* = 256 (two LSTM layers with 256 nodes each), with the three representations of the data and with all the values of L. These experiments were repeated five times and the average results are reported in Table 6.5 (classification of the video type) and Table 6.6 (classification of the device used), while boxplots are reported in Figure 6.10 (classification of the video type) and Figure 6.12 (classification of the device used).

Training times varied by the experiment which was run, the size of L and the configuration of the network. On a laptop with an Intel i7-8750H CPU and an NVIDIA GeForce GTX 1060, LSTM 128 $L = 4$ classifies the video type in 4 minutes, while the device needs 8 minutes; an increase in L increases the training time, as LSTM 128 $L = 20$ needs almost 20 minutes to classify either the video type or the device used to play the video. LSTM 256-256 $L = 4$ needed 33 minutes to classify the video type and 18 minutes to classify the device used to play it.

We can observe how the second configuration of the network usually performs better than the first, thanks to the addition of a second LSTM layer and also to the increased number of nodes. This difference is small in the classification of the video type, while it is larger when we classify the device, where the results of LSTM 256-256 $L = 8$ are comparable to LSTM 128 $L = 12$ and the results of LSTM 256-256 $L = 12$ are comparable to LSTM 128 $L = 20$: similar results with one and two seconds less available of information, respectively. A test, which is not reported here, was performed with a single layer of 256 nodes and two layers of 128 nodes each, presented results which were in between the two cases reported in this chapter.

We can also notice how the distance-based feature compares poorly with the other two: its results are always considerably worse than with the spherical or the cartesian representation of the data.

Table 6.5. Average accuracy of the neural network used to classify the video type of a window of a trajectory of size L by configuration of the model, L, and representation of the features.

	LSTM 128			LSTM 256-256		
L	**Spherical**	**Cartesian**	**Distance**	**Spherical**	**Cartesian**	**Distance**
4	72.13%	73.54%	60.82%	73.41%	73.59%	60.70%
8	78.91%	77.92%	60.15%	81.52%	80.17%	60.55%
12	84.63%	84.19%	66.92%	88.41%	86.84%	67.52%
16	88.29%	89.17%	72.18%	91.96%	91.67%	77.51%
20	90.23%	92.08%	78.37%	93.38%	93.21%	82.50%

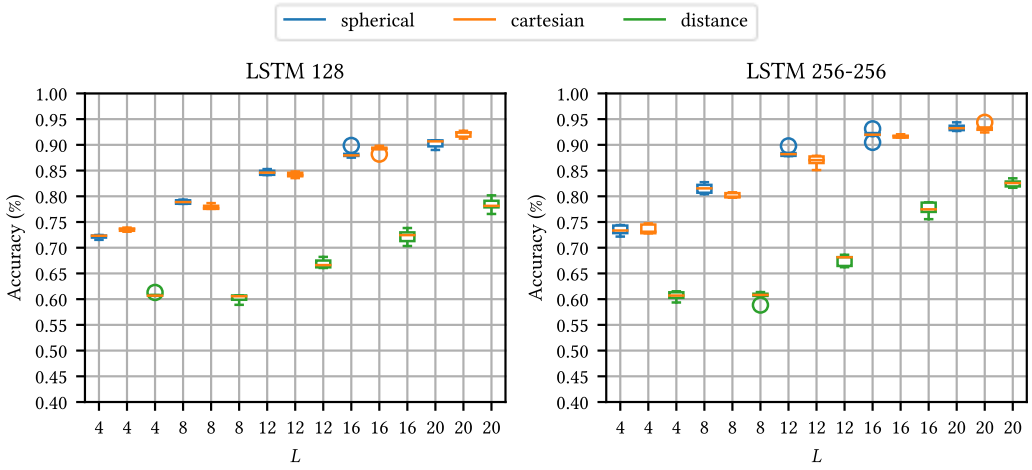

Figure 6.10. Boxplot of the accuracy of the neural network used to classify the video type of a window of a trajectory of size L by configuration of the model, L, and representation of the features.

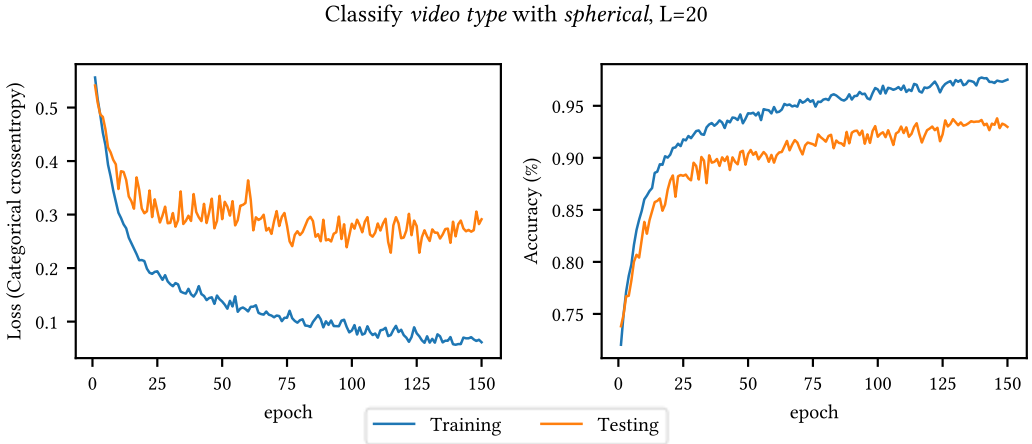

Figure 6.11. Evolution of the training and testing processes of the LSTM model (256-256) to classify the video type from the spherical coordinates in windows of $L = 20$.

With LSTM 128, $L = 12$, an accuracy of 84.63% and 84.19% is obtained to classify the video type with spherical and cartesian coordinates, respectively, which surpass the results obtained in the previous experiment (81.17%), so with three seconds of trajectory, we can improve our previous results which considered the entire session. At $L = 16$, the LSTM 256-256 model achieves an accuracy of 91.96% and 91.67% with spherical and cartesian coordinates, respectively, which can be further improved to 93.38% and 93.21% at $L = 20$.

We can observe in Figure 6.11 how the model learns to classify the data properly, although half the epochs would have sufficed to obtain similar results.

The results with the spherical representation were marginally better than with the cartesian representation to classify the video type. To classify the device, we can observe this same phenomenon in $L \in \{4, 8, 12, 16\}$ in LSTM 128 and in $L \in \{4, 8, 12\}$ in LSTM 256-256, but ultimately, the cartesian representation takes ahead in the latter results. In any case, the difference is small.

Table 6.6. Average accuracy of the neural network used to classify the device used in a window of a trajectory of size L by configuration of the model, L, and representation of the features.

	LSTM 128			LSTM 256-256		
L	**Spherical**	**Cartesian**	**Distance**	**Spherical**	**Cartesian**	**Distance**
4	54.81%	47.36%	41.94%	58.52%	53.96%	40.74%
8	64.75%	59.18%	48.10%	74.96%	70.42%	51.65%
12	74.82%	71.77%	62.00%	85.15%	84.55%	71.39%
16	81.55%	80.97%	73.37%	90.41%	91.76%	82.35%
20	84.45%	86.82%	80.68%	92.73%	94.86%	87.00%

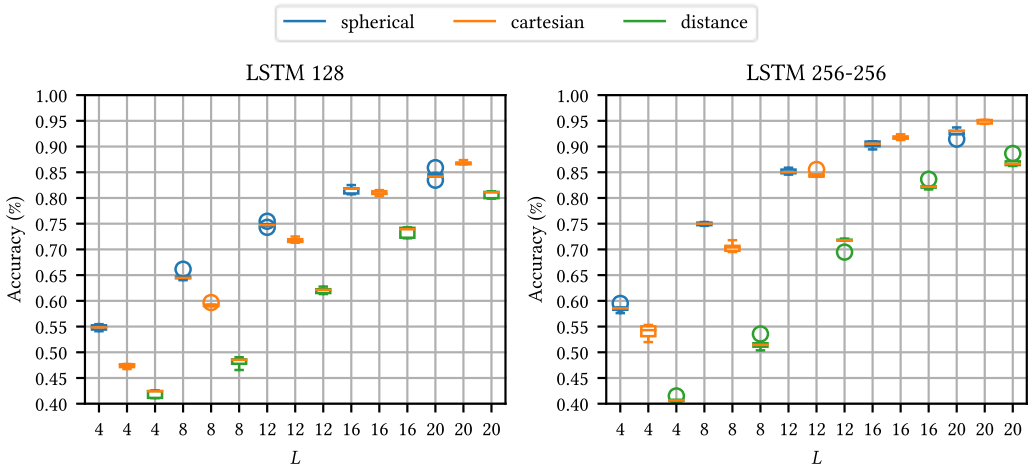

Figure 6.12. Boxplot of the accuracy of the neural network used to classify the device used in a window of a trajectory of size L by configuration of the model, L, and representation of the features.

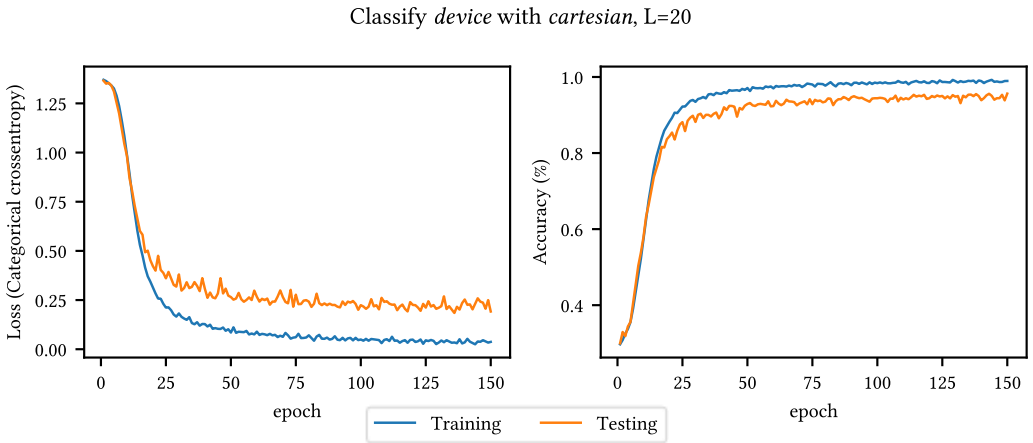

Figure 6.13. Evolution of the training and testing processes of the LSTM model (256-256) to classify the device from the cartesian coordinates in windows of $L = 20$.

With our previous experiment, we achieved an accuracy of 34.60% to classify the device used in a session, which is surpassed in all the reported cases. The LSTM 256-256 model achieves 90.41% and 91.76% accuracy at $L = 16$ with spherical and cartesian representations, respectively and even reaches 94.86% at $L = 20$ with the cartesian representation of the data.

The learning process of the classification of the device appears to be faster (see Figure 6.13): after 50 epochs, learning appears to be stalled.

With these results, we can affirm that we can correctly classify with more than a 90% of accuracy the video type and the device used with just 4 seconds of trajectory, getting very close to 95% to classify the device when 5 seconds of trajectory are used.

6.2. Prediction of future orientations

The purpose of this section is to predict future orientations in the trajectories described by the reproduction of 360-degree videos and try to improve that prediction by leveraging the contextual variables we collected during the experiment (video type and device).

For this reason, we started by building a simple network in which we fed our dataset, followed by partitioning our data by its context and then building a neural network that treats that context as features.

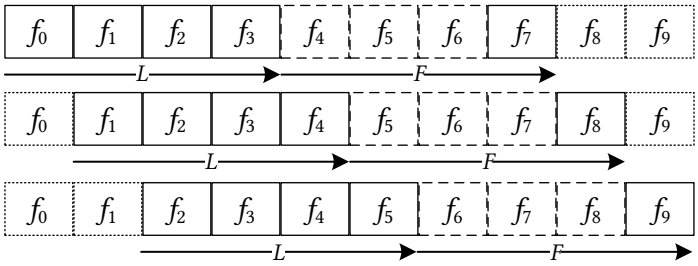

Figure 6.14. Example of the sliding window technique used to build the dataset to predict future orientations for $L = 4$ and $F = 4$.

Neural network models are capable of performing regression, which is to predict values from a given input. Since our dataset contains the user's orientation for the entire playback of each video, we can partition that data in windows of a given size ($L \in \{4, 8, 12\}$ frames) in the past and try to predict the user's orientation in a given distance ($F \in \{4, 6, 8\}$ frames) in the future (see Figure 6.14). These values of F were chosen according to the ones used in literature for adaptive streaming (between 1 and 2 seconds) [63], [64], [69], [77].

Table 6.7. Distribution of the samples per type of video, device used to play it, length of the feature window (L), and distance of the prediction (F).

