Tampere University

ANTTI LUOTO

Logging and Analyzing 360-Degree Video Users

Tampere University Dissertations 958

Tampere University Dissertations 958

ANTTI LUOTO

Logging and Analyzing 360-Degree Video Users

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the Auditorium TB109
of the Tietotalo building, Korkeakoulunkatu 1, Tampere,
on March 1 2024, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information and Communication Sciences
Finland

Responsible supervisor and Custos	Professor Kari Systä Tampere University Finland	
Pre-examiners	Professor Philippe Palanque Institut de Recherche en Informatique de Toulouse France	Doctor Matti Pouke University of Oulu Finland
Opponent	Professor Petri Vuorimaa Aalto University Finland	

Cover design: Roihu Inc.

ISBN 978-952-03-3298-3 (print)
ISBN 978-952-03-3299-0 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-3299-0

HIILINEUTRAALI PAINOTUOTE
ClimateCalc CC-000025/FI
PunaMusta Printing

Carbon dioxide emissions from printing Tampere University dissertations have been compensated.

PunaMusta Oy – Yliopistopaino
Joensuu 2024

PREFACE

This doctoral thesis was written during years 2017–2023 while I was working at the Department of Pervasive Computing at Tampere University of Technology and at the Unit of Computing Sciences at Tampere University. However, my enlistment as a doctoral student already started in 2014.

A friend of mine once said that writing a doctoral thesis as such is not challenging. I am not sure if I agree but coping with the other work-related stuff, such as teaching, lack of resources, project changes, etc. at the same time can be laborious. Of course, producing an ambitious thesis entails more work than one that meets the minimum requirements.

Thanks to my superiors, supervisors, and colleagues for the opportunity to complete this thesis. Thanks to Professor Kari Systä, Professor Tommi Mikkonen, University Lecturer Timo Aaltonen, Professor Petri Ihantola, and Professor Hannu-Matti Järvinen. Thanks to all the research personnel on the projects Need for Speed, LiquidIoT, XThings, 360 Video Intelligence, and CityIoT. And thanks to the teaching personnel on the courses Database Basics, Design of Databases, Data-Intensive Programming, and in a bunch of other database courses.

Special thanks to my family – Onerva, Ilmari, Touko, and to my mother Riitta! Thanks to Jaakko for introducing the world of computer science to me!

ABSTRACT

User logging and analysis can be an important task in the domain of 360-degree video applications. This thesis studies what user logging in 360-degree video domain is, the reasons behind it, and how to do it efficiently.

The first contribution of the thesis is a definition of 360-degree video user logging, and a categorization for the different reasons for doing it. The study material used is scientific literature and company needs from five companies operating in the field. According to the thesis, the most popular reason for logging 360-degree video users is "viewport-adaptive streaming, transmission optimization, and saving bandwidth".

Despite the general benefits of log servers and databases, the related scientific implementations have not widely adopted them. Thus, the second contribution of the thesis is a reference architecture aimed at efficient distributed user logging and analysis. The reference architecture is used as a base for practical logging and analysis solutions. The solutions implemented are compared to those found in the related scientific literature. The comparison suggests that using publish-subscribe protocol in the logging of 360-degree videos users is a relatively novel approach, and the benefits of databases are also highlighted.

Different methods for user log analysis are studied with the emphasis on real-time visualizations. The third contribution of the thesis is real-time visualizations implemented using two approaches and the comparison of these to implementations made by other authors. The first approach is by drawing visualizations over 360-degree video during video playback and the second approach is by making traditional chart visualizations. The analysis needs of the five companies operating in the field are reflected against the visualizations implemented. A comparison to the state-of-the-art related research reveals that real-time 360-degree video user analysis visualizations are not common.

TIIVISTELMÄ

Käyttäjien lokittaminen ja analysointi voivat olla tärkeitä tehtäviä 360-asteen videoita hyödyntävien ohjelmistojen kehityksessä. Tässä väitöskirjassa tutkitaan, mitä tarkoittaa käyttäjien lokittaminen 360-asteen videoiden yhteydessä, miksi sitä tehdään ja miten sitä voidaan tehdä tehokkaasti.

Vätöskirjan ensimmäinen kontribuutio on määritellä mitä tarkoittaa 360-asteen videoiden käyttäjien lokittaminen ja luokitella syitä sen tekemiselle. Materiaalina käytetään tieteellistä kirjallisuutta ja viiden alalla toimivan yrityksen tarpeita. Väitöskirjan mukaan yleisin syy 360-asteen videoiden käyttäjien lokittamiselle on "katseluikkunaan mukautuva suoratoisto, lähetyksen optimointi ja kaistanleveyden optimointi".

Vaikka lokituspalvelimet ja -tietokannat ovat yleisesti hyödyllisiä ja tunnettuja ratkaisuja, niitä ei juurikaan käytetä 360-asteen videoiden käyttäjiin liittyvässä tutkimuksessa. Väitöskirjan toinen kontribuutio onkin viitearkkitehtuuri tehokasta ja hajautettua käyttäjien lokitusta ja analysointia varten. Tutkimuksessa tehtiin suunniteltuun viitearkkitehtuuriin perustuvia käytännön toteutuksia, joita verrataan tieteellisestä kirjallisuudesta löytyneisiin saman aihepiirin toteutuksiin. Viitearkkitehtuurin erityispiirteenä on julkaise-tilaa-protokollan käyttäminen 360-asteen videokatsojien lokittamisessa, mutta myös tietokantojen käyttämisen edut tuodaan esille.

Lisäksi väitöskirja kertoo erilaisista tavoista analysoida käyttäjälokeja painotuksen ollessa reaaliaikaisissa visualisoinneissa. Kolmas väitöskirjan kontribuutio on reaaliajassa toimivat käyttäjälokeihin perustuvat 360-asteen videon päälle piirretyt visualisoinnit sekä perinteiset kaaviot. Viiden alalla toimivan yrityksen tarpeita peilataan toteutettuihin visualisointehin. Vertailu alan viimeisimpään tutkimukseen paljastaa, että 360-asteen videoiden päälle piirrettävät reaaliaikaiset visualisaatiot eivät ole yleisiä.

CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

360VI	360 Video Intelligence, the project where most of the research work was done
ACID	Atomicity, Consistency, Isolation, Durability, the transactional properties of database management systems
API	Application Programming Interface
AR	Augmented Reality
CoAP	Constrained Application Protocol, a lightweight IoT protocol
EPB	Exploring and predicting user behavior
FIWARE	An open source framework aimed for accelerating the development of smart solutions
FPS	Frames per second
HMD	Head-Mounted Display
HTTP	Hypertext Transfer Protocol, a popular network protocol
IoT	Internet of Things
MOM	Message-oriented middleware
MQTT	Message Queuing Telemetry Transport, a lightweight IoT protocol
MR	Mixed Reality
NAT	Network Address Translation
NoSQL	Not only SQL, databases that are not relational by design
OPD	Offering a public dataset for reuse
QoS	Quality of Service

RDBMS	Relational Database Management System
SDK	Software Development Kit
SQL	Structured Query Language, a declarative language used for querying databases
TCP	Transmission Control Protocol, a core protocol in the internet suite
UDP	User Datagram Protocol, a core protocol in the internet suite
UI	User Interface
VAS	Viewport-adaptive streaming, transmission optimization, and saving bandwidth
VR	Virtual Reality
XR	Cross Reality

ORIGINAL PUBLICATIONS

Publication I	A. Luoto. "Towards Framework for Choosing 360-degree Video SDK". In: *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 1: SIGMAP, (ICETE 2017).* SciTePress, INSTICC, 2017, pp. 81–86. DOI: 10.5220/0006470600810086.
Publication II	A. Luoto. "Systematic Literature Review on User Logging in Virtual Reality". In: *Proceedings of the 22nd International Academic Mindtrek conference*. ACM, 2018, pp. 110–117. DOI: 10.1145/3275116.3275123.
Publication III	A. Luoto, P. Heino, and Y. You. "Lightweight Visualization and User Logging for Mobile 360-degree Videos". In: *2018 IEEE 11th Workshop on Software Engineering and Architectures for Real-time Interactive Systems (SEARIS).* IEEE, 2018, pp. 1–8. DOI: 10.1109/SEARIS44442.2018.9180230.
Publication IV	A. Luoto and K. Systä. "Fighting Network Restrictions of Request-Response Pattern with MQTT". In: *IET Software* 12.5 (2018), pp. 410–417. DOI: 10.1049/iet-sen.2017.0251.
Publication V	A. Luoto. "Log Analysis of 360-degree Video Users via MQTT". In: *Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis*. Association for Computing Machinery, 2019, pp. 130–137. DOI: 10.1145/3318236.3318248.
Publication VI	A. Luoto, K. Systä, O. Hylli, and V. Heikkilä. "Using IoT Platform for 360-degree Video User Logging and Analysis". In: *Proceedings of the 16th International Conference on Web Information*

Systems and Technologies (WEBIST 2020). SciTePress, INSTICC, 2020, pp. 41–50. DOI: 10.5220/0010015600410050.

Author's contribution

This text describes the work the author did to publish the aforementioned articles.

Publication I	The publication was written solely by the author of this thesis. The idea and the research questions are based on the discussions with Professor Petri Ihantola and project partners during 360 Video Intelligence project. The author of this thesis did all the practical research work.
Publication II	The publication was written solely by the author of this thesis. The idea and the research questions are based on the discussions with Professors Petri Ihantola and Kari Systä during the 360 Video Intelligence project. The author of this thesis did all the practical research work.
Publication III	The author of this thesis was the first author of the publication. The other authors helped by commenting and with some minor editing. The idea of the study came from the author of this thesis and it is based on the needs of the project partners of the 360 Video Intelligence project. The author of this thesis did most of the practical research work required for the technical implementation that is in the focus of the paper. However, the study uses software made by the other researchers of the 360 Video Intelligence project (for example, Timo Kalliomäki and Pietari Heino).
Publication IV	The author of this thesis was the first author of the publication. The other author provided comments and edited some sections. The journal article is an extended version of an earlier conference publication where the author was also the first author. The initial idea to use MQTT came from Professor Kari Systä. The technical implementation is heavily based on the work of the

other researchers of the project LiquidIoT, but the MQTT related refactoring was completely made by the author of this thesis.

Publication V	The publication was written solely by the author of this thesis. The idea to use MQTT for 360-degree video user logging and analysis was invented by the author of this thesis. Further, the technical implementation and other practical research work were done solely by the author of this thesis. However, since the publication continues from where Publication III ended, the software created by the other researchers of the 360 Video Intelligence project is still present.
Publication VI	The original idea to make a demonstration using an IoT-related data integration platform came from Professor Kari Systä, but it was an idea of the author of this thesis to integrate the 360-degree video user logging solution presented in Publications III and V into the platform. The author of this thesis was the first author of the publication. The other authors helped in the writing process by providing comments, and the technical implementation was built on top of an IoT-related data integration platform implemented by them.

1 INTRODUCTION

In recent decades, the volume of logged data has increased; new log sources and new IT environments have emerged [24]. These new log sources include Cross Reality (XR) related technologies, such as 360-degree video applications. User logging can be an important feature for multiple parties in 360-degree video domain [26, 84, 85, 145]. However, there is not much research available about the architectural needs of logging and analyzing 360-degree video watchers.

The main motivation of the research conducted comes from the need to develop software for 360-degree video domain. To develop and to change the content of the software, it is important to know what the users do while using that software. Some example applications of user analysis include improving video content by getting insight about the most popular parts of the video, superimposing advertisements over videos, transferring only the needed parts of video in high quality, replaying viewing sessions, etc. Some of the applications require an efficient architecture because of real-time analysis requirements. In addition, logging can be used to save the effort of developers who need to repeatedly test 360-degree video applications using special equipment. Applying logs to applications that are under development could partly remove the need for constant manual testing.

Efficient logging is important. One way of improving efficiency is to use a distributed architecture that supports simultaneous data collection from multiple users. That way, the need to collect and combine local log files can be omitted. With a proper solution, the log data is immediately and efficiently available for analysis. Surprisingly, many 360-degree video user logging related studies use an architecture that supports only local log collection.

1.1 Research Questions

The thesis answers the following research questions:

- RQ1: What is 360-degree video user logging and why do it?
- RQ2: How to support 360-degree video user logging via development of efficient architecture?
- RQ3: How to efficiently visualize 360-degree video user logs?