		Point videos			**Route videos**		
L	**Device**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	*pc*	2049	1871	1708	9354	9163	8976
4	*oc*	4396	4134	3881	14540	14287	14035
4	*mb*	2691	2531	2379	7124	6973	6825
4	*mv*	2971	2791	2616	7792	7624	7458
4	total	12107	11327	10584	38810	38047	37294
8	*pc*	1708	1560	1425	8976	8790	8605
8	*oc*	3881	3635	3407	14035	13783	13533
8	*mb*	2379	2227	2078	6825	6679	6533
8	*mv*	2616	2448	2282	7458	7292	7126
8	total	10584	9870	9192	37294	36544	35797
12	*pc*	1425	1299	1181	8605	8421	8240
12	*oc*	3407	3185	2973	13533	13289	13047
12	*mb*	2078	1937	1802	6533	6387	6244
12	*mv*	2282	2118	1957	7126	6960	6795
12	total	9192	8539	7913	35797	35057	34326

Using a sliding window technique, we build the samples of the new dataset as follows: a total of $L + F$ frames are needed per sample, so the first L are used to build the window of past orientations and the next F frames are used to obtain the future orientation (if we follow Figure 6.14, the features of the first sample are $[f_0, f_1, f_2, f_3]$ and the label is f_7; for the second sample, the features are $[f_1, f_2, f_3, f_4]$ and the label is f_8, etc). In this case, only the last frame in F is kept. A total of $D - L - F + 1$ samples

are generated per session according to its duration D (following Figure 6.14, a hypothetical session of $D = 10$ would result in $n = 10 - 4 - 4 + 1 = 3$ samples for $L = 4, F = 4$). Increasing L or F results in a slight decrease of the number of samples (refer Table 6.7).

In the end, for each combination of L and F, our dataset contains the list of past orientations (of size L) and the orientation at a future instant. This process is performed for both the spherical coordinate representation of the data and its cartesian coordinate representation.

To use the dataset in a neural network model, it has to be formatted as input and expected output. Following Figure 6.15, the input is a rank-3 tensor, with the number of samples as its first dimension, the amount of timesteps as its second dimension (L) and the features as its third dimension (2 in the spherical representation, 3 in the cartesian representation). The output is a rank-2 tensor, with the number of samples as its first dimension and the features as its second dimension (in the same way as the input).

LSTM models are not only useful to classify time sequences, but they are also used to perform regression on them. For this reason, we built a relatively simple model with two LSTM layers followed by a Dense layer with as many neurons as labels have to be predicted. Following Figure 6.16, the first layer is an LSTM layer with $lstm_0$ nodes which returns the entire sequence, so the second dimension in the input is preserved; this is processed by a second LSTM layer with $lstm_1$ nodes, which flattens the data along the second axis and finally *labels* values per sample are received in the end.

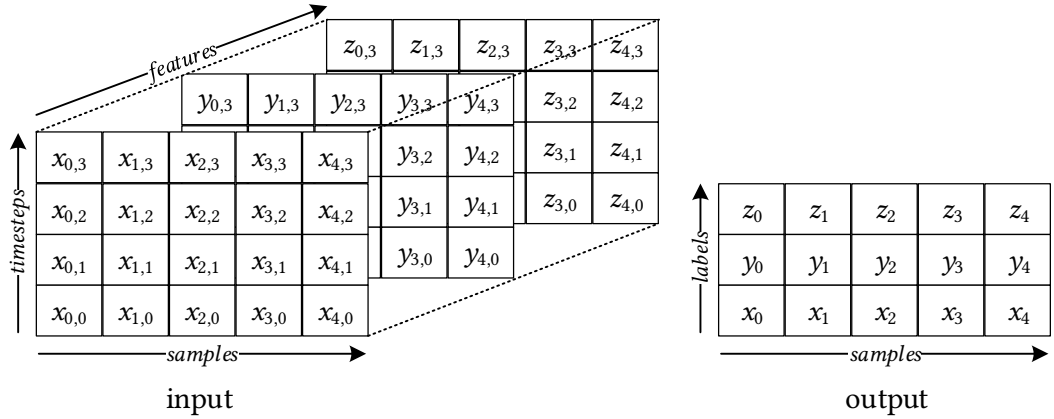

Figure 6.15. Structure of the input and output of the neural network model to predict future orientations for 5 samples, 4 timesteps and three features and labels (x, y, and z).

The *adam* optimizer was chosen, with Mean Squared Error as the loss function. As other works are based on an actual viewport-dependent streaming system, the accuracy of choosing the correct tile or the bandwidth saved can be used as metrics. In our case, as we lack that, a custom metric based on the geodesic distance was specially built to measure and compare our results, so the mean distance is returned between the output of the model and the observed values. When the cartesian coordinates are used to represent the data, both predictions and labels need to be converted to spherical coordinates before they can be introduced to the metric function.

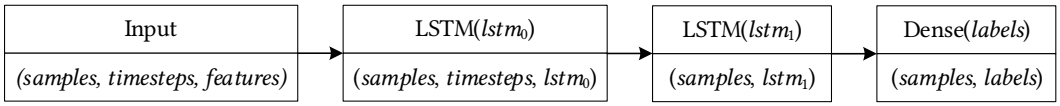

Figure 6.16. Summary of the neural network model used to predict a future orientation.

The same procedure to split the samples between training, evaluation and validation is followed in these experiments: 20% of the samples are reserved for validation, 20% of the remaining 80% (16%) are used in the evaluation process and the remaining 64% is used to train the neural network.

A total of five models have been developed to predict a future viewport. The first two, *persistence* and *linear* act as baselines, the future orientation is obtained without the need to use any kind of neural network. The other three, *blind*, *grouped*, and *informed* are variations of the model explained in this

section. All the models have been first applied to all the data and then by grouping the dataset by video type, for the difference between video types already discussed in Chapter 5.

For each model, we obtained its results as the great-circle distance between the predicted viewport and the true value. The average of these distances is reported for each configuration of $L \in \{4, 8, 12\}$ and $F \in \{4, 6, 8\}$. We also repeated the experiments using the entire dataset and by partitioning it by video type. The weighted average by number of samples (following Table 6.7) of the results by video type will be compared to the results of the entire dataset.

The following five subsections detail the process for the five proposed models and their results. After this, we will discuss the results and answer the research questions we formulated in the beginning of this chapter.

6.2.1. Persistence model

The simplest prediction that can be made with any model lies in a persistence model. This technique assumes that future values are the same as the last known value, so it *persists*. Following Figure 6.17, it is assumed that the value of f_3 is kept indefinitely, so it is kept as the prediction.

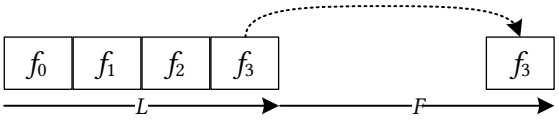

Figure 6.17. The persistence model.

The results for this model can be found in Table 6.8. We can observe that the average great-circle distance grows with F, because the farthest the prediction is made from the last known point, the more it diverges. The distance gets slightly smaller when L grows, because less samples are available.

Table 6.8. Average great-circle distance (in degrees) for the persistence model to predict a future orientation.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	19.58	25.61	30.05	28.50	37.27	43.77	16.80	22.14	26.15
8	19.20	25.17	29.57	27.98	36.64	43.03	16.71	22.07	26.11
12	18.87	24.71	29.02	27.45	35.65	41.69	16.66	22.05	26.10

We can also observe the difference between applying this model to the entire dataset and to the dataset partitioned by video type. Here we can see that average distances are bigger in point videos than in route videos, as expected by the results in Chapter 5. We have to keep in mind that our dataset contains more samples of the reproduction of route videos, and that is why the results for the entire dataset are closer to those than to the results of the point videos. Furthermore, we can observe that the weighted average between the results of point and route videos is equal to that of the entire dataset, as expected.

6.2.2. Linear model

With the aim to build a more intelligent baseline, we thought of predicting future viewports according to the great-circle route between the last two known points.

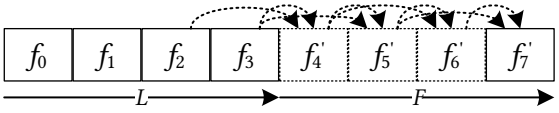

Figure 6.18. The linear model.

To do this, see Figure 6.18, we iteratively obtain the next viewport from the previous two, starting by the last two known viewports (f_2 and f_3 in Figure 6.18), until we predict a total of F future orientations. Then, only the last value is kept (f_7' in Figure 6.18).

The next point is obtained via a geometric approach (see Figure 6.19): let $A = (x_1, y_1, z_1)$ and $B = (x_2, y_2, z_2)$ be two points on the sphere in cartesian coordinates, the next point in the route between A and B on the great circle is $C = (x_3, y_3, z_3)$. To find C, we first note the vector $\overrightarrow{OB}$ between the origin $O = (0,0,0)$. We find M, the orthogonal projection of A over the line defined by $\overrightarrow{OB}$. Since M is a multiple of B, we can express it as $M = (\lambda x_2, \lambda y_2, \lambda z_2)$ and obtain λ through (6.1). Once the point M is known, we can obtain C, as M is the midpoint between A and C (6.2). Finally, we normalize C by dividing it by its norm, to force it to lie on the sphere (6.3).

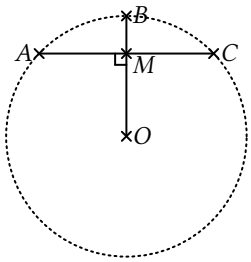

Figure 6.19. 2D representation of the approach to obtain the next point in a geodesic.

The results of the linear model are shown in Table 6.9. It can be observed that the results are similar to those of the persistence model: the average great-circle distance increases with F and it shrinks a little with larger values of L for the same reason as the previous model.

$$\begin{aligned} 0 &= \overrightarrow{MA} \cdot \overrightarrow{OB} \\ 0 &= (x_1 - \lambda x_2)x_2 + (y_1 - \lambda y_2)y_2 + (z_1 - \lambda z_2)z_2 \\ 0 &= x_2 x_1 - \lambda x_2^2 + y_2 y_1 - \lambda y_2^2 + z_2 z_1 - \lambda z_2^2 \\ \lambda(x_2^2 + y_2^2 + z_2^2) &= x_1 x_2 + y_1 y_2 + z_1 z_2 \\ \lambda &= \frac{x_1 x_2 + y_1 y_2 + z_1 z_2}{x_2^2 + y_2^2 + z_2^2} \end{aligned} \tag{6.1}$$

$$\begin{aligned} M &= \frac{1}{2}A + \frac{1}{2}C \\ C &= 2M - A \\ (x_3, y_3, z_3) &= (2\lambda x_2 - x_1, 2\lambda y_2 - y_1, 2\lambda z_2 - z_1) \end{aligned} \tag{6.2}$$

$$\frac{C}{\|C\|} = \frac{C}{\sqrt{x_3^2 + y_3^2 + z_3^2}} \tag{6.3}$$

Table 6.9. Average great-circle distance (in degrees) for the linear model to predict a future orientation.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	21.26	30.04	36.65	30.10	41.50	49.65	18.50	26.63	32.97
8	20.78	29.45	36.03	29.20	40.50	48.51	18.39	26.47	32.82
12	20.36	28.91	35.49	28.45	39.42	47.48	18.28	26.35	32.72

Conversely, the persistence model behaves better in all the cases: for $F = 4$, the linear model shows a distance ~1.5 degrees greater than the persistence model, which grows to ~4.5 degrees for $F = 6$, and to ~6.5 degrees for $F = 8$. The results are similar when the dataset is partitioned by video type: the weighted average between video types is equal to the results of the entire dataset.

6.2.3. Blind model

This is the first neural network model that we built. In this case, we simply divide the dataset in three sets for training, testing and validation and feed them into the neural network model that we specified in the beginning of the section. The size of the validation set is 20% of the entire dataset, the size of the testing set is 20% of the remaining 80% (16%), and the size of the training set is the remaining 64%. We used a batch size of 32 samples and trained the network for a total of 50 epochs. The configuration of the network was the following: 128 nodes were set up for each of the LSTM layers and each of them featured a dropout of 20%. They were both activated by the *relu* function.

We performed an initial experiment with both spherical and cartesian representations of the orientations (see Figure 6.20 and Figure 6.21, respectively) and we noticed that the spherical coordinate representation does not make the model learn correctly (notice in Figure 6.20 how the evolution of the great-circle distance for the testing process does not descend). Conversely, when cartesian coordinates are used, a much more normal behavior is achieved (see Figure 6.21). For this reason, for the rest of this prediction experiment, only the cartesian representation was considered. Regarding Figure 6.21, 50 epochs were more than enough, as the evolution of the testing process does not improve much from epoch 20 onwards. This point is affected by both L and F: the higher those values, the more difficult it is for the model to fit.

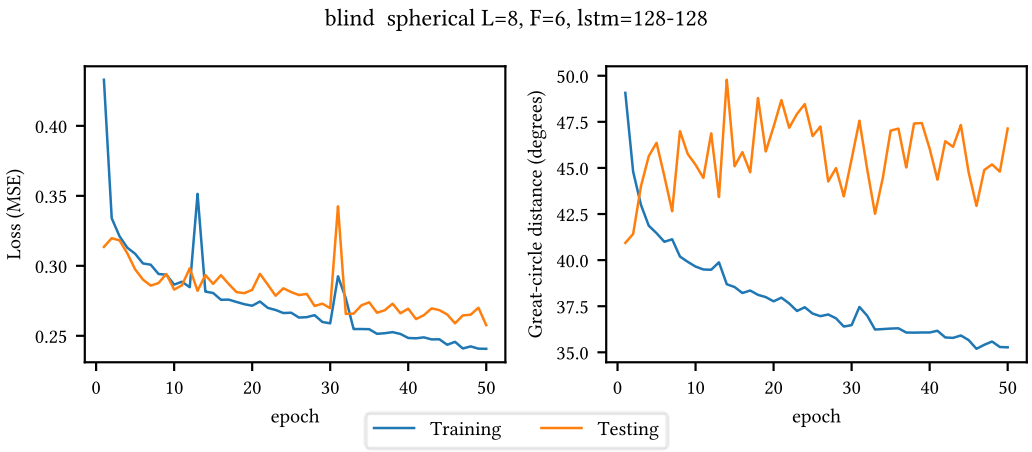

Figure 6.20. Evolution of the training and testing processes of the blind model for $L = 8, F = 6$ and spherical coordinates.