RQ1 is a relevant question because there is scientific literature that uses 360-degree video user logging related terminology but the terminology is not well defined and the author is not aware of a proper definition for 360-degree video user logging. The latter part of the question makes the question slightly more practical and emphasizes the importance of logging by directing the answer towards the concrete reasons for logging that can be found in the literature and in real-life company cases.

Logging is not possible without technology so RQ2 is directed towards the technical solutions of collecting 360-degree video user logs. 360-degree video user logging has its own characteristics and logs can be collected in many ways, but the author concentrates on efficient options. Efficient logging means, in this case, a log collection architecture that supports, for example, simultaneous usage, low latency, high performance, re-use, is generally easy to implement, and stores the data so as to be available for both real-time and non-real-time analysis.

Since logging alone is rarely highly beneficial and many of the advantages of logging are gained via log analysis, RQ3 is an important part of this research. It can be seen as a continuation of RQ1 since the answer of RQ1 lists reasons for analyzing logs, from which the answer to RQ3 continues by studying the visual presentation of 360-degree video user logs. The question is further scoped towards efficiency which links the question to the aspects of RQ2.

1.2 Research Overview

This thesis is based on the work accomplished on the research projects LiquidIoT (2016), on deploying software to IoT devices, 360 Video Intelligence (360VI) (2017–2019), that concentrated on software development related to 360-degree video applications, and CityIoT (2020), that was about IoT in the smart city context, but the emphasis is clearly on 360VI. Additional details of the projects can be seen in Table 1.1. The author's task in 360VI was to develop applications for data produced by 360-degree video analysis algorithms. Such data can be, for example, object de-

Table 1.1 Projects on which the thesis was done.

Project	Main Idea	Funding
LiquidIoT[1]	To develop concepts, technologies, and prototypes for programmable IoT devices.	Nokia Corporation
360VI	To develop solutions to support 360-degree video applications.	Business Finland
CityIoT[2]	To develop a solution for integrating data from separate silos to support digitalization in a smart city context.	European Union (European Regional Development Fund)

tection analysis results. This resulted in Publication I (PI) on the essential needs of software development for 360-degree video applications. According to the results, there was a need to analyze the behavior of users of 360-degree video applications while they watch 360-degree videos. The data for the behavioral analysis could be collected via logging. Then, to identify the state-of-the-art in logging research, Publication II (PII) studied the role of logging in scientific Virtual Reality (VR) literature using systematic literature review as a research method. One finding was that user logging related technical details are rarely explained in VR studies.

If Publications I and II provide background and justification for the thesis, the rest of the publications compose the technical part. First, in Publication III (PIII), the author developed a 360-degree video user logging system that uses Hypertext Transfer Protocol (HTTP) as a log transfer protocol, stores logs in a database, and can draw log visualizations over the video. Due to disadvantages of HTTP, the author next explored other protocols. After considering the options and getting some hands-on experience in using Message Queuing Telemetry Transport (MQTT), it started to seem like a promising solution. The selection of MQTT, often used in the IoT context, is supported by Publication IV (PIV) comparing MQTT and HTTP. Publication V (PV) presented an improved version of the logging system of PIII by replacing HTTP with MQTT and reporting user log analysis, such as calculating the most popular areas of the video. The problem of the logging and analysis system was that the implementation required custom code which had a negative effect on the

[1]https://webpages.tuni.fi/livelykernel/LiquidIoT.html
[2]https://www.cityiot.fi/english

generality. Finally, Publication VI (PVI) aimed to solve the problem by presenting an alternative solution for logging and analyzing 360-degree video users using a more general approach where the architecture is based on an IoT related data integration platform. Design Science Research methodology was used in Publications III–VI.

The research questions formulated for the introductory part of the thesis (meaning Chapters 1–10 not including Publications I–VI) could not be directly answered only with the included publications; answering required further research. The introduction is not only a summary about the publications but also includes a more profound and a far-reaching discussion to answer the research questions. In addition, the introduction has a role in gathering and summarizing the information of the publications to compose an understandable whole.

The answers to the research questions have been structured in the following way. The answers related to RQ1 provided by PII have been complemented with further research reported in Chapter 4. RQ1 is answered by combining a literature review with the experiences from the company cases of 360VI. Chapter 5 answers RQ2 by presenting a general reference architecture for 360-degree video user logging and analysis whereas publications PIII–PVI concentrate more on the technical details of the solutions implemented. Design Science Research methodology was used to answer RQ2. RQ3 is not directly answered in any of the publications included; it is answered in Chapter 6, for example, by summarizing the visualizations implemented for the publications included – PIII, PV and PVI. Design Science Research methodology was also used in answering RQ3.

1.3 Structure of the Thesis

The rest of the thesis is structured as follows. Chapter 2 starts with the basics of 360-degree videos and 360-degree video application development. Further, it introduces user logging, user logging architectures, and 360-degree video user logging systems as a distributed system. The chapter also introduces some related architectural topics, such as architectural styles and log storage techniques. Analysis and usage of log data are discussed, for example, by describing different analysis methods. The chapter finally summarizes the 360VI project, and points out the gap in related research.

Chapter 3 presents the research approach and methods adopted. It introduces Design Science Research methodology and presents a systematic literature review, etc., in addition to summarizing the research process and pointing out which research questions each publication has a part in answering.

Chapter 4 is the first chapter addressing the main results of the thesis. It studies the scientific literature and company cases of the 360VI project to find out a definition for user logging in 360-degree video domain and the reasons for doing so (RQ1).

Chapter 5 concentrates on logging architectures by presenting a reference architecture based on the requirements derived from the reasons for logging (RQ2). The chapter lists logging solutions found in the related scientific literature and compares the architectural solutions of the publications to these.

Chapter 6 reasons and presents the visualizations implemented during the research (RQ3). The implemented visualizations are compared to visualizations found in the related scientific literature.

Chapter 7 discusses related work. The chosen aspects include, for example, VR software development, 360-degree video user logging and analysis, and user logging architectures.

Chapter 8 summarizes the results in light of the research questions. It also discusses the limitations of the thesis and contemplates ideas for future work.

Chapter 9 concludes the thesis by summarizing the contributions made. Thereafter follow the publications forming part of the dissertation in their original format.

2 BACKGROUND

This chapter defines the key concepts and forms the theoretical background of the thesis. 360-degree video is the central topic that motivates the connection of the other topics, such as software development, user logging, distributed systems, logging architectures, and databases. This chapter also introduces the research project 360VI, where most of the research work was done, and it points out the gap in the related research.

2.1 360-Degree Videos

360-degree videos have recently gained popularity [1, 116, 117]. These are omnidirectional videos, where the viewer is located inside a 360-degree sphere with freedom to choose the viewport direction, as opposed to traditional videos, where the viewport is static and shows only one direction. The majority of 360-degree videos are typically viewed on computers, handheld phones, and head-mounted displays (HMD) [116]. Smolic and Kauff [131] define 360-degree videos as an extension of the planar 2D image plane to a spherical (or cylindrical) image plane, where video is captured at a certain viewpoint in multiple directions. According to Corbillon [25], 360-degree video is captured with an omnidirectional camera.

Using an HMD offers an immersive way of watching 360-degree videos. HMD is a wearable display device consisting of an optical system in a helmet with displays located in front of the user's eyes, creating an illusion of depth [128]. The sensors of the HMD sense the head movements of the user, which will lead to corresponding updates in the direction of the viewport shown. Modern smart phones offer a relatively cheap way to create an HMD since a smart phone can be a part of an HMD acting as the screen in front of the user's eyes. Such products include Samsung Gear VR and Google Cardboard. These low-cost products partly explain the reason of the recent popularity of 360-degree videos [20].

HMDs are not the only way to watch 360-degree videos. For example, many websites include 360-degree videos where the user can direct the viewport using traditional desktop input devices, such as a mouse. Similarly, handheld smart phone applications may have a 360-degree video player where the phone orientation determines the viewport seen. Handheld applications make it easy to view 360-degree videos with mobile devices with no need for often cumbersome HMD.

In this thesis, 360-degree video is sometimes discussed in the context of virtual reality (VR). The reasons for this are that 360-degree video and VR domains are often discussed in the same context [52, 26, 63, 145, 9, 124], there seems to be more literature available on VR domain, and some of the VR literature is applicable to 360-degree video domain. One example of the shared context of 360-degree videos and VR is that many 360-degree video publications explicitly discuss the use of VR technology. Another example is that Google VR Software Development Kit (SDK) includes tools for both VR and 360-degree video application development. VR can be defined as "a real or simulated environment in which a perceiver experiences telepresence" [134]. Like 360-degree videos, VR often requires special hardware, such as HMDs, including body tracking sensors and features so that the VR environment can respond to user activities, such as head movement. The main difference between VR and 360-degree videos is that, VR content is often a computer-generated 3D world, where users have more freedom to move and perform other actions, whereas 360-degree videos are less flexible video material. By using the taxonomy of Milgram and Kishino [91] one can think that world is unmodeled and there is a lack of ability to move within the world in 360-degree videos. 360-degree video allows three degrees of freedom (3DOF) (yaw, pitch, roll), whereas more immersive VR allows six degrees of freedom (6DOF) (yaw, pitch, roll, x, y, z). However, there are also specialized cases of simulated 6DOF in 360-degree videos [65].

Another related concept is augmented reality (AR). AR is something that combines real and virtual, is interactive in real time, and is registered in three dimensions [6]. AR has recently become popular due to mobile phone games and other applications that use cameras to augment real world footage with interactive graphics in real time. AR applications use video footage but usually not 360-degree videos. Similarly to VR, AR often allows more degrees of freedom than 360-degree videos.

2.2 360-Degree Video Application Development

By 360-degree video applications, the author refers to applications having a 360-degree video player that can play 360-degree videos in spherical mode. The applications may be accompanied by other features, such as visualizations over the video. An example application could be an educational mobile phone application that can be used for watching 360-degree videos about driving a car so that the watcher is situated inside the 360-degree sphere as if he or she were actually in the traffic situation. Then it would be possible to give feedback to the user based on his or her actions by visually emphasizing the objects that should have been noticed while watching the video. Another example application could be a media library where the user can choose 360-degree videos from the library menu while simultaneously watching videos. However, the application may also be simpler. It may be just an application with a simple 360-degree video player and possibly some traditional text placed next to the player.

Developing 360-degree video applications benefits from 360-degree video SDKs since they often offer an easy way to add a 360-degree video player and related functionality to applications. SDK can be defined as "a set of development tools that allows a developer to create applications for a certain software package and hardware platform. SDKs differ in terms of their programming languages, their libraries and API support" [113]. Thus, 360-degree video SDK is an SDK that helps developing 360-degree video applications. A 360-degree video SDK can, for example, offer an API that helps create a video player that can adapt to view orientation while playing spherical videos. Another useful API feature is accessing view orientation data that enables logging view orientations of users. The 360-degree video user logs allow user analysis as suggested in Publication PI.

A few 360-degree video SDKs were experimented with during the research. Google VR SDK[1] was used in Publications PIII, PV, and PVI to implement a mobile application with a 360-degree video player. The SDK offers a class (called VrVideoView) for adding a 360-degree video player to an Android application. The video player has basic controls, such as play, pause, volume controls, skip, full screen mode, etc. It supports both handheld and HMD usage. The advantages of Google VR SDK include light weight and a relatively gentle learning curve for programming. On

[1]https://developers.google.com/vr

the other hand, Google VR SDK is a relatively low-level tool that does not offer a wide-range of features. Krpano[2] offers another approach for making a 360-degree video player. It is a small and flexible JavaScript library for adding 360-degree videos to web pages [78]. There are multiple plugins available for extending the features of Krpano. Unity[3] is a popular 3D game development platform. The default approach for making a 360-degree video player in Unity is to set a 360-degree video as a texture for a spherical shape inside of which the video watcher is placed. Unity is a versatile tool that allows advanced features to be implemented relatively easily, but it can be rather complex. The author experimented with Krpano and Unity 3D in Publication PIII.

2.3 User Logging

One software logging study used dictionary definitions to define "log" [31]. It has been defined as a record of performance, events, or day-to-day activities [34]. Another definition is as a regular or systematic record of incidents or observations [35]. Those definitions work well in the software engineering domain. Logging is an important software engineering task since logs can be used for various kinds of analysis, such as application, network, and server monitoring, optimization, error debugging, vulnerability assessment, etc. [24].