On the same hardware described in 6.1.3, training times varied by the size of L: around 5 minutes for $L = 4$, 8 minutes for $L = 8$ and 12 minutes for $L = 12$. The size of F did not affect the training time.

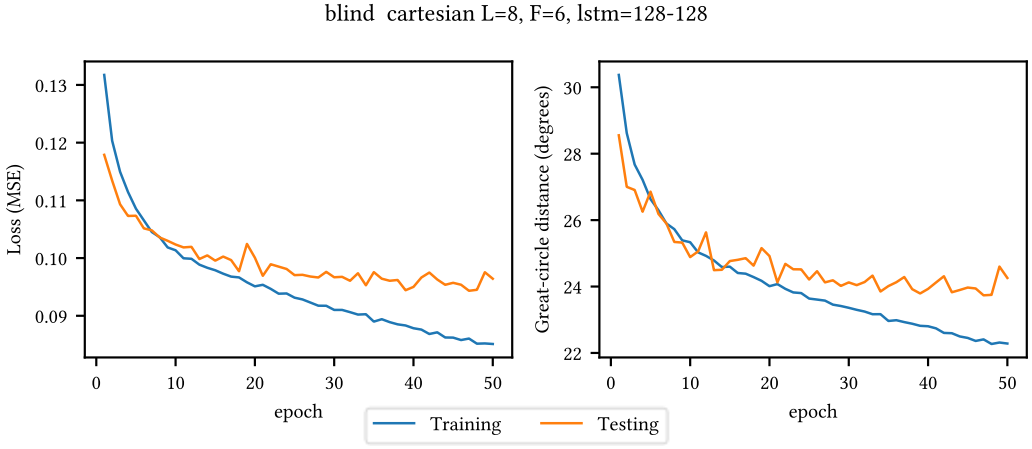

Figure 6.21. Evolution of the training and testing processes of the blind model for $L = 8, F = 6$ and cartesian coordinates.

In the end, we repeated every experiment ten times and its average results are shown in Table 6.10. As expected, the farther the prediction is made (larger values of F), the average great-circle distance grows. Moreover, greater values of L contribute to reduce the distance metric. This reduction can be no longer attributed only to the reduction in the number of samples, as its magnitude is higher than with the baselines, so the model must take advantage of a larger window of past values.

When we compare the results of the entire dataset and the weighted average of the results of splitting it by video type (see Table 6.11), we can still observe the clear divide between video types. This partition is useful, as slightly better results are obtained when video types are considered separately, especially for larger values of L and F: in the most extreme case ($L = 12, F = 8$), the weighted average

between both partitions is 24.15 degrees, slightly less than the 24.92 degrees when the entire dataset is considered.

Table 6.10. Average great-circle distance (in degrees) for the blind model to predict a future orientation.

	Entire dataset			Only point videos			Only route videos		
L	$F=4$	$F=6$	$F=8$	$F=4$	$F=6$	$F=8$	$F=4$	$F=6$	$F=8$
4	19.34	25.12	29.15	26.84	35.48	42.01	16.89	21.73	24.88
8	18.40	23.98	27.45	25.84	34.07	38.73	16.15	20.54	23.79
12	17.68	21.81	24.92	24.17	30.16	33.45	15.40	19.40	22.00

Table 6.11. Comparison of the results for the blind model to predict a future orientation using the entire dataset, the weighted average of the partitioned dataset and the improvement of using such partitioning.

	Entire dataset			Partitioned dataset			Improvement		
L	$F=4$	$F=6$	$F=8$	$F=4$	$F=6$	$F=8$	$F=4$	$F=6$	$F=8$
4	19.34	25.12	29.15	19.26	24.88	28.67	0.9957	0.9906	0.9834
8	18.40	23.98	27.45	18.29	23.42	26.84	0.9941	0.9765	0.9779
12	17.68	21.81	24.92	17.19	21.51	24.15	0.9724	0.9861	0.9689

If we compare these results to the persistence model (see Table 6.12), we can observe that it generally performs better than it, and we will compare them in detail later. The farther the prediction is made (larger values of F), the better this model performs compared to the persistence model. Larger windows of past data (larger values of L) also make this model perform better than the persistence model, especially when both F and L grow, with a reduction of up to 4.1 degrees in $L = 12, F = 8$ for the entire dataset (an improvement of 14% compared to the baseline).

Table 6.12. Difference in the results (in degrees) between the blind model and the persistence model.

	Entire dataset			Only point videos			Only route videos		
L	$F=4$	$F=6$	$F=8$	$F=4$	$F=6$	$F=8$	$F=4$	$F=6$	$F=8$
4	-0.24	-0.49	-0.90	-1.66	-1.79	-1.76	0.09	-0.41	-1.27
8	-0.80	-1.19	-2.12	-2.14	-2.57	-4.30	-0.56	-1.53	-2.32
12	-1.19	-2.90	-4.10	-3.28	-5.49	-8.24	-1.26	-2.65	-4.10

When we partition the dataset by video type, the improvement of the results in route videos are similar but the results in point videos are greatly improved, up to 8.24 degrees (19.76%) from the persistence model.

6.2.4. Grouped model

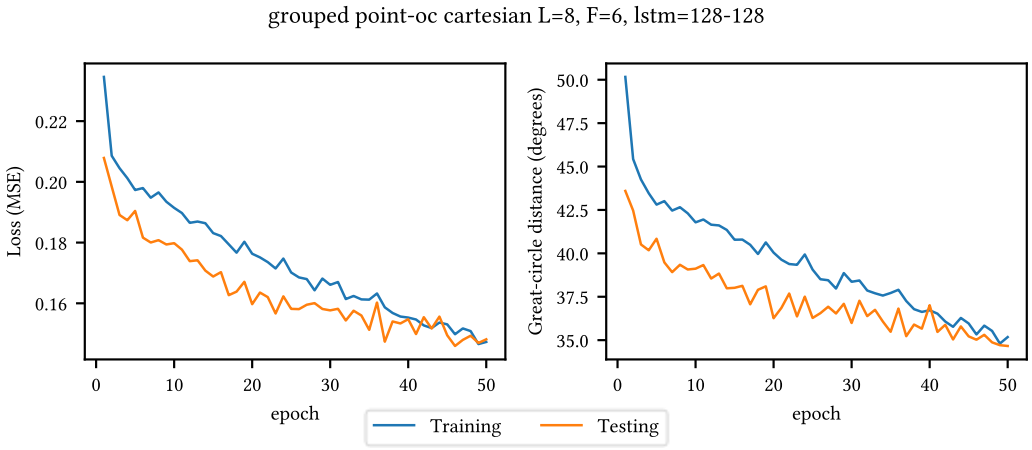

Figure 6.22. Evolution of the training and testing processes of the grouped model for $L = 8, F = 6$, cartesian coordinates, point video and Oculus interface.

This model follows the exact same principle than the blind model, but instead of running a single experiment for the entire dataset (or a partition by video type of the dataset), it is partitioned by both

video type and device used to play that sample. For each combination of L and F, there are eight different ways to partition the dataset, and each experiment is repeated ten times. In Figure 6.22 we can see the plot of the evolution of the loss and metric over the training and testing processes for one of these partitions. It has to be kept in mind that the weighted average per device (and video type when the entire dataset is considered) is reported in the following tables to be able to compare this model with the others.

This model was quicker to train than the blind model, as the number of samples in each group are a fraction of the dataset available in that model. Times less than a minute were reported for some configurations, while the slowest was around three minutes.

The average results, weighted by the total amount of samples per partition (refer Table 6.7), are shown in Table 6.13. To keep it consistent with the other experiments, three groups are made: for the entire dataset, only for point videos and only for route videos. As expected, the weighted average of the results for point videos and route videos coincides with those for the entire dataset.

Table 6.13. Average great-circle distance (in degrees) for the grouped model to predict a future orientation.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	19.88	25.02	28.40	29.14	37.26	42.57	16.99	21.38	24.38
8	18.47	22.79	25.45	27.26	33.77	36.80	15.98	19.82	22.54
12	16.83	20.03	21.83	24.76	29.12	30.71	14.79	17.82	19.78

The average great-circle distance grows the farther the prediction is made (larger F). An increase of the window of past values (larger L) contributes to reduce the average distance, especially when both F and L grow. Results by video type differ but follow the same trend than in previous models.

Table 6.14. Difference in the results (in degrees) between the grouped model and the persistence model.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	0.30	-0.59	-1.65	0.64	-0.01	-1.20	0.19	-0.76	-1.77
8	-0.73	-2.38	-4.12	-0.72	-2.87	-6.23	-0.73	-2.25	-3.57
12	-2.04	-4.68	-7.19	-2.69	-6.53	-10.98	-1.87	-4.23	-6.32

If we compare these results with the persistence model (see Table 6.14), the grouped model generally performs better. The baseline is only better than the grouped model in the shortest configuration of L and F, which makes sense because a persistent prediction will be closer to the real value if it is not so far away (F) and because with less data available (L), the neural network model cannot learn as much.

Although less samples per run were available, since the dataset was partitioned, this model seems to perform as good as or even better than the blind model. We will discuss this further in the next pages.

6.2.5. Informed model

This is the last model we developed to predict a future orientation from a window of past values. Since there is an apparent effect of the video type and device used to play it on the performance of the prediction, we proceeded to label the samples with the categorical information that we have available along the third axis in Figure 6.15.

x	y	z	p	r	oc	pc	mb	mv
.4924	.1736	.8529	1	0	0	1	0	0

Figure 6.23. Example of the features in a timestep of a sample for the informed model.

Two pieces of categorical data are one-hot encoded: the video type and the device used to play the video, for a total of two and four features each (see in Figure 6.23 the features for a timestep: the three cartesian coordinates (x, y, z), the video type (p, r), and the device used to play it (oc, pc, mb, mv). When the data is partitioned by video type, the features about the video type are not included, as they would be redundant with the partition.

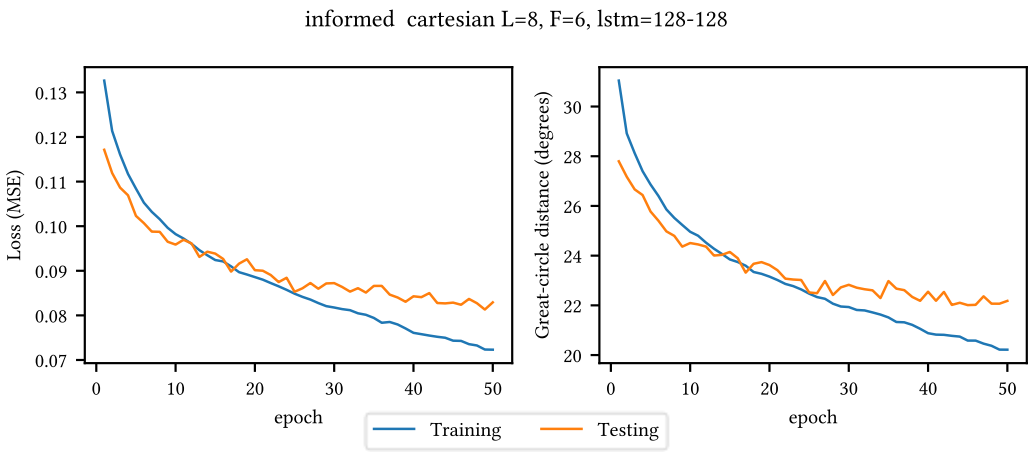

Figure 6.24. Evolution of the training and testing processes of the informed model for $L = 8, F = 6$ and cartesian coordinates.

The evolution of the training and testing processes for this model (see Figure 6.24) is similar to those of the blind model (Figure 6.21): 20 epochs might have sufficed for this combination of L and F.

This model showed training times very similar to the ones reported for the blind model, so including these additional features does not seem to have had an impact on its performance.

Table 6.15. Average great-circle distance (in degrees) for the informed model to predict a future orientation.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	18.82	24.09	27.83	26.78	34.60	40.59	16.59	21.25	24.00
8	17.77	22.00	24.71	25.03	31.06	34.84	15.59	19.60	22.05
12	16.38	19.57	21.40	22.04	25.41	27.09	14.80	17.86	19.79

We can appreciate the usual behavior in the results of this model (see Table 6.15): the farther the prediction (larger values of F), the greater the distance, but the more data is available (larger values of L), the better results are obtained. When high values of L are used, the increase in the average great-circle distance is less than with lower values of L. The divide between point and route videos can also be observed in this model, as future orientations in point videos are more difficult to predict.

Table 6.16. Comparison of the results for the informed model to predict a future orientation using the entire dataset, the weighted average of the partitioned dataset and the improvement of using such partitioning.

	Entire dataset			**W.A. of partition**			**Improvement**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	18.82	24.09	27.83	19.01	24.31	27.67	1.0103	1.0092	0.9942
8	17.77	22.00	24.71	17.68	22.04	24.66	0.9948	1.0017	0.9981
12	16.38	19.57	21.40	16.28	19.34	21.16	0.9938	0.9882	0.9887

As we did with the blind model, we can calculate the weighted average of the results for point and route videos and compare them with the results for the entire dataset (see Table 6.16). In this case, for most combinations of L and F, the partition does not bring any improvement, although a meager improvement of an average of 0.24 degrees is obtained with $L = 12, F = 8$. This is reasonable, as splitting the dataset does not bring any advantage to this model, as the information about which video type is being played is already included as a feature.

We can compare the results of this model with the persistence model (see Table 6.17): we can observe that the informed model performs better than the baseline in all the cases and that this improvement is higher the higher the values of both L and F are, because there is more data available for the model to make its prediction (larger values of L) and with a farther away prediction (larger values of F), the persistence model will be less accurate, while a neural network can learn how to predict it better.

Table 6.17. Difference in the results (in degrees) between the informed model and the persistence model.

	Entire dataset			Only point videos			Only route videos		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	-0.76	-1.52	-2.22	-1.72	-2.67	-3.18	-0.21	-0.89	-2.15
8	-1.43	-3.17	-4.86	-2.95	-5.58	-8.19	-1.12	-2.47	-4.06
12	-2.49	-5.14	-7.62	-5.41	-10.24	-14.60	-1.86	-4.19	-6.31

We can observe an improvement of an average of 7.62 degrees in the most extreme case for the entire dataset (26.26% better than the baseline), which is translated to an average of 14.60 degrees for point videos (35.02% better than the baseline) and 6.31 degrees for route videos (24.18% better than the baseline).

6.2.6. Comparison and discussion

Once we have analyzed all the models to predict a future orientation, we will compare them in order to answer the research questions we have formulated.

To answer RQ1 (Can we predict future orientations with neural network models?), we will compare the average results of these models with the results obtained by the baselines, which were simpler models.

Following Figure 6.25 we can compare the results of all the experiments. It can clearly be seen that the linear model is the one that performs the worst in all the cases. The persistence model is usually the second in performance, in some cases very close to the neural network models.

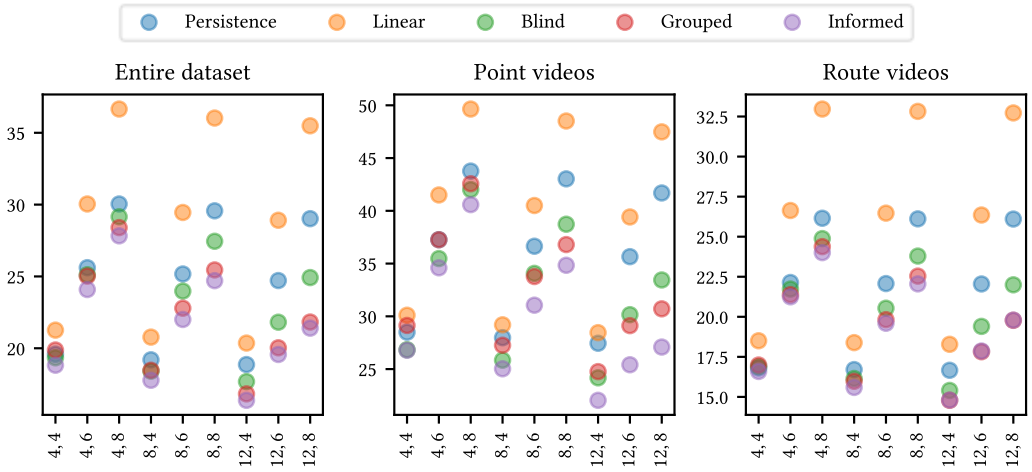

Figure 6.25. Comparison of the results of all the models with the entire dataset and each partition by video type for each combination of L and F.

For low values of L and F, as we have previously shown, the deep learning models perform similarly or some even slightly worse than the persistence model. This is especially true in route videos, as the average movement is less than in point videos, because many future viewports will be very close to the past ones.

As soon as the prediction is done farther (higher values of F) or with more data (higher values of L), the deep learning models perform better than the persistence model, as we can observe in Figure 6.25.

For this reason, we answer RQ1: we can predict future viewports with neural network models, especially when enough data is provided (higher values of L) or when it is more difficult to make an accurate prediction (higher values of F).

Although we have partially answered RQ2 (Can we achieve better results with more data (higher values of L)?) now, we study in Figure 6.26 the effect of L in the prediction of a future orientation.

We can see in Figure 6.26 that all the models improve when more data is available to perform the prediction (larger L). We have noted that the improvement for the baselines is not related to the

amount of data to perform the prediction, as they only use one value in the persistence model and in the linear model, so it is due to the slight reduction in the number of samples in the dataset. It can also be noted that the effect is bigger the farther away the prediction is made (F).

To assess if L has a significant effect in the prediction, we have performed a series of one-sided Mann-Whitney U tests with the results of the ten executions of each experiment with increasing values of L. Under the null hypotheses, the distributions of both populations are equal and under the alternate hypotheses, the distributions with higher values of L are less than lower values of L. The null hypotheses were rejected in all the cases with a significance level of $p < 0.001$, so we can accept the alternate hypotheses and conclude that the answer to RQ2 is that with higher values of L a better prediction is made. This is true when the entire dataset is considered at once and when it is partitioned by video type. The rank-biserial correlation was used to measure the effect size, which is $r = -1.0$ in all the cases except with $F = 4$ in the blind model, in which $-1 \leq r \leq -0.78$.

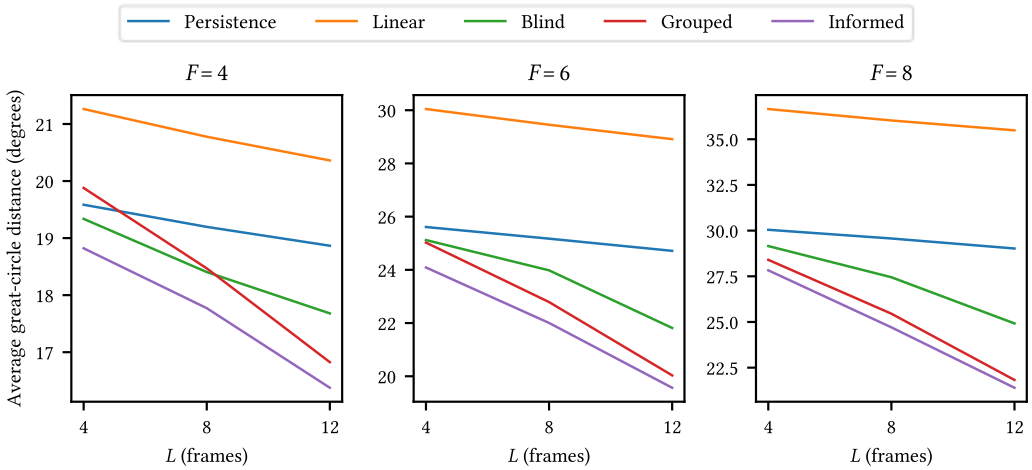

Figure 6.26. Effect of L in the prediction of a future viewport for the entire dataset.

For the sake of completeness, in Figure 6.27, we can observe how the distance at which the prediction is made affects its performance. Here, we can notice that with larger L, the performance of the deep learning models (specially in *Grouped* and *Informed*) does not degrade as much than that of the baselines.

To answer RQ3 (Does a neural network model improve its results if only samples with the same video type and device are used?) we compare the results obtained by the blind and grouped models.

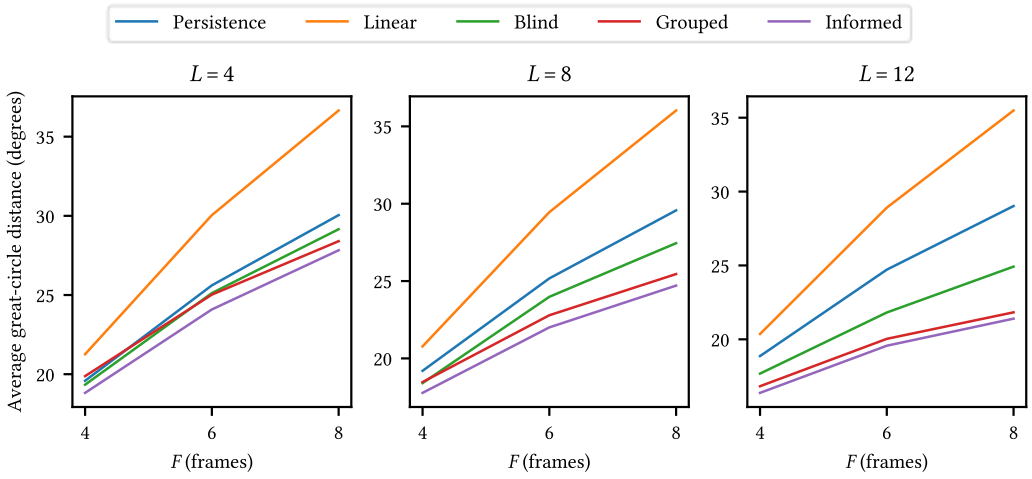

Figure 6.27. Effect of F in the prediction of a future viewport for the entire dataset.

A series of one-sided Mann-Whitney U test were performed. Under the null hypotheses, the distribution of the results of the grouped model and the blind model are equal, while under the

alternate hypotheses, the distribution of the results of the grouped model is less than the distribution of the results of the blind model. For the sake of completeness, the opposite test was also performed. The p-values are reported in Table 6.18 and Table 6.20, and the effect sizes are reported in Table 6.19.

Following Table 6.18, in many cases, especially at higher values of both L and F, the null hypothesis can be rejected at $\alpha = 0.05$ and we can state that the grouped model performs better than the blind model, even when the first has the handicap of a reduced number of samples with which to train the neural network model. In some cases, the comparison is the opposite, and the blind model performs better (see Table 6.20), because the combination of a reduced number of samples and a reduced amount of timesteps per sample make it more difficult for the grouped model to perform well. This is more prevalent in point videos, as a prediction in that video type is more difficult to make. The effect sizes reported in Table 6.19 are moderate or high when either set of hypotheses are rejected.

Table 6.18. P-values of the one-sided Mann-Whitney U test (grouped < blind).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	0.9943	0.5451	< 0.001	0.9999	0.9997	0.9190	0.7863	0.0188	0.0086
8	0.7637	< 0.001	< 0.001	0.9993	0.3669	< 0.001	0.1365	< 0.001	< 0.001
12	< 0.001	< 0.001	< 0.001	0.9555	0.0070	< 0.001	< 0.001	< 0.001	< 0.001

Table 6.19. Effect sizes (rank-biserial correlation) of the one-sided Mann-Whitney U test (grouped < blind).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	0.660	0.020	-0.960	1.000	0.900	0.360	0.200	-0.560	-0.640
8	0.180	-0.980	-1.000	0.840	-0.100	-0.840	-0.300	-0.880	-1.000
12	-0.980	-1.000	-1.000	0.440	-0.660	-0.980	-0.920	-1.000	-1.000

Table 6.20. P-values of the one-sided Mann-Whitney U test (blind < grouped).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	< 0.01	0.4849	0.9999	< 0.01	< 0.01	0.0929	0.2363	0.9844	0.9930
8	0.2603	0.9999	0.9999	< 0.01	0.6612	0.9993	0.8793	0.9996	0.9999
12	0.9999	0.9999	0.9999	0.0521	0.9943	0.9999	0.9998	0.9999	0.9999

In the end, the answer to RQ3 is that except for the lowest values of L and F, the grouped model performs better than the blind model at $\alpha = 0.05$. We can appreciate an average improvement of up to 13.40% for the entire dataset when $L = 12, F = 8$ (see Table 6.21).

Table 6.21. Ratio between the average great-circle distances of the grouped and the blind models.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	1.0279	0.9960	0.9743	1.0857	1.0502	1.0133	1.0059	0.9839	0.9799
8	1.0038	0.9504	0.9271	1.0550	0.9912	0.9502	0.9895	0.9649	0.9475
12	0.9519	0.9184	0.8760	1.0244	0.9655	0.9181	0.9604	0.9186	0.8991

To answer RQ4 (Does a neural network model improve its results when the information about which video type and device were used are included?) we will compare the informed model with both the blind model first and then the informed model, using the same procedure as previously.

Table 6.22. P-values of the one-sided Mann-Whitney U test (informed < blind).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	0.0036	< 0.001	< 0.001	0.5451	0.0270	0.0023	0.0270	0.0129	< 0.001
8	0.0086	< 0.001	< 0.001	0.0070	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
12	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	0.0023	< 0.001	< 0.001

We can observe in Table 6.22 that in all the cases except with $L = 4, F = 4$ when only point videos are considered in the dataset, the null hypotheses for the Mann-Whitney U tests are rejected at $\alpha = 0.05$ and we can conclude that the informed model performs better than the blind model. In that specific case, we cannot reject the null hypotheses for the reverse case either. The effect sizes reported in Table 6.23 are again high or moderate when the hypotheses are rejected.