On a technical level, a log message is a computer-generated message containing information about the events of a specific system. Log messages need to be stored somehow. A simple way is to write log messages into a log file that can be a simple text file, but this may not suffice for all purposes. A log message typically contains a timestamp, data source, and actual data [24]. Logging often lacks formal practices and systematic design [123].

Logging in general is an important software engineering practice, but there are only few definitions available for *user* logging. Nielsen [103] mentions that "Normally, logging is used as a way to collect information about field use of a system after release, but logging can also be used as a supplementary method during user testing to collect more detailed data." One attempt to define user logging is as follows: "The recording of a set of user actions, often by means of automatic recognition of events

[2]https://krpano.com/home/
[3]https://unity.com/

related to user activity, for example, touching a sensor or moving into a controlled area." [10].

Common log sources have been categorized to include operating systems, network daemons, network infrastructure, and applications [24]. While the term *user logging* is not actually used, it is noted that application logs may include user activity data [24]. User (activity) logging can therefore be considered a common logging action, which is also supported by the use of the term *logging* in the human-computer interaction literature [103], despite the original meaning of logging as a debugging method [137].

It is good to note the following remark by Chuvakin et al. [24]: "Many of the terms used in logging, log analysis, and log management – including these very terms we just used – have vague, misleading, or multiple meanings." Further, the concept of logging has expanded from tracing statements assisting in troubleshooting and debugging to a broader concept of telemetry, where the production environment events continuously stream data to the monitoring dashboards of software engineers [11].

As a summary about the terminology, while the dictionary definitions for logging fit well into software engineering domain, and there are some definitions for *user* logging, logging related terminology in general is vague. The term logging has evolved from debugging and troubleshooting methods to have a broader meaning.

Software users have generally been logged in different domains, and for various reasons. For example, in human-computer interaction work, logging can be used to obtain statistics about the use of a system after the release or during user testing [62]. Runtime traces have been analyzed for improving architecture, performance, design, and usability in agile software development [125]. Many authors discuss tracking and analyzing users in web [19, 22, 92]. Logging techniques include manual implementation, automatic instrumentation, aspect-oriented programming, alternative library implementation, and monitoring the execution environment [138].

This thesis concentrates on a sub-task of user logging. The main idea is to the log view orientations of 360-degree video viewers by using the sensors of mobile devices as a data source. The idea is scoped by the needs of the companies involved in project 360VI so that the aim is to collect logs for user analysis for various applications where view orientations are important. The primary aim does not include logging user actions, such as pressing pause or play, although logging such actions could be included relatively easily.

2.4 Implementation and Architectural Issues

This section introduces implementation and architectural design issues of a 360-degree video user logging and analysis system. The main aspects are distributed systems, architectural styles, general logging architectures, data storage, and data processing.

2.4.1 360-Degree Video User Logging System as a Distributed System

A computer system composed of many networked geographically dispersed computers is often called a distributed system [133]. There are four goals in making distributed systems: (1) sharing common resources, (2) hiding implementation details, (3) openness, and (4) scalability [133]. In the context of this thesis, the first goal is the most important because there is a need to share a dataset in some common reasons for analyzing 360-degree video users. The data is shared so that it is possible to produce analysis results based on the shared dataset collected from multiple users. A common way to share such a dataset is to make it available via log server with database. There is also a need to hide implementation details because end-users (for example, video viewers in the context of this thesis) should not generally need to worry about technical implementation details, such as the address of a logging server. The third goal is relatively important – there is a need for openness because development is often easier when components can be easily used by, or integrated into other systems. For example, the ability to easily change a database may be important if the needs of logging or analysis change. Finally, it is advisable to prepare for scalability needs because the number of 360-degree users and related log data is expected to grow in the future. For example, a ready-made general and scalable solution, such as a data integration platform for IoT, may be a better solution from the scalability perspective than a single database.

The components of a distributed system require communication between each other to achieve common goals. Different communication types include connection-oriented, connectionless, synchronous, and asynchronous communication. In connection-oriented communication a logical channel is established between a sender and a receiver, whereas in connectionless communication services do not build a connection before the communication starts [90, 50]. In general, connectionless communication is considered more lightweight and faster but not as reliable [36]. In a distributed 360-degree video user logging system, a connection-oriented communication means, for example, that an HMD sending the log data with Transmission Control Protocol (TCP) first needs to make a connection with a log server after which it is possible to send log data, and the connection eventually needs to be terminated. However, a connectionless solution implemented, for example, with User Datagram Protocol (UDP) could be useful for many 360-degree video user logging use cases if it is not important that every log message is received by the log server. The advantages of connection-oriented communication include less packet loss and correct order of packets, and the advantages of connectionless communication include performance improvements and resource savings [36].

In synchronous communication the sender stops execution after sending a message and waits for a response that may lead to inefficient communication [69]. In asynchronous communication the sender continues to execute after sending a message [69]. In a distributed 360-degree video user logging system, asynchronous communication would allow the video player application to send log messages with no need to wait for responses, whereas with synchronous communication the video player application would need to wait for responses that could, in some cases, prevent the video player application from proceeding with the playback. In addition, responding puts more load on the log server and on the network. Zepeda and Nuñez [148] mention that there is not much research on the efficiency between asynchronous and synchronous communication (presumedly in the web context) although we can intuitively assume that asynchronous is more effective. The author of this thesis shares their view with the experience that finding general and explicit performance comparisons of synchronous and asynchronous network protocols is relatively challenging. While it is clear that there are performance comparisons between synchronous and asynchronous network protocols available (for example [13]), the studies do not necessarily discuss synchronicity or provide generalizable conclusions on the efficiency [132]. The study by Zepeda and Nuñez [148] in web context suggests that asynchronous communication requires significantly less network usage. In any case, synchronous communication requires coupling between sender and receiver that might be considered inefficient [69].

According to Fielding [46]: "The client-server style is the most frequently encountered of the architectural styles for network-based applications. A server component, offering a set of services, listens for requests upon those services. A client component, desiring that a service be performed, sends a request to the server via a connector. The server either rejects or performs the request and sends a response back to the client." Separated client and server enable development work on separated platforms and keep the components reusable [87]. HTTP is an example protocol that is used with client-server architecture. It is a request-response network protocol where a client first sends a request to a server with a request method, URI, protocol version, request modifiers, client information, and possible body content, and the server responds with a structured message header containing protocol version, server information, meta-information, etc. [45]. HTTP is primarily a web messaging protocol whose first standard was published in 1997 [97]. It has standard methods, such as GET, POST, PUT, and DELETE, many return codes, and can be considered complex since the specification is more than 160 pages long [80]. The protocol overhead of HTTP causes serious problems when transferring a large number of tiny packets [146]. However, it is the most widely used and most readily available protocol [80]. Request-response communication is traditionally synchronous [130], which makes it inefficient because the sender needs to wait for a response before continuing execution. Thus, different techniques, such as HTTP long polling, HTTP callbacks, and WebSocket, have been developed to by-pass the limitations and enable asynchronous style. However, request-response was not originally designed for such a functionality.

Message-oriented middleware (MOM) is an architectural style where a middleware serves as a connecting entity between components sending messages to each other [133]. It helps to implement loose coupling by providing support for asynchronous communication [133]. Publish-subscribe is a messaging pattern that can be used with MOM [133]. It is a messaging pattern where senders publish messages to subscribers who receive a certain type of messages. Publishers do not send messages directly to subscribers; subscribers announce their interest in a certain type of messages, which are then delivered to them. In broker-based publish-subscribe, a broker stands in the middle and redirects the published messages to interested subscribers. This enables one-to-many communication, meaning that a broker delivers a message from one sender to many subscribers. Implementing similar functionality with a request-response protocol may be inefficient [87]. In a 360-degree video user logging and analysis system, one-to-many communication can be used, for example, by a log server to send analysis results to interested end-user applications. It can also be used by log sources to send log data not only to a log server but also to additional subscribers interested in logs.

While the client-server has a request-response form, publish-subscribe is more a push model: there is neither request nor polling [87]. Publish-subscribe has been claimed to be an ideal pattern for communicating between distributed application nodes in real-time systems [87]. It is suitable for distributed real-time systems since events are delivered to subscribers immediately after an event has occured, it is asynchronous which frees the data sender from waiting for an acknowledgement, it eliminates the need for two messages for each request-reply, and publishers need to send only one event to a broker, which is then multiplied by delivering it to many subscribers (one-to-many communication) [87, 104, 66]. Publish-subscribe provides full decoupling in time, space, and synchronization [41]. It is important because decoupling can improve efficiency [69]. Publish-subscribe is an important communication pattern in modern logging architectures [77, 23].

Message Queuing Telemetry Transport (MQTT) is an example of a broker-based publish-subscribe protocol [8]. It is designed to be lightweight, simple, and for constrained devices in unreliable networks. Light weight partly comes from the small transport overhead and minimized protocol exchanges [8]. MQTT is often proposed for communication from sensors to (edge) gateways in IoT domain. However, this thesis discusses another type of usage, where mobile phone applications receiving data from mobile phone sensors are used as data collection sources. Compared to HTTP, the light weight of MQTT is beneficial when trying to conserve resources [80, 102, 97]. Several studies compare the performance and latency between HTTP and MQTT suggesting that in many cases MQTT performs significantly better [70, 126, 59, 71]. However, in a stress test MQTT was not a clear winner [143]. The differences in the performance are caused by the different characteristics of the protocols, such as communication pattern, congestion control, communication complexity, signaling traffic, connection establishment speed, session management, message overhead, and payload format [71]. The light weight of MQTT is also supported by Publication PIV, which compared resource usage of HTTP and MQTT. The chosen Quality of Service (QoS) level has an effect on the latency and network usage of MQTT [2, 82, 141]. In addition, several authors mention the benefits of MQTT when communicating beyond Network Address

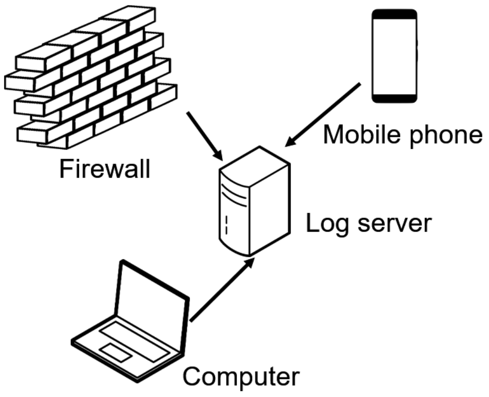

Figure 2.1 Logical layout of different devices sending log messages to a log server. Adapted from Chuvakin et al. [24].

Translation (NAT) [142, 15, 48, 140]. Relatively weak interoperability can be seen as a disadvantage of MQTT (version 3.1.1) since publish-subscribe does not cover all cases of communication use [97].

2.4.2 Logging Architectures

Architecture is a high-level specification of a system, its parts, the relationships between the parts, and their description. A basic logging architecture is the distributed log server architecture presented in Figure 2.1. The idea is that different kinds of log sources, firewall among them, send their logs to a log server via network [24]. The log server architecture may suffice for a local organization [24].

However, for a more decentralized organization, there may be a need for a set of distributed servers. Databases can be attached to the configuration, for example, to enable efficient reporting, analysis, and backups [24]. Figure 2.2 depicts a set of distributed log servers with databases. However, while a set of distributed log servers is not essential in this thesis, the idea of using databases for log collection is essential. The solutions developed for this thesis are somewhere between Figures 2.1 and 2.2 as there is a single log server with a database. Another way to see a logging architecture is as tiers composed of a log generation tier, a log collection and storage tier, and a log monitoring tier [4].