Table 6.23. Effect sizes (rank-biserial correlation) of the one-sided Mann-Whitney U test (informed < blind).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	-0.720	-0.900	-1.000	0.020	-0.520	-0.760	-0.520	-0.600	-0.860
8	-0.640	-1.000	-1.000	-0.660	-1.000	-1.000	-0.840	-0.940	-1.000
12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-0.760	-0.980	-1.000

Table 6.24. Ratio between the average great-circle distances of the informed and the blind models.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	0.9731	0.9590	0.9547	0.9978	0.9752	0.9662	0.9822	0.9779	0.9646
8	0.9658	0.9174	0.9002	0.9687	0.9117	0.8996	0.9653	0.9542	0.9269
12	0.9265	0.8973	0.8587	0.9119	0.8425	0.8099	0.9610	0.9206	0.8995

Per Table 6.24, we can see that there is an average improvement of up to 14.13% for the entire dataset with $L = 12, F = 8$. It can also be noted that in point videos, this model performs much better than the blind model (up to 19.01% better on average with $L = 12, F = 8$), so knowing which device was used to play the video produces an important improvement on average.

When we compare the informed model with the grouped model, there is not as much improvement. We use the same statistical test to assess if the distribution of the average results of the informed model is less than that of the grouped model.

Following Table 6.25, it can be observed that we can reject the null hypotheses at $\alpha = 0.05$ in all the cases with the entire dataset and when only point videos are considered; when only route videos are considered, this is not so clear. In that case, we cannot reject the null hypotheses when $L = 12$ or when $F = 6$. In these cases, the grouped model performs as good as the informed model (the opposite case cannot be rejected: we cannot say that the grouped model performs better than the informed model either at $\alpha = 0.05$). This is because future orientations in route videos are easier to predict, so even with the reduced number of samples that are present in the grouped model, it is strong enough to make an accurate prediction.

Table 6.25. P-values of the one-sided Mann-Whitney U test (informed < grouped).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	< 0.001	< 0.001	0.0086	< 0.001	< 0.001	< 0.001	0.0156	0.1923	0.0188
8	0.0011	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	0.0070	0.0606	0.0320
12	< 0.001	0.0023	0.0226	< 0.001	< 0.001	< 0.001	0.7397	0.4251	0.6331

Similarly to the previous experiments, the effect sizes reported in Table 6.26 are smaller when we cannot reject the hypotheses, but high when we can.

Table 6.26. Effect sizes (rank-biserial correlation) of the one-sided Mann-Whitney U test (informed < grouped).

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	-1.000	-0.900	-0.640	-1.000	-1.000	-0.880	-0.580	-0.240	-0.560
8	-0.820	-0.960	-0.960	-0.980	-1.000	-1.000	-0.660	-0.420	-0.500
12	-0.860	-0.760	-0.540	-1.000	-1.000	-1.000	0.160	-0.060	0.080

The improvement between informed and grouped can be observed in Table 6.27. We can see that it is slim when the entire dataset is considered and between that and non-existent when only route videos

are considered, but there is a sizeable improvement when only point videos are considered, of up to 11.79% with $L = 12, F = 8$. In these cases, the neural network model benefits from the increased number of samples, as it can learn the differences between devices and use them to improve the prediction.

Table 6.27. Ratio between the average great-circle distances of the informed and the grouped models.

	Entire dataset			**Only point videos**			**Only route videos**		
L	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**	***F* = 4**	***F* = 6**	***F* = 8**
4	0.9467	0.9628	0.9799	0.9190	0.9286	0.9535	0.9765	0.9939	0.9844
8	0.9621	0.9653	0.9709	0.9182	0.9198	0.9467	0.9756	0.9889	0.9783
12	0.9733	0.9770	0.9803	0.8901	0.8726	0.8821	1.0007	1.0022	1.0005

In the end, we can answer RQ4: the informed model tends to provide better results than both the blind and the grouped models.

In general, each successive neural network model performs better than the previous one. As we can observe in Figure 6.28, we can see how the grouped model performs better than the blind model and the informed model tends to perform better than the other two when $L = 8, F = 6$.

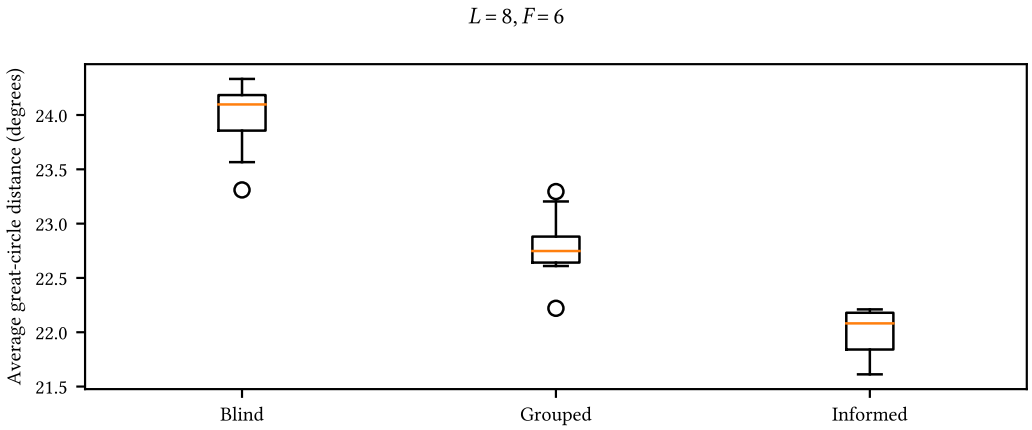

Figure 6.28. Boxplot of the results obtained by the three neural network models with the entire dataset and $L = 8, F = 6$.

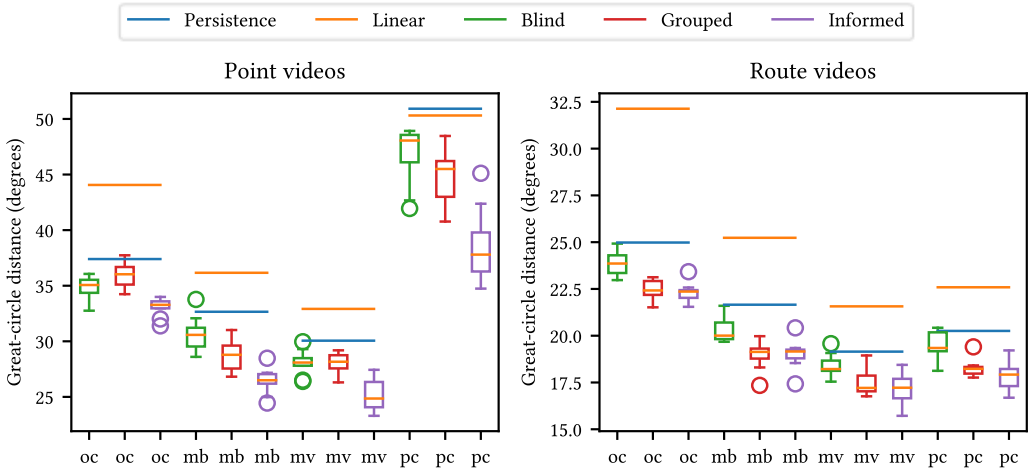

Figure 6.29. Results of the five models grouped by video type and device for $L = 8, F = 6$.

To assess the differences between models in each device, we evaluated the models with samples of each video type and device. A sample for $L = 8, F = 6$ is shown in Figure 6.29. We can observe that all three neural network models perform better than the baselines in all the cases. We can also observe

that the general trend is followed here by device, as apparently the grouped model tends to perform better than the blind model and the informed model tends to perform better than the other two.

One-sided Mann-Whitney tests were performed to evaluate which models perform better in each video type and device for $L = 8$, $F = 6$. Null hypotheses were rejected in the following cases (p-values and rank-biserial correlations are reported): grouped achieves better results than blind in route videos with all devices ($p < 0.01, r \in [-0.92, -0.72]$,)) and in point videos with non-immersive interfaces: *mb* ($p < 0.01, r = -0.66$) and *pc* ($p = 0.044, r = -0.46$); blind is only better than grouped in point videos with *oc* ($p = 0.027, r = -0.52$); informed performs always better than blind ($p < 0.01, r \in [-1, -0.72]$) and informed gets better results than grouped in point videos with all devices ($p < 0.01$, $r \in [-1, -0.86]$). We cannot reject the null hypotheses in the other cases, which leaves inconclusive results between the blind and grouped models in point videos with *mv* and between the grouped and informed models in route videos for similar reasons to what we discussed before.

In the end, to answer RQ5, there is an evident difference in the results per video type and device. Furthermore, the relative results of the different models per video type and device are not the same: we can observe how the grouped model in point videos with non-immersive interfaces performs better than the blind model, but the reverse is true in the *oc* interface; similarly, results are inconclusive between the grouped and the informed model in route videos with any device.

6.3. Conclusion

In this chapter, we used the data collected in the reproduction of 360-degree videos to run two experiments: we first classified the reproductions into their contextual variables (video type and device which was used to play the video) and then we used that data to predict the future orientations and we improved the results of the classification by leveraging that same contextual information.

We have observed how a recurrent neural network provides better results in this kind of problems, as they deal with a trajectory of data. We also observed how we can obtain vastly better results with untreated data (the raw value of the users' orientation, in spherical or cartesian coordinates)

We have seen how we could correctly classify with a high degree of accuracy (90%) the video type and the device which was used in the reproduction with only four seconds of trajectory with a recurrent neural network model based on LSTM.

These results are encouraging: a deep learning model is able to tell the differences between video types and devices with a high degree of accuracy, but we can also reproduce this contextual data from trajectories which do not have it available, such as third-party datasets or to be used in streaming software that feeds on data that does not take this information into account.

Regarding to the prediction of future orientations, we have implemented and analyzed five models, based on a window of past observations (L) and the distance in which the prediction is made (F). We have observed that the spherical representation of the data, although suitable to classify the reproductions, does not work well to perform a prediction of future data: while the mean squared error of the coordinates in the cartesian representation works towards minimizing the spherical distance, it does not seem to happen with the spherical representation and the model is unable to learn properly.

We have seen that neural network models perform better than the baselines (RQ1), although the results are mixed with lower values of L and F, especially in route videos, we have shown that in the other cases, they do perform better. This special case is due to the combination of an easier to predict video type (user behavior route videos show less variation than in route videos); an easier to perform prediction (a lower F often makes the future value to be closer to the last value); and a more limited window of past data (a lower L which hinders the learning capacity of deep learning models).

Relative to this last point, we have observed that providing more data to the deep learning models does improve its performance in all the cases (RQ2).

The successive improvement between models is because the differences between video types and devices that we have already highlighted: with the blind model, there is no information about what video type or device is used; with the grouped model, we split the data, so each prediction is very

specific, at the cost of a reduced set of samples; with the informed model, we can use all the samples in the dataset and the information about the video type and device is used to improve the prediction. When we partition the dataset by video type, the only improvement that the grouped model brings over the blind model is that only samples with the same device are trained together. This improves the results of the blind model but is severely affected by a reduced number of samples. The informed model brings the solution, as all the samples of all the devices are used and, since they are labelled accordingly, a better prediction than with both the blind and grouped models can be made (especially in point videos, which are more difficult to predict).

The implications of the answers of RQ3 and RQ4 are that yes, knowing which device was used to play the video and taking it into account when making the prediction (either by dealing with groups of only that device or by labelling them in the features) improves the accuracy of their predictions. We have seen that, although the grouped model performs better than the blind model, in some cases, the informed model performs even better. This is due to the fact that the blind model sacrifices learning capacity (number of samples) for a more accurate learning (samples of only the relevant kind). The informed model provides the best of the two worlds, as we can utilize the entire dataset and the deep learning model learns which categories help it to perform up to 19% better.

Finally, as seen in RQ5, these results differ when the models are evaluated with data of different video types and devices. Although, in some cases, the grouped model does not perform better than the blind model (probably due to the issue of a reduced sample size), the informed model always performs better than the blind model in a statistically significant way. The grouped and the informed model perform similarly in route videos, as the number of samples in the grouped model is moderate and the prediction is easier to make.

In the end, the conclusion is similar: better results may be achieved by reducing the sample set and increase its specificity by only including samples with the desired contextual data, but this may be counter-productive, as these kinds of models need a high volume of samples to train properly. By using an informed model, the specificity of the model is retained in the form of additional labels without sacrificing volume of samples, so better results are obtained.

In any case, we have demonstrated that a better prediction can be made if we leverage the fact of knowing which device is being used to play a 360-degree video, which was the primary purpose of this study. Furthermore, we have seen that if that contextual information is missing, we are able to replicate it with our classification models with a high accuracy.

Chapter 7. Conclusion and future work

Throughout this thesis we have collected and studied the users' behavior in virtual reality environments and 360-degree videos with various interfaces based on different devices.

We have reported the automatic collection of user behavior in VR multimedia, facilitated by an omnidirectional multimedia system. We used this system to build interactive virtual experiences, play them in multiple devices and obtain their behavior, so we could later analyze it. This system was evolved in four different experiments, two of which are analyzed in depth.

This system achieved its goals in each of the experiments throughout its evolution: interactive experiences could be designed with ease, they were able to be played by users with an increasing number of compatible devices, and their behavior was able to be extracted and analyzed with a variety of tools. The reproduction of these multimedia experiences and the collection of users' behavior were successful.

While the specifics of some parts of the system varied between iterations, the core in which the collection of user behavior is based on did not: the exchange of information between the application that plays these interactive experiences and the behavioral database is based on the Experience API (xAPI), which was extended to properly work with each of the experiments, with their particular differences, with a common ground. In fact, the behavioral database, the Learning Record Store (LRS), was not modified at all during the evolution of the system, as it is general enough to store any experience as long as it follows the xAPI format.

The use of xAPI proved to be a success, as this data was correctly collected, stored, and retrieved in all the cases, making it possible to perform analysis on the users' actions in these experiments.

The greatest benefit of this approach was that it made it possible the division of work between the development of the Learning Record Provider (LRP) and the Learning Record Consumer (LRC), which in our system are the applications that are used to play these interactive experiences (which are reported in Chapter 3) and the application that is used to analyze the data (which is reported by our colleague in [93]). Note that the behavioral data was exported from that application for further analysis in the last two experiments and is reported in later chapters in this thesis.

Four experiments were produced, based on an increasingly evolved player application, which were used to perform an initial analysis on the collected behavior and provided valuable insight in the production of immersive experiences.

The first experiment was introduced as a means to play interactive educative activities based on 360-degree videos, which were to be played on computers with a drag-based interface. We noticed the issues with video quality and the particularities with spherical coordinates, as plane interpolation was not sufficient to move an object over the sphere in which a 360-degree video is displayed. In the end, this iteration served as a demonstration of the collection of user behavior with xAPI, as we showed how we could reproduce the events that happened during the reproduction of the 360-degree video activity.

The second iteration served to reproduce a hypervideo based on a main 360-degree video track with multiple regular videos linked in specific points in the spacetime of the main spherical video track. While the experience was still to be played on computers with a drag-based interface, efforts were made to improve the video quality with MPEG-DASH and the positioning of the markers on the sphere was accurately made with great-circle formulas. This iteration served to study the spatiotemporal consumption of the 360-degree video track, the temporal consumption of the linked regular videos and the selection of the markers, and compare them between users who registered their personal details and those who did not, reported in [93]. Notably, markers that appear sooner are more likely to be selected; the more time a user watches the main video track, the more markers they select (even

more in registered users); and the spatial exploration in the main 360-degree video is significantly correlated to the selection of markers when in registered users, but it is not when they did not register their details. A difference in engagement between the two classes of users seems evident at this point: registered users behave more actively, while unregistered users behave more passively, in general.

The third step consisted in a virtual reality environment, not based on 360-degree videos, in which we introduced a virtual museum of antique Macintosh computers. In this case, we first showed how we could extend our behavior collecting system based on xAPI from 360-degree videos a to general-purpose VR application and we finally introduced multiple devices and four specialized user interfaces for them: a computer interface based with a mouse to orientate (similar to the one in previous experiments, but this time the cursor is captured, instead of having to drag across the scene) and select objects and a keyboard to move around the scene; a mobile interface that leveraged the sensors in the device to change the user orientation in the virtual environment and relied in the selection of objects to move around; an immersive interface for mobile VR visors (such as the Google Cardboard), which functioned similarly to the previous one, but relied in a gaze control to select objects, instead of tapping the touchscreen of the device; and an immersive interface for standalone VR headsets with controllers (such as the Oculus Rift), which used the sensors in the visor to orientate the user, the joystick in the controllers to move around and a gesture to select objects. In a user experiment, we first noticed difficulties in the interaction with the immersive interfaces, especially with the gesture required to select objects in the oculus interface, and we also collected the users' behavior with the last two of these four interfaces. We observed some initial differences between these devices, which had been reviewed in a later chapter.

The fourth and final case marked the return to spherical videos with the 360-degree hypervideo platform. We leveraged the advancements made in the second iteration (adaptive streaming and great-circle equations) and in the third evolution (multiple devices and immersive interface) to create a general hypervideo format and demonstrated it with a tour through the streets with two kinds of videos (point and route videos), which were played with the same four devices than in the previous experiment, adapted for this experiment (one cannot move around in a 360-degree video), and in this case, the mobile interface relied in dragging over the touchscreen instead of on the sensors and in the oculus interface, instead of performing a gesture, a laser-based control was used to select the markers. We noticed differences in the characteristics of both kinds of videos and in the usability of the four interfaces after a user experiment. We collected the users' behavior, which showed initial results of difference in the dispersion of the users' orientation between video types and devices that had been reviewed more in-depth in a later chapter.

Overall, we have seen that interaction in immersive experiences such as 360-degree videos and virtual reality are worth of study, and that is impossible without collecting the users' behavior. We have presented a way of doing that automatically, in a way that could be extended through four different use cases, and that could be further extended to support future productions.

With the evolution of this system, we satisfied Objective 1, as we could collect the users' behavior with multiple devices in virtual reality environments and 360-degree videos.

We studied the difference in behavior in the virtual museum (third iteration) between the interfaces of the immersive devices *cardboard* and *oculus*, which showed that different interfaces, although similar (as they were both for immersive devices), may produce differences in behavior.

We found a significant difference in the time between interactions between the virtual devices: selections in *cardboard* were more frequent than in *oculus*, which was probably due to the more complicated way of selecting objects in oculus, as they needed to perform a specific gesture.

Selection patterns of virtual objects differed between devices: the distribution of selections was always significantly different between them. We noticed how the selection of objects in *cardboard* was significantly inversely correlated with the distance from the center of the environment, while in *oculus*, it was significantly inversely correlated with the value of the z axis of the objects' position.

We observed how *oculus* users were more spatially disperse in the environment than *cardboard* users, with a statistically significant difference between the positions of the two interfaces. Users of the

cardboard interface were often based in the middle of the scene, while behavior with the *oculus* interface was often made towards the northern part of the environment.

These two last observations are linked, as users selected objects in the positions where they explored. The difference may lie in the difference of the interaction between interfaces: *cardboard* users started in the middle of the scene, so it is normal that the most explored area is the middle, while *oculus* users continued where the previous user left, so this center bias should be eliminated. Instead, we observed that they explored the northern part of the environment.

We analyzed the orientation of the users' heads and saw how, on average, *oculus* users looked towards the northern part of the scene, while *cardboard* users looked towards the eastern part. Orientation in *cardboard* was significantly more disperse than in *oculus*.

The general direction of the users of the *oculus* interface can be traced to the relative orientation in which we set up the monitors and sensors for that device, so that needs to be taken under consideration when virtual reality experiences are designed.

After grouping the behavior of the *cardboard* interface by the web browser used, we noticed differences in the overall orientations per web browser, but also in the orientation of the first selection in each session, which could be caused by differences per browser in the initialization of the sensors of the device. We observed how the dispersion was smaller when it was based on the orientation of that first selection than when it is based on the centroid of the orientations, so that first orientation must have played a role in the overall orientation of the users. In the end, we found that the difference in dispersion between the Android browser and either iOS browsers was significant, but it was not significant between both iOS browsers.

In the end, differences in the interaction, like the difference to perform a selection, differences in the initial values, like the starting position or the initial orientation, even miscellaneous differences like the presence of a physical element that could influence the general orientation (the table in the *oculus* interface), can make a difference in behavior between devices and must be considered if a uniform experience (within reason) is to be achieved in all the devices.

We studied the users' behavior in the 360-degree videos of the immersive hypervideo (fourth iteration) with four interfaces (*pc*, *mb*, *oc*, *mv*), which resulted in very evident differences in behavior between video types (whether the camera is static or not in the video) and more subtle differences between devices.

We first compared the spatiotemporal exploration of the video sessions. Route videos were significantly more temporally explored than point videos and that between devices, the differences were only significant between some devices. Then we found that spatial exploration with immersive devices (*oc*, *mv*) tends to be lower than with non-immersive devices (*pc*, *mb*); in fact, between video types, we could not find a significant difference with the immersive devices, while in non-immersive devices, we found that route videos were significantly more explored spatially. Finally, we found that temporal and spatial exploration are significantly moderately correlated with the same combination of contextual variables (video type and device), with more similar values in immersive devices than in non-immersive devices. We understand that this is logical, as interaction in those devices is constrained by the same parts of the human body, while in the others, the changes of orientation can be performed by a mouse or a touchscreen.

We analyzed the dispersion of the users' orientation and we found that users are significantly much more disperse in point videos than in route videos and that there are small, significant differences by device per video type: in route videos, users of non-immersive devices showed a greater dispersion than users of immersive devices, while in point videos, the opposite was true (except between *mb* and *mv*).

We also analyzed the movement of the users' heads in segments of various lengths, and we found it to be significantly larger in point videos than in route videos. Between devices, the movement with *pc* was found to be significantly greater than the other devices in point videos, while in route videos the values are quite similar, especially between immersive and non-immersive devices. This difference

in *pc* is thought to have been able to be made thanks to the ease of rapidly changing the orientation with a simple movement of the mouse.

To conclude that study, we analyzed the trajectories described by the users by applying clustering method based on trajectories on the reproductions of each video. We analyzed the average relative size of the clusters and the relative size of the main cluster in multiple trajectory windows. We observed significant differences between video types: the average relative size of the clusters and the relative size of the main cluster are smaller in point videos than in route videos, as users are more disperse in the first category, more clusters per user are created. Between devices, the results are not so clear, but in all the cases, the results were significantly better when we only used a single device than when we clustered all the devices together, which proves that different devices result in different trajectories.

These two studies, one based on a virtual reality environment, and another based on 360-degree videos, helped us complete Objective 2 by comparing the behavior between different interfaces in these multimedia productions.

Based on the same behavioral data from the fourth iteration, we ran two deep learning experiments, one to classify the reproductions into their contextual variables (video type and device), and another to improve a prediction model of future orientations with that contextual data.

We observed that a recurrent neural network based on trajectory of raw orientation coordinates performs better than other models (like a dense model) or based on other data (like the movement in segments). In the end, we could correctly classify with a high degree of accuracy (greater than 90%) the video type and the device used in the reproduction with only four seconds of trajectory with either the spherical representation or the cartesian representation of the orientation.

These results evidenced the difference between video types and devices, as a neural network model was able to find them from the raw coordinates of orientation to classify the trajectories.

In the prediction model, we have found that those models based on deep learning perform well, significantly better than the baseline models. We observed how a larger window of data helps improve the performance of the models.

We designed three deep learning models in total: in blind, all the samples were used altogether without any distinction; in grouped, only samples with the same contextual variables were considered; and in informed, all the samples were also considered altogether but they were labelled with the contextual variables (video type and device).

We found that knowing these contextual variables and acting accordingly can be used to improve the prediction. We first saw that the grouped model performed significantly better than the blind model, as the differences between video types and devices outweigh the penalties of having a reduced training set. Then we observed that the blind model performs even better than the grouped model, especially in point videos, as they are more difficult to work with, and there are fewer samples to train the model.

Finally, these results differ per video type and device: in some devices, the grouped model may perform worse than the blind model, probably due to an excessive reduction of the sample size, but the informed model always performs significantly better than the blind model. The difference between the grouped and the informed models in route videos is not significant, as the number of samples in the grouped model is considerable and the prediction is easier to make.

In the end, we demonstrated how a prediction model for the users' orientation in 360-degree videos can benefit from the contextual variables of the trajectories and that, if these values are missing, we can guess them with more than 90% of accuracy. With these results, we mark Objective 3 as complete.

After we satisfied the three objectives we set up in 1.1, we have been able to collect the users' behavior with multiple devices in virtual reality environments and 360-degree videos, we have analyzed the differences in that behavior in a virtual reality environment and a set 360-degree videos, and we have seen how a model to predict the future orientations of a user in 360-degree videos can be improved by leveraging the contextual information we have recorded or we have shown how to recreate from the raw users' orientation.

This makes an advancement in the state of the art, as the users' behavior in different devices and interfaces are not compared in studies. We have shown a way to collect that behavior, that there are differences in it and that these differences can play a part in the prediction of the users' future orientations in 360-degree videos.

Although the objectives were fulfilled and positive results were obtained, we need to reflect on some limitations of this work.

The studies performed are made ad-hoc for our datasets, so if a new experiment was designed or a third-party dataset was to be used, new scripts to analyze these new datasets would need to be created. An interactive dashboard to dynamically analyze this data would be desirable but fell out of the scope of this work.

The study described in Chapter 4 only included immersive interfaces. It would have been interesting to compare immersive and non-immersive interfaces, as we did in the next study. Furthermore, the data on the Oculus interface had to be aggregated, so per-session analyses in that interface could not be made.

In Chapter 5 we based our study in an interactive hypervideo, but we only studied the users' orientation in the 360-degree videos. It would have been interesting to study that inter-video behavior. The existence of hyperlinks to jump from video to video made possible that users could skip parts of the videos, so our dataset is not uniform and that had to be addressed when we analyzed the users' orientations.

For both of these studies, we could not fully exploit the potential of the system we presented in Chapter 3, as the data was collected in experimental setups. Had we distributed these immersive productions, we could have obtained much larger datasets from users through the Internet.

Finally, we studied the classification of the users' trajectories and the use of these contextual information to improve the prediction of future orientations. To better assess the prediction models, we would need to use them in an adaptive streaming system. This also fell outside of the scope of this current work.

In the future, we would like to continue our work in this research line by extending the present work in the following aspects.

The implementation of an adaptive streaming technique that leverages our informed prediction model would be a good starting point. We have seen that if the contextual information is not available, we can use our classification model to reproduce it.