Logging sources also affect the architecture of a logging system. Chuvakin et al. [24] categorize logging sources into two general categories: push-based and pull-

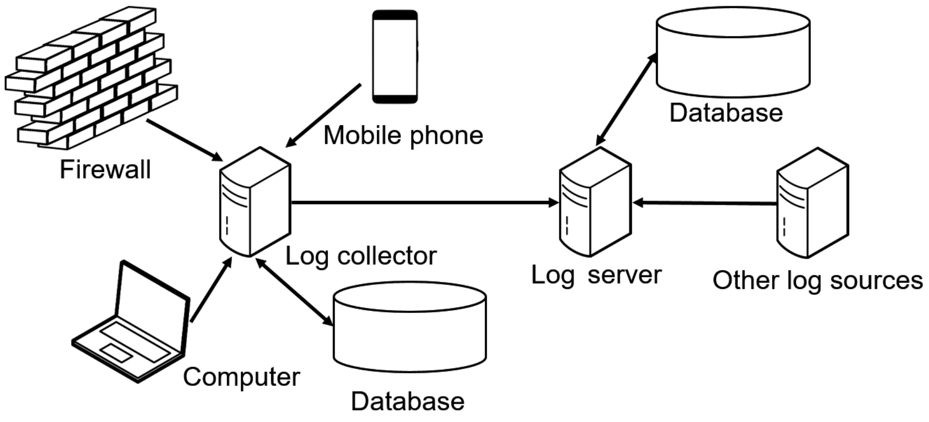

Figure 2.2 A set of distributed log servers and databases in a logging architecture. Adapted from Chuvakin et al. [24].

based. With push-based log sources, the log is sent over a network to a log collector [24]. A pull-based log source most often means a client-server solution where log can be retrieved from the source [24]. In addition, publish-subscribe is considered a useful log delivery model [23]. Some popular log transport mechanisms include syslog, HTTP, and regular file transfer, such as FTPS and SCP [24]. Common content encoding for log storage includes XML, text file, CSV, binary, and relational format used by relational databases [24].

2.4.3 Databases and SQL

Logged data needs to be stored. Storage media can be roughly categorized into four types: local file system, database, distributed file system, and cloud-based storage. All of these have their own characteristics, but the distributed nature of logging multiple users requires networked data storage. Distributed file system allows distributed usage, but the disadvantages include complexity, and performance overhead. Using cloud storage obviates the necessity to set up and maintain a distributed file system, but writing to storage especially requires more time [24]. Thus, databases seem to be the solution with the greatest potential for distributed 360-degree video user logging with real-time analysis requirements. There are various kinds of databases, such as Relational Database Management Systems (RDBMS), NoSQL databases, and NewSQL databases.

RDBMS is a data storage and management software based on a relational data model that works best with structured data [81]. Data in RDBMSs is managed via a high-level language called SQL that helps in writing database queries [81]. Non-relational databases, also known as NoSQL databases, manage unstructured data. They often scale better and may offer a better performance compared to relational databases [81]. The disadvantages of NoSQL databases include weakened reliability, compromised consistency, developer inexperience, and limited customer support [81]. A subgroup of NoSQL databases are time series databases that have recently gained popularity [32]. Time series databases record events happening in time storing each value change, which allows analyzing change in the past, monitoring current change, and predicting change in the future [120]. In addition, there are NewSQL databases that aim to enable the scalability of NoSQL while offering the benefits of RDBMSs.

Althought traditional RDBMSs have their disadvantages and competing technologies have emerged, SQL remains a popular tool that has been integrated into modern platforms that are not RDBMSs by design. One example is a modern data processing platform called Apache Spark, which has an SQL module as one of its core components [5, 29]. Several reasons make SQL an attractive tool for data analysis. SQL has a common set of functions needed for data analysis and allows relatively complex queries. For example, SQL makes it easy to compute multiple aggregates concisely [5]. It is a mature technology with strong community support [122], use of popular standards is an important principle in data processing [14], and using a standard query language improves portability [100]. Further, developers often prefer to write declarative queries, and SQL can be extended with user defined functions [5]. SQL often comes with a query optimizer. One example is the optimizer of the SQL module of Apache Spark [5]. Not surprisingly, SQL is an attractive option for 360-degree video user log analysis. It provides a standard and a developer-friendly way to start analyzing data. Some of the weak performance reputation of SQL is related to poor database design, not to SQL itself [135].

2.5 360-Degree Video User Log Analysis

One purpose of logging is to obtain meaningful results via log analysis. There are methods to help with analysis since understanding masses of logs is challenging. A study lists log analysis methods: manual inspection, usage of command line tools, ignoring irrelevant messages, replay, aggregation, approximation, statistics, machine learning, combining heterogeneous logs, correlation of different types of logs, prediction, profiling, and visualizations [106]. Another study states that statistics, correlations, and visualizations are used for analyzing behavioral patterns from usage logs [55]. 360-degree video user log analysis and general user log analysis share common characteristics. Naturally, the suitability of the analysis method depends on the needs. For example, manual inspection of logs is not useful if there is a need for real-time visualizations.

2.5.1 Tile-Based Analysis

A specific method of analyzing users viewing spherical (or other wide) video content is to divide the content into tiles [117]. An example of a tiled sphere is presented in Figure 2.3. Tiling can be used as an approximation method since it may suffice to know the tile instead of the exact coordinates. The motivation for such an approximation is to make analysis simpler; for example, it may be easier to refer to a certain area of the video as Tile1 than "the tile that covers the area of 0–30 degrees of yaw and 0–30 degrees of pitch". In addition, it may be simpler, for example, to count the number of labeled tiles (how many times Tile1 appears in a log) instead of making a comparison for each log record to see whether they are inside a certain outlined area (0–30 degrees of yaw and 0–30 degrees of pitch). Thus, tiling can be also seen as a preprocessing method that helps with the analysis. Tile-based analysis can also support the tiling-based presentation of the analysis results and other use cases where tiling is used. Tiling is also used in video transfer optimization to stream in high quality only those tiles that are in the current field of view.

2.5.2 Real-Time Analysis

Real-time computation may be either hard or soft. Traditionally hard real time means that computations must meet their deadlines while soft real time means the computation does not have as strict deadlines [68]. In other words, for soft real-time computation it is more acceptable to miss deadlines. From that perspective, this thesis is about soft real time since the 360-degree video logging and analysis systems discussed can perform correctly enough even if for some reason the computation is occasionally late. Real-time analysis in the context of this thesis means that log

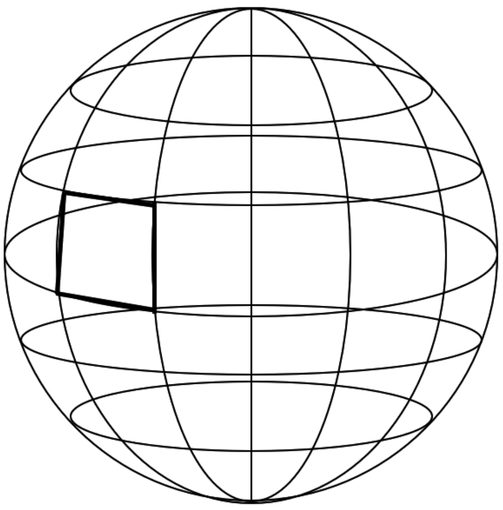

Figure 2.3 Division of a sphere into tiles. The borders of one tile have been emphasized.

analysis results computed from the collected logs can be used immediately in various applications so that the delay between sending a log record and receiving log analysis results is not distracting.

In the 360-degree video context of this thesis, the need for real-time analysis may be continuous. This is because the video content changes continuously and the actions of the users need to be analyzed accordingly. For example, when analyzing viewers of a live 360-degree video sports event to determine which parts of the video will be streamed in high quality, the time to perform an action may need to be almost immediate so that the viewing experience does not suffer, and such decisions are made continuously. Another real-time example is usage of learning environments, where it may be important to give users continuous feedback during a view session, for example, when teaching traffic safety. This thesis studies company cases related to the examples mentioned.

The architecture for the centralized analysis of distributed 360-degree video users must support real-time features by using efficient technologies, such as lightweight protocols, asynchronous communication, and optimized analysis tools. In addition, preprocessing of log data, for example, via tiling, may be important when implementing real-time features.

2.5.3 Statistical Analysis

One characteristic of 360-degree video user logs is that view orientations are often in a numerical format, which obviates the conversion from text to numerical format required by many statistical techniques. While some 360-degree video user analysis is manually feasible, it can be impossible in real-time cases. Luckily, statistical analysis can be automated with programmatic methods. It is difficult to fully separate different analysis methods since visual, statistical, and programmatic methods may be combined and overlapping so that visualizations are often based on statistical calculations while visualizations and statistics can be produced programmatically. In the related literature, is not always clear if statistical methods have been automated.

Statistical methods have been used in 360-degree video user analysis example cases. Corbillon et al. [26] calculated statistics based on angular distances and regions of interest from a 360-degree video user logging dataset. Wu et al. [145] calculated statistics of the angular speed of participants turning their heads. Duanmu et al. [38] calculated various statistics related to maximum angular distances, angular velocity of view center motion, and viewing velocity. Fremerey et al. [49] created graphs showing the percentage of users and time used for exploring certain areas of the video. Nasrabadi et al. [99] also calculated various kinds of statistics.

When statistical methods are used programmatically, it is possible to accomplish tasks like pattern recognition, prediction, etc. with machine learning, which combines data, algorithms and statistical methods [24]. For example, ML has been suggested for analyzing the use of pyramidally encoded 360-degree videos [79]. Another example is that viewers' head movements can be predicted with over 90% accuracy with simple methods, such as linear regression [117]. SQL is one programmatic implementation technique for statistical analysis. It can be used even for advanced analysis, such as linear regression [42] and k-means clustering [111].

A special characteristic in 360-degree video domain is spherical content. Rossi et al. [124] proposed a spherical clustering method for users navigating in 360-degree content. The idea of the method is to identify more meaningful user clusters for spherical content. Some studies have used the clustering method developed by Rossi et al. [84, 99].

There may be a separate analysis component for programmatic analysis in a log collection and analysis architecture. For example, a centralized networked analysis component can enable analysis based on logs from distributed users.

2.6 Visualization and Other Usage of 360-degree Video User Logs

In addition to log analysis, another purpose of logging is to use the logs in various applications. Some of the applications visualize the log data but there are also other types applications, such as changing the content of the video according to logs. The usage and presentation of 360-degree video user logs can be based on programmatic log analysis.

2.6.1 Visualization of 360-Degree Video User Logs

Visualizations can be categorized into visualizations drawn over the video and visualizations that are meaningful even without seeing the video at the same time. The visualizations drawn over the video require the presence of the video to be meaningful. Visualizations can be produced in different parts of the logging architecture. Visualizations over the video are drawn by the same components that show the video whereas visualizations not drawn over the video can be produced by some other component that does not have a video player. For example, there can be a separate admin dashboard showing chart visualizations with no need for video playback. Furthermore, the programmatic analysis on which the visualizations are based can be executed in a separate analysis component hosted by a separate service of the logging architecture.

Scan path is a linearly connected dot sequence presenting a change in the user's view orientation in time [39]. However, it is not a good technique for visualizing log data from multiple users since markers and labels are easily confused if a lot of data is visible at the same time [39]. Löwe et al. [85] generate both scan paths and heat maps (which they call attention maps).

Heat map shows how often a part of a video (for example, a tile) was included in the users' field of view. It works by coloring the video, i.e., changing the color so that, for example, the most often seen parts of the video are in red, less often seen in green, and the least often seen in blue. Applications for heat maps include finding the areas of interest in 360-degree videos and comparing users' behavior when using different technologies [26, 139]. Duanmu et al. [38] made "view center plots" that display the view center distribution of participants in a heat map style. Not all heat maps are not drawn over the videos [49, 99]. Many related publications include heat map styled visualizations, but it is not usually clear if the heat maps are generated in real time.

Nguyen et al. [101] created a visualization showing collaborator's viewport which enables a user to understand what a collaborator sees while watching the same 360-degree video. This is a clear example of a real-time visualization taking account of distributed usage.

Traditional charts not drawn over the video have been much used in the related literature, but there are also some special cases. For example, Löwe et al. [85] produced a visualization showing users' field of view branching and joining over time on a timeline. Duanmu et al. [38] created somewhat similar plots, where each color corresponds to a specific participant. Ozcinar et al. [112] created polar charts, which is an interesting approach for taking account of the spherical context of 360-degree videos.

2.6.2 Other Applications of 360-degree Video User Logs

The logs can be used for other applications, where visual presentation is less obvious. On an abstract level, there are a few categories for the applications: changing the content of the video application according to logs during playback, product development, and recommendation systems that can learn based on previous logs.