Regarding the contextual data, we could study if we can find more categories: there are probably more types of videos in other productions that are not limited to videos of the streets. In the same way, there are probably more classes of devices or ways to interact with them, as there are probably user-dependent factors, such as eagerness of exploration, which are independent of the device.

The deep learning models presented in this document are relatively simple, to research if the prediction made by a given model could be improved by leveraging that contextual information. An improvement of these models is a task in the future.

Although we created an interactive hypervideo experience with our own 360-degree videos, it would be interesting to perform another analysis without all the hypervideo characteristics with third-party 360-degree videos, either pulled from the video platforms from the Internet, such as YouTube, or available in research datasets, and observe if we can identify more video types and observe to what extent we can replicate our results with this new set of videos.

References

[1] G. Zoric, L. Barkhuus, A. Engström, and E. Önnevall, "Panoramic video," in *Proceedings of the 11th european conference on Interactive TV and video - EuroITV '13*, 2013, p. 153, doi: 10.1145/2465958.2465959.

[2] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual display," *IEICE Trans. Inf. Sys., D*, vol. 77, no. 12, pp. 1321–1329, 1994, Accessed: Nov. 24, 2021. [Online]. Available: https://ci.nii.ac.jp/naid/110003209335.

[3] W. R. Sherman and A. B. Craig, *Understanding virtual reality : interface, application, and design*. Morgan Kaufmann, 2003.

[4] M. Slater, "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments," *Philos. Trans. R. Soc. B Biol. Sci.*, vol. 364, no. 1535, pp. 3549–3557, Dec. 2009, doi: 10.1098/rstb.2009.0138.

[5] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: a class of displays on the reality-virtuality continuum," in *Telemanipulator and Telepresence Technologies*, Dec. 1995, vol. 2351, pp. 282–292, doi: 10.1117/12.197321.

[6] S. M. LaValle, *Virtual Reality*. 2020.

[7] R. Wojciechowski, K. Walczak, M. White, and W. Cellary, "Building Virtual and Augmented Reality museum exhibitions," in *Proceedings of the ninth international conference on 3D Web technology - Web3D '04*, 2004, p. 135, doi: 10.1145/985040.985060.

[8] Y. Zhao, M. Forte, and R. Kopper, "VR Touch Museum," in *25th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018 - Proceedings*, Mar. 2018, pp. 741–742, doi: 10.1109/VR.2018.8446581.

[9] A. Subramanian, J. Barnes, N. Vemulapalli, and S. Chhawri, "Virtual reality museum of consumer technologies," in *Advances in Intelligent Systems and Computing*, 2017, vol. 498, pp. 549–560, doi: 10.1007/978-3-319-42070-7_51.

[10] I. Giangreco *et al.*, "VIRTUE: A virtual reality museum experience," in *International Conference on Intelligent User Interfaces, Proceedings IUI*, Mar. 2019, pp. 119–120, doi: 10.1145/3308557.3308706.

[11] J. L. Lugrin *et al.*, "A location-based VR museum," in *2018 10th International Conference on Virtual Worlds and Games for Serious Applications, VS-Games 2018 - Proceedings*, Sep. 2018, pp. 1–2, doi: 10.1109/VS-Games.2018.8493404.

[12] M. Carrozzino and M. Bergamasco, "Beyond virtual museums: Experiencing immersive virtual reality in real museums," *J. Cult. Herit.*, vol. 11, no. 4, pp. 452–458, Oct. 2010, doi: 10.1016/j.culher.2010.04.001.

[13] F. Izzo, "Museum Customer Experience and Virtual Reality: H.BOSCH Exhibition Case Study," *Mod. Econ.*, vol. 08, no. 04, pp. 531–536, 2017, doi: 10.4236/me.2017.84040.

[14] H.-L. Chang, Y.-C. Shih, K. Wang, R.-H. Tsaih, and Z. Lin, "Using Virtual Reality for Museum Exhibitions: The Effects of Attention and Engagement for National Palace Museum," in *PACIS 2018*, Jun. 2018, p. 145, Accessed: Nov. 14, 2018. [Online]. Available: https://aisel.aisnet.org/pacis2018/145.

[15] G. Schofield *et al.*, "Viking VR: Designing a virtual reality experience for a museum," in *DIS 2018 - Proceedings of the 2018 Designing Interactive Systems Conference*, Jun. 2018, pp. 805–816, doi: 10.1145/3196709.3196714.

[16] M. Shehade and T. Stylianou-Lambert, "Virtual Reality in Museums: Exploring the Experiences

of Museum Professionals," *Appl. Sci.*, vol. 10, no. 11, p. 4031, Jun. 2020, doi: 10.3390/app10114031.

[17] A. Puig *et al.*, "Lessons learned from supplementing archaeological museum exhibitions with virtual reality," *Virtual Real.*, vol. 24, no. 2, pp. 343–358, Jun. 2020, doi: 10.1007/s10055-019-00391-z.

[18] E. Lantz, S. Bryson, D. Zeltzer, M. T. Bolas, B. de La Chapelle, and D. Bennett, "Future of virtual reality: Head mounted displays versus spatially immersive displays," 1996, doi: 10.1145/237170.237289.

[19] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen projection-based virtual reality: The design and implementation of the CAVE," 1993, doi: 10.1145/166117.166134.

[20] P. Rajesh Desai, P. Nikhil Desai, K. Deepak Ajmera, and K. Mehta, "A Review Paper on Oculus Rift-A Virtual Reality Headset," *Int. J. Eng. Trends Technol.*, vol. 13, no. 4, pp. 175–179, Jul. 2014, doi: 10.14445/22315381/ijett-v13p237.

[21] M. Slater, "Immersion and the illusion of presence in virtual reality," *Br. J. Psychol.*, vol. 109, no. 3, pp. 431–433, Aug. 2018, doi: 10.1111/bjop.12305.

[22] W. Powell, V. Powell, P. Brown, M. Cook, and J. Uddin, "Getting around in google cardboard – exploring navigation preferences with low-cost mobile VR," in *2016 IEEE 2nd Workshop on Everyday Virtual Reality (WEVR)*, Mar. 2016, pp. 5–8, doi: 10.1109/WEVR.2016.7859536.

[23] Y. LI, J. HUANG, F. TIAN, H. A. WANG, and G. Z. DAI, "Gesture interaction in virtual reality," *Virtual Real. Intell. Hardw.*, vol. 1, no. 1, pp. 84–112, Feb. 2019, doi: 10.3724/SP.J.2096-5796.2018.0006.

[24] S. Yoo and C. Parker, "Controller-less interaction methods for google cardboard," in *SUI 2015 - Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, Aug. 2015, p. 127, doi: 10.1145/2788940.2794359.

[25] Web3D Consortium, "The Virtual Reality Modeling Language," Apr. 12, 2002. https://www.web3d.org/documents/specifications/14772-1/V2.1/index.html (accessed Apr. 04, 2022).

[26] Web3D Consortium, "X3D Version 4 Overview | Web3D Consortium." https://www.web3d.org/x3d4 (accessed Apr. 04, 2022).

[27] L. Chittaro and R. Ranon, "Web3D technologies in learning, education and training: Motivations, issues, opportunities," *Comput. Educ.*, vol. 49, no. 1, pp. 3–18, Aug. 2007, doi: 10.1016/j.compedu.2005.06.002.

[28] W3C, "WebXR Device API," Jul. 23, 2020. https://www.w3.org/TR/webxr/ (accessed Aug. 14, 2020).

[29] W3C, "WebVR," Dec. 11, 2017. https://immersive-web.github.io/webvr/spec/1.1/ (accessed Aug. 14, 2020).

[30] Khronos Group, "WebGL Overview - The Khronos Group Inc," 2020. https://www.khronos.org/webgl/ (accessed Aug. 14, 2020).

[31] T. Pakkanen *et al.*, "Interaction with WebVR 360° video player: Comparing three interaction paradigms," in *2017 IEEE Virtual Reality (VR)*, 2017, pp. 279–280, doi: 10.1109/VR.2017.7892285.

[32] C. Dibbern, M. Uhr, D. Krupke, and F. Steinicke, "Can WebVR further the adoption of Virtual Reality?," in *Mensch und Computer 2018 - Usability Professionals*, 2018, pp. 377–384, doi: 10.18420/muc2018-up-0249.

[33] S. Gunkel, M. Prins, H. Stokking, and O. Niamut, "WebVR meets WebRTC: Towards 360-degree social VR experiences," in *Proceedings - IEEE Virtual Reality*, 2017, pp. 457–458, doi: 10.1109/VR.2017.7892377.

[34] H. Hadjar, A. Meziane, R. Gherbi, I. Setitra, and N. Aouaa, "WebVR based Interactive Visualization of Open Health Data," in *Proceedings of the 2nd International Conference on Web*

Studies - WS.2 2018, 2018, pp. 56–63, doi: 10.1145/3240431.3240442.

[35] U. Neumann, T. Pintaric, and A. Rizzo, “Immersive panoramic video,” in *Proceedings of the eighth ACM international conference on Multimedia - MULTIMEDIA ’00*, 2000, pp. 493–494, doi: 10.1145/354384.376408.

[36] W. LYU, Z. ZHOU, L. CHEN, and Y. ZHOU, “A survey on image and video stitching,” *Virtual Real. Intell. Hardw.*, vol. 1, no. 1, pp. 55–83, Feb. 2019, doi: 10.3724/SP.J.2096-5796.2018.0008.

[37] E. Kuzyakov and D. Pio, “Next-generation video encoding techniques for 360 video and VR,” *Facebook Code*, 2016. https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/.

[38] K. Kwiatek and M. Woolner, “Embedding interactive storytelling within still and video panoramas for cultural heritage sites,” in *VSMM 2009 - Proceedings of the 15th International Conference on Virtual Systems and Multimedia*, 2009, pp. 197–202, doi: 10.1109/VSMM.2009.36.

[39] L. A. R. Neng and T. Chambel, “Get around 360° hypervideo,” in *Proceedings of the 14th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek ’10*, 2010, p. 119, doi: 10.1145/1930488.1930512.

[40] M. Slater and M. V. Sanchez-Vives, “Enhancing Our Lives with Immersive Virtual Reality,” *Front. Robot. AI*, vol. 3, p. 74, Dec. 2016, doi: 10.3389/frobt.2016.00074.

[41] J. R. Williamson, D. Sundén, and J. Bradley, “GlobalFestival: Evaluating Real World Interaction on a Spherical Display,” *Proc. Jt. Int. Conf. Pervasive Ubiquitous Comput. Int. Symp. Wearable Comput.*, pp. 1251–1261, 2015, doi: 10.1145/2750858.2807518.

[42] S. Kasahara, S. Nagai, and J. Rekimoto, “JackIn Head: Immersive Visual Telepresence System with Omnidirectional Wearable Camera,” *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 3, pp. 1222–1234, Mar. 2017, doi: 10.1109/TVCG.2016.2642947.

[43] A. De Abreu, C. Ozcinar, and A. Smolic, “Look around you: Saliency maps for omnidirectional images in VR applications,” in *2017 9th International Conference on Quality of Multimedia Experience, QoMEX 2017*, May 2017, pp. 1–6, doi: 10.1109/QoMEX.2017.7965634.

[44] E. Upenik, M. Rerabek, and T. Ebrahimi, “Testbed for subjective evaluation of omnidirectional visual content,” in *2016 Picture Coding Symposium (PCS)*, 2016, pp. 1–5, doi: 10.1109/PCS.2016.7906378.

[45] V. Sitzmann *et al.*, “Saliency in VR: How Do People Explore Virtual Environments?,” *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 4, pp. 1633–1642, Apr. 2018, doi: 10.1109/TVCG.2018.2793599.

[46] M. Yu, H. Lakshman, and B. Girod, “A framework to evaluate omnidirectional video coding schemes,” in *Proceedings of the 2015 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2015*, Sep. 2015, pp. 31–36, doi: 10.1109/ISMAR.2015.12.

[47] X. Corbillon, F. De Simone, and G. Simon, “360-degree video head movement dataset,” in *Proceedings of the 8th ACM Multimedia Systems Conference, MMSys 2017*, Jun. 2017, pp. 199–204, doi: 10.1145/3083187.3083215.

[48] E. J. David, J. Gutiérrez, A. Coutrot, M. P. Da Silva, and P. Le Callet, “A dataset of head and eye movements for 360° videos,” in *Proceedings of the 9th ACM Multimedia Systems Conference on - MMSys ’18*, 2018, pp. 432–437, doi: 10.1145/3204949.3208139.

[49] Z. Zhang, Y. Xu, J. Yu, and S. Gao, “Saliency Detection in 360° Videos,” in *The European Conference on Computer Vision (ECCV)*, vol. 11211 LNCS, Springer Verlag, 2018, pp. 504–520.

[50] T. Xue, A. El Ali, T. Zhang, G. Ding, and P. Cesar, “CEAP-360VR: A Continuous Physiological and Behavioral Emotion Annotation Dataset for 360 VR Videos,” *IEEE Trans. Multimed.*, pp. 1–1, 2021, doi: 10.1109/TMM.2021.3124080.

[51] N. Sawhney, D. Balcom, and I. Smith, “HyperCafe,” in *Proceedings of the the seventh ACM*

conference on Hypertext - HYPERTEXT '96, 1996, pp. 1–10, doi: 10.1145/234828.234829.