One important reason for logging is viewport-adaptive streaming, where the idea is that only the visible area of 360-degree video is streamed in high quality. That way it is possible to optimize the video transmission and save bandwidth. This is also linked to predicting user behavior so that it is possible, for example, to buffer the expected high-quality video content in advance. Another example application for predicting user behavior to change the content of the application is to place advertisements over the video.

The logs can be used in product development for various purposes exploring user behavior. For example, developing UI, training algorithms, or studying storytelling is possible with log data. Logs can also help to study the effects of different video projections and video quality. Also, it is possible to reuse logged 360-degree video user data, and it can be inventively connected with other data sources, for example, to study physiological reactions.

2.7 Case: 360 Video Intelligence

The research for this thesis was mostly conducted on a research project called 360 Video Intelligence (360VI). The author's work in 360VI constitutes an illustrative practical case for the themes discussed in this thesis. The university's role in the project was to develop a platform that manages various 360-degree video analysis algorithms. For example, it was possible to send a 360-degree video file as an input to the platform and define which object detection algorithms should be executed for that video. Then, after the execution of the algorithm, the output contained information on the objects detected. The author's task was to develop applications for such output data combined with the original video using 360-degree video equipment, such as HMDs.

The 360VI project involved five local companies with different needs and interests regarding logging 360-degree video users. The interests of the companies, named from A to E, were the following. Company A-UI was developing a UI for 360-degree video media library application and was interested in logging the users to improve user experience. Company B-SPORTS developed 360-degree video applications for watching sports events and was interested in transferring only the essential parts of the video in high quality, where log analysis could help. Company C-TRAFFIC was developing a traffic safety education application and was interested into analyzing user logs to see if the users actually had the essential objects of a traffic video in their field of view. For example, they were interested into analyzing whether a user saw a traffic sign while watching the video. Company D-LEARNING developed a web learning environment with 360-degree video support, where students could potentially be analyzed with the help of logs. Company E-ALGORITHMS was primarily interested in 360-degree video analysis algorithms, so in contrast to the others, was not especially interested in user analysis. However, video analysis algorithms could be combined with user analysis in many scenarios as proposed in Publication PIII.

2.8 Gaps in the Existing Research

Some definitions for logging and user logging have been proposed, but the author is not aware of any definitions for 360-degree video user logging. Since the terminology related to logging is inadequate, it is important to establish proper definitions. Further, since logging alone is not very meaningful, it is important to study the reasons for logging. The author is not aware of the availability of extensive research on the reasons for 360-degree video user logging.

In general, logging practices lack formal specifications and systematic design [123]. This also applies to 360-degree video user logging. In the related scientific research, the logging is often done locally – the log data collected from 360-degree video watchers is stored in a local file system and (different) logs are integrated with each other in a separate processing step. This may be acceptable if there is no need for logging distributed users and making analysis in real-time. A log server is a solution that supports distributed data collection. However, the use of such an architecture has not been well studied in the context of 360-degree video user logging. The same applies to other related architectures. There are some examples available, but in general, there are only few properly presented logging architectures in the related scientific field. In addition, even if a presented architecture has potential for distributed data collection and real-time analysis, many practical details may be missing from the publication. The reason for not concentrating on logging related architectural details is probably that scientists are more often interested in presenting their applications or analysis results derived from their logs rather than in how logs are collected. In addition, their analysis is often something that can be done without an efficient solution.

Databases support distributed data collection and concurrent analysis. Again, use of databases has not been well studied in the context of 360-degree video user logging. This is understandable if there is not a log server available or a need for real-time analysis. Still, even if the logging and analysis were local, a database could be useful for collecting the logs in one place, thereby making analysis easier and more efficient. Further, the author is not aware of extensive research on real-time visualizations used for 360-degree video user log analysis.

Since the logging solutions are often local, only little information is likewise available on the architectural styles used for network-based applications because local solutions do not need a network. Distributed logging entails using a network which requires a communication protocol. There are some examples of client-server communication available in the literature, but the author is not aware of other communication styles, such as the explicit use of publish-subscribe for 360-degree video user logging. WebSocket has been used in a few publications, but it is not entirely clear if it has been used in publish-subscribe style [17, 107, 95, 108].

Let us imagine an example scenario. We are developing a mobile phone application with a 360-degree video player showing a live sports event. We want to analyze the users in real-time to place an advertisement over the video so that many users can see it. The challenge is to know where most of the people are watching during the sports event since we do not want the advertisement to totally block the video content. We decide to implement logging features for view orientation because it is possible to analyze users with the help of logs. However, for efficient logging of distributed users, the architecture of the logging system needs to be distributed. We decide to set up a log server with a database. This allows efficient distributed log collection of simultaneous users with support for efficient analysis. In addition, we need to choose a communication model and a protocol since logs and analysis results are transferred over a network. Publish-subscribe pattern has potential as a communication model since it allows an undefined number of users to easily publish log data, and one-to-many messaging allows the back end to publish analysis results which are then delivered by the broker to all interested users. The analysis of stored logs can be performed efficiently with the help of the database and the analysis result, the view orientation currently most popular can be used for placing the advertisement on the users' screens in real time.

3 RESEARCH APPROACH, METHODS, AND PROCESS

This chapter discusses the research approach adopted. Most of the research was qualitative, but there were also quantitative aspects. Design Science Research was used in the majority of the included publications and a systematic literature review was made in one publication, but non-systematic literature reviews and an interview were also made.

3.1 Design Science Research

Design Science Research was used in Publications PIII–PVI so it was the most important methodology. Design Science Research is often used in software engineering research [144]. It was chosen because it yields new knowledge via the development and analysis of innovative artifacts [144]. Development of software artifacts is a characteristic approach for the author as an engineer. Design Science Research methodology includes six steps: (1) problem identification and motivation, (2) definition of the objectives for a solution, (3) design and development, (4) demonstration, (5) evaluation, and (6) communication [115].

The downsides of Design Science Research include that there is no general agreement on "terminology, methodology, evaluation criteria, etc." [12]. For example, it is not obvious what is an IT artifact [12]. In addition, producing an IT artifact is not Design Science Research as such; the design of an artifact must lead to the development of new knowledge. Thus, Design Science Research implies "an ethical change from describing and explaining of the existing world to shaping it" [67]. Because of this, the researcher has responsibility to adhere to good research ethics knowing that it is clear that Design Science Research cannot be value-free [67].

The whole consisting of Publications PIII, PV, and PVI is described in accordance with the steps of Design Science Research in this and the following paragraphs. *1.*

Problem identification and motivation. The problem is that there is a need to log distributed 360-degree video users and analyze them in real time. There are not many easily accessible ready-made solutions available for the task. Motivation for logging comes from the high demand for user tracking in 360-degree video domain [26, 84, 85, 145], and from the needs of the 360VI project, such as exploring and predicting user behavior.

2. *Definition of the objectives for a solution.* The logging and analysis system must be distributed (need to share a dataset) and work in real time (continuous analysis). It must be able to efficiently store user logs into a database and perform flexible analysis of the logs. The logging communication via a network must have low latency to allow real-time functionality, and the communication pattern must not stress the back end unnecessarily because of the expected growth of usage in the future. The system must allow log visualizations, for example, by drawing over 360-degree videos on the end-user clients or by making traditional charts on an admin dashboard.

3. *Design and development.* The system has been designed as a distributed system with thin clients, a log server, and a database with SQL support. An early version presented in Publication PIII was developed with synchronous HTTP protocol but replaced with MQTT during the incremental development process as presented in Publication PV. The visualization technique for drawing over 360-degree videos was developed for Publication PIII and Publication PV used the same technique for drawing visualizations over the videos. Publication PVI used the client of the earlier publications but replaced the back end and the database with an IoT-related data integration platform. This helped to prepare for larger scale usage and made a general visualization dashboard with traditional charts easily available.

4. *Demonstration.* The system (Publications PIII and PV) was demonstrated in the project meetings of the 360VI project, and after the end of the project to IoT researchers (Publication PVI).

5. *Evaluation.* In Publication PIII the logging and visualization capabilities of the system were evaluated by visualizing previous user traces over 360-degree videos. In Publication PV the real-time performance of the system was evaluated by testing user analysis queries with a varying amount of log data and complexity. In Publication PVI, the system was evaluated from the viewpoint of the development challenges encountered during the study.

6. *Communication.* Peer-reviewed publications and the accompanying software

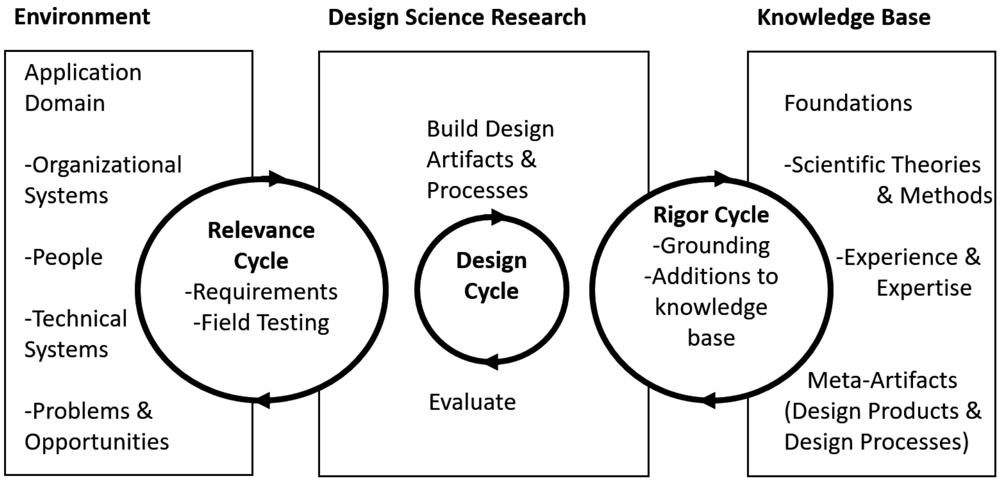

Figure 3.1 The three cycle view of design science research [60].

have been presented in scientific conferences.

Design Science Research can be seen as an embodiment of three cycles of activities: the Relevance Cycle, the Rigor Cycle, and the Design Cycle [60]. "The recognition of these three cycles in a research project clearly positions and differentiates design science from other research paradigms." [60]. The three cycles of Design Science Research are summarized in Figure 3.1.

The idea of the Relevance Cycle is to initiate the research with an application context that provides the requirements and the acceptance criteria for the research [60]. The output of the research must be tested in the real application domain, for example, by making a field study whose results determine possible additional iterations of the cycle [60].

In the course of the research, the requirements part of the Relevance Cycle was actively iterated. For example, the requirements of the system were updated according to the needs of the companies involved in the 360VI project. At the time of Publication PIII, the requirements were targeted at getting the results of the video analysis algorithms visualized on a 360-degree video player. These requirements were iterated, which resulted in leaning towards making visualizations over video as presented in Publication PV. Since such visualizations over video are not a good approach for all needs, the requirements were reiterated. This time the requirements included more emphasis on traditional chart visualizations as presented in Publication PVI. This iterative work culminates in the introductory part of the thesis, which explicitly discusses the architectural requirements of an effective 360-degree video logging and analysis system (RQ2). The requirements, including aspects, such as distribution, storage, analysis, and communication, were processed multiple times and polished to achieve a scientific credibility suitable also for the Rigor Cycle. The weakest part from the perspective of the Relevance Cycle is that proper field testing was not conducted. In addition, the system developed for PVI was not demonstrated to relevant 360-degree video software companies but only to local IoT researchers and in a scientific poster session.

The Rigor Cycle ensures the innovation of the research conducted based on the existing knowledge base of theories and engineering methods [60]. It is important to thoroughly research and reference the knowledge base to ensure that the research output is not routine design [60]. "Additions to the knowledge base as results of design science research will include any extensions to the original theories and methods made during the research, the new meta-artifacts (design products and processes), and all experiences gained from performing the research and field testing the artifact in the application environment." [60]

Publication PI started the Rigor Cycle by locating scientific literature related to 360-degree video software development, Publication PII made a more systematic study of the role of logging in VR research, and Publications PIII–PVI all have a related work section reasoning that the work presented is not routine design. PIV contributes especially to The Rigor Cycle by adding knowledge about MQTT. The introductory part of the thesis also includes literature reviews intended to ensure that the work presented is not routine design (RQ2 and RQ3), and also highlights how the results – as a whole – will contribute something worthwhile to the existing knowledge base.

The Design Cycle is about generating design alternatives and evaluating them against requirements until a satisfactory design is achieved [60]. "The design cycle is where the hard work of design science research is done." [60] It is a relatively independent cycle but it is important to understand the connection to other cycles [60]. The work in the Design Cycle includes balancing between relevance and rigor while doing the construction and evaluation both of which must be convincingly based on relevance and rigor [60].

During the research, the Design Cycle was the most active during the practical implementation presented in Publications PIII, PV, and PVI. However, the design work is based on the Rigor Cycle started in Publication PI. The first publication presenting a design alternative is Publication PIII, where the design is based on HTTP, an RDBMS, and a visualization system drawing over a 360-degree video. After that, the Rigor Cycle went forward with Publications PIV and PII, and, with the added knowledge of these, it was possible to create the second design alternative presented in Publication PV. The design is based on MQTT and RDBMS, and the publication presents more a complex user analysis. With the contribution of Publication PV to the Rigor Cycle, it was possible to go forward with the Design Cycle and build the design alternative presented in Publication PVI. That alternative uses an IoT-related data integration platform, MQTT, and a combination of NoSQL and NewSQL databases. The balance with the Relevance Cycle was struck by emphasizing the evaluation against the requirements of the system, especially towards the end of the research, while writing the introductory part of the thesis (RQ1, RQ2, and RQ3). All the main publications of the Design Cycle (PIII, PV, PVI) also include some kind of evaluation but the introductory part of the thesis takes explicit account of the requirements of the companies of 360VI (RQ1, RQ2, and RQ3).

3.2 Literature Review

Systematic literature review (SLR) methodology was used in Publication PII. According to Kitchenham [73], "A systematic literature review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest." There are various reasons for conducting a systematic review, such as to summarize the existing evidence related to technology, to identify current gaps in the research, or to provide background for new research activities [73]. SLR method was chosen primarily to confirm the suspicion about a gap in the related logging development research. The results are presented both qualitatively and quantitatively.

Non-systematic literature reviews are feasible when conducting a systematic review is not possible [56]. These reviews have poorer replicability and transparency, and a higher risk of bias. However, systematic reviews are time- and resource-intensive, which may discourage researchers operating on limited resources from conducting full systematic reviews [56]. The method was chosen due to lack of resources in a situation where the author reasoned that a systematic review was not essential. Publication PI used a non-systematic literature review to make a list of the important features and characteristics of 360-degree video SDKs. A non-systematic literature review was used in Publication PI because initial searches suggested that there was not much relevant literature available on the topic so making a full systematic literature review would have been too heavy a process. Non-systematic literature review was also used in all the other publications to present a "Related Work" section or improve background. Non-systematic literature reviews also help to answer Research Questions RQ1, RQ2, and RQ3 in the introductory part of this thesis.

3.3 Interview

One semi-structured face-to-face interview was conducted for Publication PI. Interview is not the only method used in Publication PI, but it extends the findings based on other methods. The questions were related to experiences of360-degree video SDKs and to the most important and missing features in these. The method was chosen to gain expert insight into the topic at an early phase of the research. Face-to-face interview is a method for eliciting expert knowledge about a certain topic, but it requires the interviewer to concentrate intensely on the questions and answers [109]. The semi-structure helped the author to follow a certain plan but allowed digressions into for other interesting topics. The interview was conducted adhering to good research ethics. For example, the interviewee signed a document confirming that the data would only be used for research purposes, that the interviewee could withdraw from the interview whenever he wanted, and he would have an opportunity to see the text prior to its publication, etc. The interview was recorded which helps in making more accurate interview reports [109]. Conducting and transcribing interviews can be laborious [109]. However, in the case of a single interview, time consumption and transcription were not a problem.

3.4 Process

The relationships between the publications included, the main chapters presenting results and Research Questions RQ1, RQ2, and RQ3 are summarized in Table 3.1. A relationship shows which research questions the publication or the chapter contributed to answering. In some cases, the relationship is clearer than in others, but the table provides guidance.

Table 3.1 Relationships between the publications, chapters and the research questions.

Publication	RQ1	RQ2	RQ3
PI	X	X	
PII	X		X
PIII		X	X
PIV		X	
PV		X	X
PVI		X	X
Chapter 4	X		
Chapter 5		X	
Chapter 6			X

The research process progressed as follows. First, the author wrote a publish-subscribe related preliminary conference version of Publication PIV which is not included in the thesis. The paper reports a study where HTTP used in an IoT system was replaced with an IoT-oriented protocol called MQTT. Then, the 360VI project started up, after which began the writing of Publication PI. The idea of the paper was to collect the principal needs of 360-degree video software development. The publication lays the groundwork for logging discussed in the later publications. At the time of finalizing and publishing Publication PI, the preliminary conference version of Publication PIV was published. After these, the author simultaneously worked with 360-degree video user logging server related Publication PIII and publish-subscribe related Publication PIV. Publication PIII presents the first implementation of a 360-degree video logging system developed by the author. Publication PIV extends the mentioned early conference version, for example, by emphasizing the benefits of MQTT and presenting a resource consumption comparison between HTTP and MQTT. At that time, an idea emerged that publish-subscribe could be used for 360-degree video logging. However, there was a need to justify the idea of the thesis properly by studying the role of and the reasons for logging, which led to publishing the SLR of Publication PII. Publication PV then combined a 360-degree video user logging server with publish-subscribe, and presented real-time user log visualizations. This was the last publication written during the 360VI project. Finally, Publication

PVI, that was written on an IoT-related project, presented another logging server solution using publish-subscribe pattern. The solution was based on a general-purpose data integration platform which reduced the amount of custom code and allowed easy usage of a web dashboard for visualizations.

The chronological publishing order was PI, PIII, PIV, PII, PV, and PVI. Publications PIII, PV, PVI describe the core constructive results, whereas Publications PI and PII relate more to the background. Publication PIV is rather different from the others, since it concentrates on IoT domain, but the publish-subscribe related results are also applicable to 360-degree video user logging.

The research questions of the introductory part cannot be properly answered only with the help of the included publications and the introduction presents a broader discussion on the topics. For example, from the viewpoint of RQ1, Publications PI and PII do not properly define "360-degree video user logging"; that is the task of Section 4.1. Publications PI and PII include some reasons for logging 360-degree video users, but the introduction includes a more profound analysis in Section 4.2. The role of the introduction in answering RQ1 is to contribute to the Rigor Cycle by mainly using non-systematic literature reviews.

RQ2 is partly answered in Publications PIII, PV, and PVI by presenting examples of efficient distributed logging and analysis systems. Publication PI lists performance among important development aspects, and Publication PIV proposes MQTT as an efficient replacement of HTTP. The introductory part of the thesis (Chapter 5) presents a designed reference architecture, based on the needs identified in the literature and the company cases of 360VI, and an architectural comparison of related implementation alternatives based on a non-systematic literature review. Thus, the role of the introduction in answering RQ2 is to contribute to the Relevance Cycle and to the Rigor Cycle.

Finally, to answer RQ3, Publication PIII presents a real-time visualization technique for drawing over 360-degree videos, that is also used in Publication PV, and Publication PVI presents visualizations made with another technique. The introduction of the thesis (Chapter 6) discusses visualizations in light of the needs of the companies of the 360VI project and compares them to those found by the non-systematic literature review of RQ1. The answer to RQ3 also summarizes the visualizations developed in the included publications. The role of the introduction in answering RQ3 is to contribute to the Rigor Cycle and to the Relevance Cycle.

4 USER LOGGING IN 360-DEGREE VIDEO DOMAIN

This chapter is the first chapter presenting results. First, a definition of user logging in 360-degree video domain is formulated using scientific literature as material, after which the reasons for such logging found in the literature and the needs of the companies on the 360VI project are categorized. This results in a list of popular reasons for user logging in 360-degree video domain. The results of the chapter are partly based on the results of the systematic literature review in Publication PII.

4.1 Definition of User Logging in 360-Degree Video Domain

360-degree video user logging has its own special characteristics. A definition for it can be created with the help of the relevant literature. Scientific publications discussing the topic do not define the term or indeed use it as such. Instead, they often use synonyms for logging, such as "data collected from users watching 360-degree videos" [26], "head position samples are recorded" [26], "we captured rotational head movements" [26], "collect" [43, 84, 38], "record" [43, 84, 85, 117, 9, 98, 99, 28, 49], "user study traces" [117], "user behavior records" [145], and "sensors signal acquisition" [86].

However, the term "logging" has also been explicitly used, for example, in the following way:

- [26]: "log files", "head movement logger"
- [43]: "sensor logger", "logged sensor data"
- [84]: "sensor logger", "data log file", "to log"
- [145]: "log" as a part of a directory structure containing user datasets
- [9]: "motion is recorded and logged"
- [38]: "data log"
- [101]: "we logged"

The following common log items have been mentioned in the literature:

- View orientation in yaw, pitch, and roll format [43, 84, 117, 86, 9, 38, 49] and in quaternion format [26, 145, 99]
- Video timestamp [26, 43, 84, 145, 99, 38]

Other logged data mentioned contain video viewer's position in Unity tool's 3D space [26], local timestamp [26], frame id [26, 84], seen tile [84], gaze direction [85], gyroscope [86], accelerometer [86], and magnetometer [86]. Liu et al. [83] plan to collect video URL, user's rating of the video, contextual information (indoor, outdoor, watching mode), mobility, and posture. In addition, some studies make a video capture of what is seen by the users [43, 84].

The logged data is most commonly data received via the sensors of the device. That data is then processed as view orientations with video timestamps. However, 360-degree video user logging is not limited to such data. Fan et al. [43], for example, describe the head orientation logging process as follows: "We use the word orientation loosely to indicate features that can be inferred from HMD sensors. Our current implementation only considers orientation, but can be readily extended to other features". In general, enumerating the most interesting log data for 360-degree video domain is difficult since, as Chuvakin and Petersen [23] note, it is impossible to create a comprehensive "what to log" list for every application. In addition, it must be noted that the idea is not to log the contents of the video, but the user's actions while watching the video.

One special characteristic of 360-degree video user logging is the pace of logging, which comes from the frequently updating video frame and relatively quickly moving viewing device (often attached to the user's head). If the user's view orientation is to be logged for every video frame, then, for example, a video with 23 frames per second can trigger a log entry every 43th millisecond. The following sampling rates can be found in the literature: 4 Hz [28], 7-9 Hz [9], 10 Hz [98], 30 Hz [26], 60 Hz [99], 30-250 Hz [84], 100 Hz [85], 250 Hz [117], 100 Hz [145], and "1000 Hz at most" [86]. Of course, there may be other triggers than an updating video frame for making a new log entry, but an updating video frame makes it possible to see the differences in view orientations between sequential frames. In any case, as the sampling rates found in the literature suggest, the logging pace is relatively fast although a slower pace might suffice for many needs.

Based on these characteristics, the author of this thesis defines "360-degree video user logging" as follows: *360-degree video user logging is a systematic and frequently occurring recording of user data of video watching equipment or software. The recording happens while using 360-degree video watching software. The most typical recorded data are view orientations with the related current video timestamp. View orientation data may be accompanied by other data that is not directly related to view orientations. For example, it is possible to log user actions, such as pressing pause or play, and physiological responses, without a direct linkage to view orientations.*

View orientation logging overlaps to some extent with the more general term "body tracking". As the definition suggests, 360-degree video user logs may contain other related data such as general body tracking data. However, when discussing logging of 360-degree video users, calling the logging process "body tracking" would be misleading. In addition, the definition does not consider the collection of data via questionnaires (during or after video playback) 360-degree video user logging but general data collection.

In addition, it is good to note the difference between logging head orientation and eye orientation [85]. While head orientation provides information about the viewport watched, eye tracking can determine where the user is focused within that viewport. Eye tracking has multiple applications, for example, in usability development [53] and user experience work [16], and it is becoming increasingly important since the consumer devices are starting to support it. Nevertheless, the research presented in this thesis only covers the information based on the device orientation because the focus is on platforms that do not directly support eye-tracking. However, the work is also applicable to eye tracking. As interesting and informative user interactions related to video playback such as "pause" and "skip" could be, this thesis does not take such information into account, although adding such functionality to the systems implemented should not entail much work.