[52] L. A. R. Neng and T. Chambel, "Get Around 360º Hypervideo Its Design and Evaluation," *Int. J. Ambient Comput. Intell.*, vol. 4, no. 4, pp. 40–57, 2012, doi: 10.4018/jaci.2012100103.

[53] M. Wijnants, K. Van Erum, P. Quax, and W. Lamotte, "Augmented ODV: Web-driven annotation and interactivity enhancement of 360 degree video in both 2D and 3D," in *Lecture Notes in Business Information Processing*, 2016, vol. 246, pp. 47–69, doi: 10.1007/978-3-319-30996-5_3.

[54] G. Noronha, C. Álvares, and T. Chambel, "Sharing and navigating 360° videos and maps in sight surfers," in *Proceedings of the 16th International Academic MindTrek Conference 2012: "Envisioning Future Media Environments", MindTrek 2012*, 2012, pp. 255–262, doi: 10.1145/2393132.2393189.

[55] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE Multimed.*, vol. 18, no. 4, pp. 62–67, Apr. 2011, doi: 10.1109/MMUL.2011.71.

[56] D. K. Krishnappa, D. Bhat, and M. Zink, "DASHing YouTube: An analysis of using DASH in YouTube video service," in *Proceedings - Conference on Local Computer Networks, LCN*, 2013, pp. 407–415, doi: 10.1109/LCN.2013.6761273.

[57] V. K. Adhikari *et al.*, "Unreeling netflix: Understanding and improving multi-CDN movie delivery," in *Proceedings - IEEE INFOCOM*, 2012, pp. 1620–1628, doi: 10.1109/INFCOM.2012.6195531.

[58] T. Fautier, "VR video ecosystem for live distribution," in *IBC 2016 Conference*, 2016, pp. 28 (13 .)-28 (13 .), doi: 10.1049/ibc.2016.0028.

[59] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim, "Mpeg Dash Srd," in *Proceedings of the 7th International Conference on Multimedia Systems - MMSys '16*, 2016, pp. 1–8, doi: 10.1145/2910017.2910606.

[60] M. Hosseini and V. Swaminathan, "Adaptive 360 VR Video Streaming: Divide and Conquer," in *2016 IEEE International Symposium on Multimedia (ISM)*, Dec. 2017, vol. 118, no. 1, pp. 107–110, doi: 10.1109/ism.2016.0028.

[61] M. Hosseini and V. Swaminathan, "Adaptive 360 VR Video Streaming Based on MPEG-DASH SRD," in *2016 IEEE International Symposium on Multimedia (ISM)*, Dec. 2016, pp. 407–408, doi: 10.1109/ISM.2016.0093.

[62] M. Hosseini, "View-aware tile-based adaptations in 360 virtual reality video streaming," 2017, doi: 10.1109/VR.2017.7892357.

[63] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," 2017, doi: 10.1109/ICC.2017.7996611.

[64] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "OpTile: Toward Optimal Tiling in 360-degree Video Streaming," in *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, 2017, pp. 708–716, doi: 10.1145/3123266.3123339.

[65] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012, doi: 10.1109/TCSVT.2012.2221191.

[66] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile Based HEVC Video for Head Mounted Displays," in *2016 IEEE International Symposium on Multimedia (ISM)*, Dec. 2016, pp. 399–400, doi: 10.1109/ISM.2016.0089.

[67] A. Zare, A. Aminlou, and M. M. Hannuksela, "Virtual reality content streaming: Viewport-dependent projection and tile-based techniques," 2018, doi: 10.1109/ICIP.2017.8296518.

[68] J. Van Der Hooft, M. T. Vega, S. Petrangeli, T. Wauters, and F. De Turck, "Tile-based adaptive streaming for virtual reality video," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 15, no. 4, pp. 1–24, Dec. 2019, doi: 10.1145/3362101.

[69] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular

networks," in *Proceedings of the 5th Workshop on All Things Cellular Operations, Applications and Challenges - ATC '16*, 2016, pp. 1–6, doi: 10.1145/2980055.2980056.

[70] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a Moving Target: Motion-Prediction-Based Transmission for 360-Degree Videos," in *2016 IEEE International Conference on Big Data, Big Data 2016*, Dec. 2016, pp. 1161–1170, doi: 10.1109/BigData.2016.7840720.

[71] C. L. Fan, J. Lee, W. C. Lo, C. Y. Huang, K. T. Chen, and C. H. Hsu, "Fixation prediction for 360° video streaming in head-mounted virtual reality," in *Proceedings of the 27th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2017*, Jun. 2017, vol. 6, pp. 67–72, doi: 10.1145/3083165.3083180.

[72] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[73] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000, doi: 10.1162/089976600300015015.

[74] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the International Joint Conference on Neural Networks*, 2000, vol. 3, pp. 189–194, doi: 10.1109/ijcnn.2000.861302.

[75] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, Oct. 2018, pp. 1190–1198, doi: 10.1145/3240508.3240669.

[76] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang, "Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2693–2708, Nov. 2019, doi: 10.1109/TPAMI.2018.2858783.

[77] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "CUB360: Exploiting Cross-Users Behaviors for Viewport Prediction in 360 Video Adaptive Streaming," in *Proceedings - IEEE International Conference on Multimedia and Expo*, Oct. 2018, vol. 2018-July, doi: 10.1109/ICME.2018.8486606.

[78] L. Xie, X. Zhang, and Z. Guo, "CLS," in *Proceedings of the 26th ACM international conference on Multimedia*, Oct. 2018, pp. 564–572, doi: 10.1145/3240508.3240556.

[79] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming," in *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, 2017, pp. 315–323, doi: 10.1145/3123266.3123291.

[80] Advanced Distributed Learning, "What is xAPI aka the Experience API or Tin Can API." https://xapi.com/overview/ (accessed Dec. 06, 2021).

[81] Advanced Distributed Learning, "The xAPI Specification," 2016. https://github.com/adlnet/xAPI-Spec/. (accessed Jul. 08, 2019).

[82] Advanced Distributed Learning, "The Registry," 2016. https://registry.tincanapi.com/ (accessed Jan. 26, 2017).

[83] A. Bakharia, K. Kitto, A. Pardo, D. Gašević, and S. Dawson, "Recipe for success: Lessons Learnt from Using xAPI within the Connected Learning Analytics Toolkit," in *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge - LAK '16*, 2016, pp. 378–382, doi: 10.1145/2883851.2883882.

[84] A. Berg, M. Scheffel, H. Drachsler, S. Ternier, and M. Specht, "Dutch Cooking with xAPI Recipes: The Good, the Bad, and the Consistent," in *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, Jul. 2016, pp. 234–236, doi: 10.1109/ICALT.2016.48.

[85] T. Rabelo, M. Lama, J. C. Vidal, and R. Amorim, "Comparative study of xAPI validation tools," in *2017 IEEE Frontiers in Education Conference (FIE)*, Oct. 2017, pp. 1–5, doi: 10.1109/FIE.2017.8190729.

[86] R. A. Sottilare, R. A. Long, and B. S. Goldberg, “Enhancing the Experience Application Program Interface (xAPI) to Improve Domain Competency Modeling for Adaptive Instruction,” in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale - L@S ’17*, 2017, pp. 265–268, doi: 10.1145/3051457.3054001.

[87] G. Rozinaj, M. Vanco, R. Vargic, I. Minarik, and A. Polakovic, “Augmented/Virtual Reality as a Tool of Self-Directed Learning,” in *International Conference on Systems, Signals, and Image Processing*, Jun. 2018, vol. 2018-June, pp. 1–5, doi: 10.1109/IWSSIP.2018.8439309.

[88] P. Sommerauer and O. Müller, “AUGMENTED REALITY FOR TEACHING AND LEARNING – A LITERATURE REVIEW ON THEORETICAL AND EMPIRICAL FOUNDATIONS,” 2018.

[89] C. Lecon and M. Herkersdorf, “Virtual Blended Learning virtual 3D worlds and their integration in teaching scenarios,” in *Proceedings of the 9th International Conference on Computer Science and Education, ICCCSE 2014*, Aug. 2014, pp. 153–158, doi: 10.1109/ICCSE.2014.6926446.

[90] P. Caserman, A. Garcia-Agundez, A. Gámez Zerban, and S. Göbel, “Cybersickness in current-generation virtual reality head-mounted displays: systematic review and outlook,” *Virtual Real.*, vol. 25, no. 4, pp. 1153–1170, Dec. 2021, doi: 10.1007/s10055-021-00513-6.

[91] J. C. Servotte *et al.*, “Virtual Reality Experience: Immersion, Sense of Presence, and Cybersickness,” *Clin. Simul. Nurs.*, vol. 38, pp. 35–43, Jan. 2020, doi: 10.1016/j.ecns.2019.09.006.

[92] S. Weech, S. Kenny, and M. Barnett-Cowan, “Presence and cybersickness in virtual reality are negatively related: A review,” *Frontiers in Psychology*, vol. 10, no. FEB. Frontiers Media S.A., p. 158, Feb. 04, 2019, doi: 10.3389/fpsyg.2019.00158.

[93] J. del Molino, “How analytics dashboards contribute to assessing the behavior of Virtual Reality users from xAPI tracking logs,” 2021.

[94] T. Bibiloni, A. Oliver, and J. del Molino, “Automatic collection of user behavior in 360° multimedia,” *Multimed. Tools Appl.*, vol. 77, no. 16, pp. 20597–20614, 2018, doi: 10.1007/s11042-017-5510-3.

[95] Three.js Contributors, “Three.js – JavaScript 3D Library.” https://threejs.org/ (accessed May 17, 2021).

[96] “The Registry.” https://registry.tincanapi.com/ (accessed Jul. 17, 2017).

[97] Advanced Distributed Learning, “Experience API (xAPI) - Video Vocabulary,” 2016. http://xapi.vocab.pub/datasets/video/ (accessed Jan. 26, 2017).

[98] A. Oliver, J. del Molino, M. E. Vidal, and T. Bibiloni, “360º Hypervideo: an interactive documentary around the refugee crisis in Greece,” in *Proceedings of the 6th Iberoamerican Conference on Applications and Usability of Interactive TV - jAUTI 2017*, 2017, pp. 162–171.

[99] A. Oliver, J. del Molino, and A. Bibiloni, “Automatic View Tracking in 360º Multimedia Using xAPI,” in *Communications in Computer and Information Science*, vol. 813, Springer, Cham, 2018, pp. 117–131.

[100] J. del Molino, T. Bibiloni, and A. Oliver, “Keys for successful 360° hypervideo design: A user study based on an xAPI analytics dashboard,” *Multimed. Tools Appl.*, vol. 79, no. 31–32, pp. 22771–22796, Aug. 2020, doi: 10.1007/s11042-020-09059-2.

[101] E. Williams, “Aviation Formulary V1.46.” https://www.edwilliams.org/avform.htm (accessed Nov. 25, 2020).

[102] A. Oliver, J. del Molino, M. Cañellas, A. Clar, and A. Bibiloni, “VR macintosh museum: Case study of a webVR application,” in *Advances in Intelligent Systems and Computing*, Apr. 2019, vol. 931, pp. 275–284, doi: 10.1007/978-3-030-16184-2_27.

[103] ITU-T Recommendation P.910, “Subjective video quality assessment methods for multimedia applications,” *Int. Telecommun. Union, Geneva*, 2009.

[104] C. Kumar, R. Menges, and S. Staab, “Eye-Controlled Interfaces for Multimedia Interaction,”

IEEE Multimed., vol. 23, no. 4, pp. 6–13, Oct. 2016, doi: 10.1109/MMUL.2016.52.

[105] J. Nielsen, T. Clemmensen, and C. Yssing, "Getting access to what goes on in people's heads?," in *Proceedings of the second Nordic conference on Human-computer interaction - NordiCHI '02*, 2002, p. 101, doi: 10.1145/572020.572033.

[106] J. Brooke, "SUS - A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996, doi: 10.1002/hbm.20701.

[107] A. Bangor, P. T. Kortum, and J. T. Miller, "An Empirical Evaluation of the System Usability Scale," *Int. J. Hum. Comput. Interact.*, vol. 24, no. March 2015, pp. 574–594, 2008, doi: 10.1080/10447310802205776.

[108] W. H. Press and S. A. Teukolsky, "Kolmogorov-Smirnov Test for Two-Dimensional Data," *Comput. Phys.*, vol. 2, no. 4, p. 74, Jun. 1988, doi: 10.1063/1.4822753.

[109] S. Rossi, F. De Simone, P. Frossard, and L. Toni, "Spherical Clustering of Users Navigating 360° Content," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, May 2019, vol. 2019-May, pp. 4020–4024, doi: 10.1109/ICASSP.2019.8683854.

[110] C. Bron and J. Kerbosch, "Algorithm 457: Finding All Cliques of an Undirected Graph [H]," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, Sep. 1973, doi: 10.1145/362342.362367.

[111] M. Orduna, P. Perez, C. Diaz, and N. Garcia, "Evaluating the Influence of the HMD, Usability, and Fatigue in 360VR Video Quality Assessments," in *Proceedings - 2020 IEEE Conference on Virtual Reality and 3D User Interfaces, VRW 2020*, Mar. 2020, pp. 683–684, doi: 10.1109/VRW50115.2020.00192.