4.2 Reasons for Logging Users in 360-Degree Video Domain

The discussion in Ritchie et al. [121] about the benefits of user logging in VR applies equally to 360-degree video user logging: it is an almost non-intrusive method of capturing a rich data source for analysis by minimizing user interactions during data capture, and the captured data can be reused. It is a non-intrusive method, for example, when compared to methods where users describe their actions verbally or in writing.

Watchers of videos may be logged for various reasons. In general, by recording the user activity while watching 360-degree video, it is possible to conduct an analysis of the user's behavior [85]. The example reasons for logging listed in Table 4.1 can be found in the research literature. In addition, the table includes the logging reasons of the companies of the project 360VI. There are many duplicates in the table as, for example, some of the publications in "exploring and predicting user behavior" category can be also found in the "viewport-adaptive streaming..." category.

From the listed examples, "viewport-adaptive streaming, transmission optimization, and saving bandwidth" is the reason that has received the most attention. It aims at efficient 360-degree video delivery by maximizing the video quality in the viewport and minimizing the waste of bandwidth on those parts of video that are never displayed to the user [26]. However, logging has an even greater range of applications. For example, Lo et al. [84] note that their dataset can be "leveraged in various novel applications in a much broader scope." and Fan et al. [43] consider that their head orientation-based approach is readily extensible for eye-tracking. "Scientific research" is a very broad reason as well [86].

Logging as such can be relatively simple. It is the domain and the use cases which make the topic interesting and challenging. For example, real-time log analysis for multiple simultaneous users makes the logging task more complex. In such a case, logging requires a distributed architecture where different parties communicate via network. Designing such a system requires architectural knowledge that is the topic of the next chapter.

Table 4.1 Example reasons for logging users in 360-degree video domain.

Category	**Publication**	**Company**
Offering a public dataset for reuse	[26, 84, 145, 49, 38]	-
Viewport-adaptive streaming, transmission optimization, and saving bandwidth	[26, 117, 63, 64, 147] [84, 145, 9, 99, 98] [27, 38, 136, 83, 43] [107, 108]	B-SPORTS
Training systems and algorithms	[84]	-
Investigating storytelling and analyzing viewing experiences	[85, 99]	-
Exploring and predicting user behavior	[145, 38, 49, 84] [43, 17, 95]	A-UI, B-SPORTS, C-TRAFFIC, D-LEARNING
Scientific research	[86]	-
Investigating video projections and quality	[27]	-
Studying collaboration	[101]	-

5 ARCHITECTURES FOR LOGGING 360-DEGREE VIDEO USERS

This chapter discusses user logging architectures in 360-degree video domain. The chapter starts by specifying the architectural requirements given on the reasons for logging. To fulfill the requirements, a reference architecture for distributed logging and analysis is presented. Then, implementing the reference architecture is discussed in light of the 360VI project. Finally, architectures found in the related scientific literature are compared to those presented in the publications.

5.1 Architectural Requirements Derived from Reasons for Logging

Table 4.1 categorized reasons for logging and analyzing 360-degree video users found in the literature and in the company cases of the 360VI project. This section specifies the architectural requirements that can be derived from these reasons. The following three reasons, the most popular according to Table 4.1, are used specify the architectural requirements: "Viewport-adaptive streaming, transmission optimization, and saving bandwidth" (VAS), "Exploring and predicting user behavior" (EPB), and "Offering a public dataset for reuse" (OPD). The two most popular reasons (VAS and EPB) have been prioritized in the requirements, but the third most popular reason (OPD) has been also taken into account. OPD is a slightly different need because it does not directly relate as much to efficient analysis support as publishing scientific datasets does not have equally strict time requirements. The listed functional requirements (R1–R4) can also support the other reasons in Table 4.1 to some extent, but the idea was not to prioritize them. The non-functional requirements (R5–R7) are also important, but they cannot be directly derived from the reasons in Table 4.1. However, they are important characteristics of a successful general-purpose architecture.

R1-Distribution. The architecture must be distributed since the three most popular reasons benefit from shared resources and the software has geographically distributed simultaneous users. For example, dynamic VAS should be adaptive in real time for best results. Distribution is required because it would be impossible to first store logs locally, combine them without a network, and analyze them in real time. Those steps need to be executed in a distributed system as well, but they can be automated and executed concurrently starting immediately from the first log event. Another problematic solution would be to use a single end-user device for all the logging, for log storage, and for real-time analysis since it would not allow geographically distributed simultaneous usage. As in VAS, EPB with a distributed architecture would make it easier to collect data from multiple simultaneous users into a common dataset, which can be used for studying user behavior taking into account multiple global users and not just one local user. From the perspective of OPD, collecting data into a shared common dataset removes the need to integrate different geographically fragmented datasets when publishing it.

R2-Storage. Since data is being collected for both real-time (VAS) and non-real-time analysis (some EPB cases), it needs to be efficiently stored so that the solution supports simultaneous high-performance usage. Real-time processing does not always need data storage, but there may be situations where real-time analysis is prioritized and there are plans to do more complex non-real-time analysis that necessitates storing data for later use. Efficiency in storage includes data access and management, flexibility, and support for the other requirements (R1-Distribution, R3-Analysis). For example, from the viewpoint of R1-Distribution, the storage must support use cases requiring a shared common dataset. From the viewpoint of R3-Analysis, the storage solution must support accessing the data and performing log analysis because it would be a waste of resources to always move the data out of storage for analysis. Efficient storage also supports the need to make public datasets (OPD) since efficient data storage techniques allow better data management features than, for example, simple text files.

R3-Analysis. There must be a way to efficiently analyze the log data so that the results are available in real time, for example, for adjusting the quality of continuously running video stream (VAS). The analysis results must be readily available for visualization components or other possible interested parties, that improves the versatility of the system. The architecture must also support efficient non-real-time analysis because, for example, some EPB use cases do not require real-time analysis.

Some public datasets include ready-made data aggregations or analysis examples so analysis support from the perspective of OPD can be also beneficial.

R4-Communication. The logging and analysis communication must have low latency to allow real-time adaptation to users' behavior, for example, for VAS. The sampling rate of 360-degree video user data can be relatively fast, so light weight is required. In some EPB cases one-to-many communication is needed for delivering real-time analysis results to multiple distributed parties simultaneously.

R5-Reusability. Reusability is required because any of the components may be subject to change because of changing requirements. Thus, it is important to able to reuse components when applicable.

R6-Extensibility. It must be possible to extend the functionality of existing components. The architecture must support the addition of new features and the modification of existing functionality without significant architectural changes.

R7-Scalability. It must be possible to scale the system if the number of users grows. The architecture should support horizontal scaling. For example, replicating critical back-end components must be possible without significant re-designing.

To summarize the requirements, the architecture must be distributed, it needs an efficient data storage and analysis solution, and the communication must be lightweight and allow one-to-many communication. In addition, the most important non-functional requirements include reusability, extensibility, and scalability.

5.2 Reference Architecture

The reference architecture based on the requirements in Section 5.1, is presented in Figure 5.1. In the figure, arrows represent different kinds of communication: thick arrows present the movement of log data, dashed arrows present the movement of log analysis results, and thin arrows represent the direction of publish-subscribe communication (containing the log data and analysis results as payloads).

The components of the reference architecture are defined in Table 5.1. The reference architecture is described as follows. User interface providing a 360-degree video player is located on end-user devices (Data Sources). Log messages from users are sent via a network to a log server. The role of a log server is to receive log messages and to manage other common logging session related functionality, such as registering view sessions and sending data analysis requests to the database. A log server is

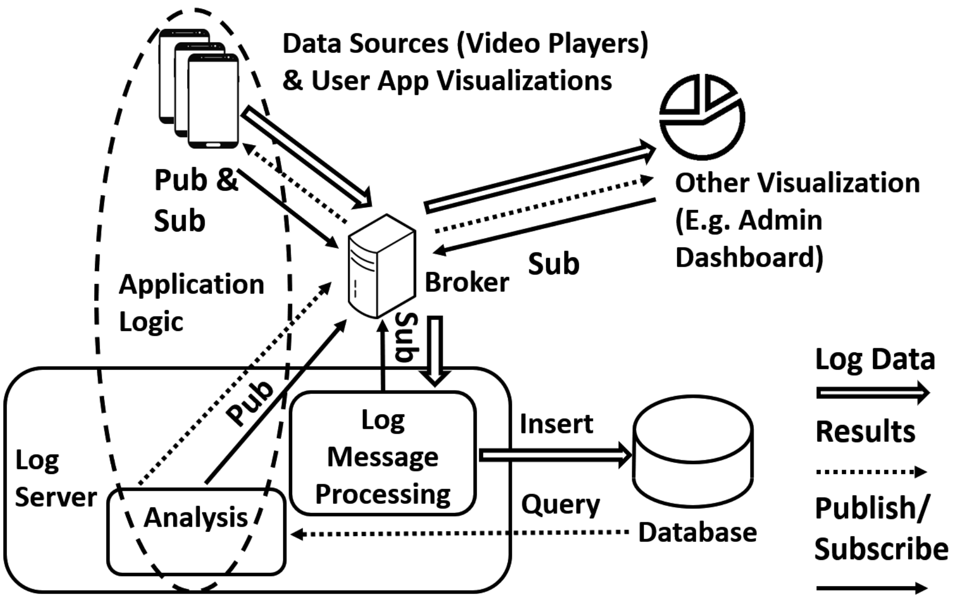

Figure 5.1 Reference architecture for logging and analyzing 360-degree video users.

connected to a database for efficient storage and management of log data. The log server sends analysis results back to end-user devices for visualization. However, the end-user UI is not the only party interested in analysis results. Figure 5.1 also includes another visualization component that receives analysis information from the log server. It can present the analysis results on a different platform and in a different format compared to the end-user UI. An example of such a component could be an admin dashboard. However, the reference architecture does not require usage of such additional visualization components. The 'Other Visualization' is only an example showing that the architecture supports various applications via the shared services offered by the log server.

Communication between users and a logging server, depending on the use case, may be connection oriented, connectionless, message-oriented, synchronous or asynchronous. However, it is recommended to use MOM because it enables asynchronous communication and publish-subscribe pattern. For example, the reference architecture with a MOM could be based on publish-subscribe pattern that is used for communication between data sources, log server, and additional (visualization) components. Such a solution would take into account the low-latency viewpoint of Requirement R4-Communication. Logging via publish-subscribe can be effective since

Table 5.1 Definitions of the components of Figure 5.1.

Component	Definition
Log Server	A server that is used for log message processing and forwarding the logs to a database.
Log Message Processing	Log message processing is a part of the log server. Its role is to subscribe for log events from the broker, process log messages, and write them to the database.
Analysis	Analysis component does data analysis in which it needs to query the database for data and (temporary) results. It also publishes the analysis results to the broker.
Database	Database is the place where individual log records are stored. It also has a technique for processing log analysis queries and returning the result.
Broker	Broker handles message delivery in publish-subscribe communication pattern.
Data sources	Data sources are end-user devices that produce 360-degree video user log data and send it to the broker for further delivery.
User App Visualizations	The end-users of the data sources are also interested in the log analysis results so their 360-degree video player applications have features to visualize or make use of the log data analysis results in other way.
Other Visualization	Other Visualizations refers to other visualization applications that are not installed in the end-user devices. Such applications include, for example, admin dashboards.

a reply of request-reply pattern is not always needed, it is possible for different parties to dynamically subscribe to log data, and asynchronous communication can be considered more effective due to decoupling. An architecture with publish-subscribe supports the addition of various visualization components because it is easy to subscribe to analysis results. Publish-subscribe communication could be implemented with MQTT.

Figure 5.1 depicts the data sources as handheld mobile phones having a logging 360-degree video player, but they could equally well be HMDs. Those logging devices send log data to a log server via a publish-subscribe broker. A log server contains two modules, a module for processing incoming log messages and a module for executing log analysis. The log message processing module writes log entries into the database, and the analysis module sends queries to the database. Then, the analysis module sends the results to interested subscribers. The analysis module could be implemented as a separate component although in the figure it is placed inside the logging server. The analysis module of the log server sends the results to interested parties via publish-subscribe broker that delivers the results to all the interested parties. The dashed 'Application Logic' area means that the analysis module of the logging server and mobile applications form an end-to-end application. The UI of data sources can then visualize the analysis results, such as the most popular tiles, over 360-degree video. However, the users of logging devices were not the only subscribers interested in the analysis results. The results can be also delivered to a web-based visualization dashboard intended for administrators of the system. The administration view contains only traditional chart visualizations. This emphasizes that the analysis results can be visualized differently according to the needs of the subscriber. It would be relatively easy for any new component to subscribe to the data published via publish-subscribe.

5.3 Implementations for the 360VI Project

The author implemented a 360-degree video user logging and analysis system for the needs of the project 360VI. The architectural requirements of Section 5.1 were fulfilled as follows.

R1-Distribution. The reference architecture is implemented as a distributed system in Publications PIII, PV, and PVI. In Publication PIII, the implementation is made using a request-reply protocol HTTP, a custom Node.js log server, and a relational database. In Publication PV, the implementation is made using the same components as in Publication PIII, but HTTP is replaced with a publish-subscribe protocol MQTT. In Publication PVI, the implementation is made using MQTT, an IoT-related data integration platform as a log server, and a combination of NoSQL and NewSQL databases.

R2-Storage. Databases are used for storing the data in all the relevant publications (Publications PIII, PV, and PVI).

R3-Analysis. The architecture used in Publications PIII and PV allows SQL-based analysis. Similarly, in Publication PVI the usage a NewSQL database enables SQL. The analysis results are readily available due to architecture with a log server and a database.

R4-Communication. MOM used in Publications PV and PVI provides support for loose coupling and asynchronous communication. MQTT is a lightweight publish-subscribe protocol with low latency and support for asynchronicity. It also allows one-to-many communication that can be used for efficient delivery of analysis results to multiple subscribers.

R5-Reusability. The openness of the distributed reference architecture supports reusability as can be seen in Publications PIII, PV, and PVI. The log source implementation of Publication PIII required only minor modifications for Publication PV when changing the HTTP server to a server with MQTT support. Further, the log server was again changed in Publication PVI to an IoT-related data integration platform while continuing the reuse of the same log source components.

R6-Extensibility. The openness of the distributed reference architecture also helps with extensibility. For example, it was possible to develop new visualizations without significant changes to other components. In other words, the UI of Publication PIII was extended in Publication PV.

R7-Scalability. RDBMSs were used in Publications PIII and PV, but they are not traditionally very scalable. Scalability was improved in Publication PVI by using an IoT-related data integration platform with NoSQL and NewSQL databases that could be relatively easily scaled further. The chosen MQTT implementation used in Publications PV and PVI supports horizontal scaling.

Publications PIII, PV, and PVI present more details of the implementations, but they are summarized here. The end-user device is a mobile phone that has an application with 360-degree video player implemented with Google VR SDK. The SDK did not provide support for drawing graphics over the video, so the author developed a way to draw simple graphics over the video. In Publication PIII, the implemented architecture with a log server and a database made storing of logs and visualizing earlier view sessions relatively efficient. In Publication PIII, the log data were transferred via HTTP to a log server, and then stored into an RDBMS. Since the data were stored in an RDBMS, they were available for analysis with SQL. Due to the drawbacks of HTTP for the use cases, it was replaced with a publish-subscribe protocol MQTT in Publication PV. Additional reasons for choosing MQTT include the experience the research group had with it so it just happened to be an easy protocol to adopt. The original reason for the research group for using MQTT was in IoT studies that required traversing of network restrictions as presented in Publication IV.

Visualizations over end-user video player are not practical in every case. A web dashboard with more traditional charts can sometimes be more useful. Therefore, in Publication PVI mobile phones with 360-degree video players were connected to an IoT-related data integration platform based on FIWARE that used Grafana tool for visualizations. The connection was implemented with MQTT. By default, FIWARE uses NoSQL database MongoDB in such a way that only the latest values are stored. Storing data that way is not useful for investigating temporal changes in data, so FIWARE can be connected to a NewSOL database CrateDB that stores the whole value history. The combination of CrateDB and Grafana visualization tool worked well for the task. The change of the log server and the database also serves as an example of how the architecture supports openness – it is relatively easy to change components.

5.4 Architectures in Related Research

This section presents the architectural characteristics of 360-degree video user logging and analysis systems found in the related literature. The characteristics included are architectural style (distributed architecture with a log server or local architecture), network communication protocol, and database usage. Architectural style was included because it can have a huge impact on the characteristics of software. For example, using a local architecture can prevent the implementation of certain use cases

Table 5.2 Architectural alternatives in the scientific literature.

Publication	Architecture	Comm. protocol	Database
Bibiloni et al. [17]	Log server	HTTP, WebSocket	NoSQL
Oliver et al. [107]	Log server	HTTP, WebSocket	NoSQL
Molino et al. [95]	Log server	HTTP, WebSocket	NoSQL
Oliver et al. [108]	Log server	HTTP, WebSocket	NoSQL
Nguyen et al. [101]	Log server	Client-server protocol	RDBMS
Duanmu et al. [38]	Log server	?	-
Qian et al. [117]	?	UDP	?
Fremerey et al. [49]	Local	-	-
Almquist et al. [3]	Local	-	-
David et al. [30]	Local	-	-
Luo et al. [86]	Local	-	-

(R1-Distribution). The author of this thesis was interested to see how log servers have been adopted in 360-degree user logging research. In general, architecture is important because architecture is the primary focus of software engineering for the production and development of successful high-quality products [89]. Databases are included because data storage is an essential part of a logging system. Data needs to be stored somewhere so that it is accessible for further processing. Chuvakin et al. [24] state that "Storage of logs and being able to quickly retrieve and analyze logs is a critical issue within an organization." Network communication protocols are included because distributed logging requires network usage [24]. The communication protocol can have an impact on the effectiveness of an architecture since protocols have different features and characteristics [71]. Table 5.2 lists the logging solutions found in the related literature. More publications (for example, [9, 43, 112]) present somewhat relevant logging and analysis studies, but do not contain enough architectural content to be included into the table.

The possible values for the architecture column of Table 5.2 are "Log server", "Local", and "?". Log server refers to an architecture with a central log server as presented in Figure 2.1 (page 34). Local means that the log collection is non-distributed. An unknown architectural style is marked with a question mark (?). The solutions with a log server could have potential for effective distributed logging and analysis [38, 17, 107, 95, 108, 101]. The local solutions do not have a distributed architecture, which makes them unsuitable for many use cases. The publication marked with a question mark in the architecture column most likely does not have an efficient distributed architecture because it was presumably not designed for distributed usage. There may be more such publications where the architecture is not clear, but the author believes that the list serves as an adequate example and that adding more publications with missing information would not produce much useful knowledge. For example, many publications related to viewport-adaptive streaming have a 360-degree video streaming server, an HTTP-based video transmission, user tracking, and may include logging in a small role, but is not clear if the logging is local or server-based.

The communication protocol column in Table 5.2 indicates whether the publication discusses a usage of communication protocol suitable for distributed log collection and real-time analysis. Local solutions do not have a communication protocol, which is marked with a dash (-), and unknown is marked with a question mark (?). A group of publications [17, 107, 95, 108] discuss an architecture that can be used for distributed logging using HTTP and WebSocket, but, for example, Bibiloni et al. [17] only express a wish for real-time analysis. Nguyen et al. [101] mention using a client-server architecture, so they presumably use some client-server protocol. Qian et al. [117] use a connectionless UDP protocol and possibly have a some kind of client-server architecture, but the log collection is not geographically distributed.

The database column of Table 5.2 shows the database type (NoSQL or RDBMS). Solutions presumably lacking a database are marked with a dash (-), and unknown is marked with a question mark (?). Nguyen et al. [101] use an RDBMS. A group of publications uses the same solution based on a NoSQL database [17, 107, 95, 108].

5.5 Architectural Comparison of Related Research and Developed Solutions

This section presents an architectural comparison of the solutions presented in the included publications to those found from the related scientific literature. Table 5.3 has a format very similar to Table 5.2. The differences are that Table 5.3 contains only the publications included in this thesis and the database column can have a database type NewSQL.

Table 5.3 Architectural solutions in the publications.

Publication	Architecture	Comm. protocol	Database
PIII	Log server	HTTP	RDBMS
PV	Log server	MQTT	RDBMS
PVI	Log server	MQTT	NoSQL & NewSQL

5.5.1 Architecture

Publications PIII, PV, and PVI discuss a log server solution based on the reference architecture. In Table 5.2, there are six publications with a log server architecture. Bibiloni et. al [17] implemented a server-based modular system for logging 360-degree video users. The solutions presented by Molino et al. and Oliver et al. are based on the same or on a very similar solution as that by Bibiloni et al., but they do not explain the architectural details [107, 95, 108]. In one case, they reason their architecture to allow real-time analysis [107], whereas in another case they considered it as future work [95]. Comparing their architecture to the architectures implemented in the publications, it can be seen that they all have log server architectures. Thus, those solutions have potential for efficient logging and analysis.

Nguyen et al. [101] studied collaboration in 360-degree videos and briefly mention having a client-server architecture. Duanmu et al. [38] used a server for storing logs, but no additional details are provided so it is difficult to say whether their solution supports logging of simultaneous users. However, both of these solutions have been interpreted as having a log server similarly to the implementations described in the publications. In general, solutions with a server have more potential than local solutions for efficient logging and analysis.

There is not much to say about the rest of the scientific papers in Table 5.2 in the terms of the comparison. Usually the description of logging and analysis architecture is rather poor in them. The author concludes that those systems were designed for local user log collection and the analysis is not generally in real time or automated. The aim of these studies is something other than presenting an efficient distributed architecture.

The benefits gained via distributed architecture can be reflected against the four goals of distribution (sharing common resources, hiding implementation details, openness, and scalability [133]). The reference architecture implemented in Publications PIII, PV, and PVI is distributed primarily because of Requirement R1-Distribution that includes *shared resources*. Databases can be used for sharing a common dataset. They are designed to work in a distributed setting and offer many beneficial features. Nevertheless, the other goals of distribution are also important. A distributed system *hides implementation details* so that users do not need to know that the system they are using is a distributed system [133]. While using a system based on the reference architecture, the end-user should not need to worry about the implementation objects. For example, the end-user may know they are being logged but there is no need to know if the log is stored into a local device or into a networked database. *Openness* allows easier development because components can be easily used by, or integrated into other systems. For example, when using the reference architecture, it is possible to change or re-implement components relatively easily with no need to modify the implementations of other components. This can be seen in the research prototypes developed during the study because there were no strict requirements to use particular platforms or software components, which made it possible to change the components of the architecture relatively freely. *Scalability* helps with the expected growth in the number of future users. During the research, the reference architecture was tested with only a few users but the expected growth of usage was kept in mind during the development.

To summarize, the positive consequences of using a distributed architecture with a central log server are that it is possible to use a common dataset, the server is easy to integrate with a database, thereby enabling efficient storage and analysis, and distributed architecture allows replacing and reusing components relatively easily. On the other hand, distribution increases complexity; there is a need to use a network for communication and there is a need to design the responsibilities of each component of the distributed system. Implementing a log server includes the trouble of implementing and setting up the server with a database. The work pays itself back if there is a need for distributed logging since it would be difficult to combine geographically fragmented logs for analysis.

5.5.2 Communication Protocol

Publication PIII uses HTTP, and Publications PV and PVI use MQTT. The publications in Table 5.2 do not explicitly discuss publish-subscribe although WebSocket can be used in publish-subscribe style [72]. Bibiloni et. al [17] (and the other related solutions [107, 95, 108]) made an implementation that supports WebSocket and HTTP, but they do not reason why they used them. However, WebSocket can provide efficient messaging [72]. Qian et al. [117] used UDP protocol, which serves as an example of using a connectionless protocol.

There are characteristics in MQTT that make it more attractive than HTTP (low latency, light weight, simplicity, one-to-many communication, and decoupling). Several studies suggest that MQTT performs significantly better than HTTP [70, 126, 59, 71].

MQTT is decoupled in several dimensions: publishers and subscribers do not need to know each other, operations on both components do not need to be interrupted during publishing or receiving, and the interacting parties do not need to participate in the interaction at the same time [41]. These features can be implemented to some extent in HTTP-based solutions by using detours, but they are already in MQTT by design.

If a publish-subscribe protocol is chosen, it is good to note that an IoT development survey 2016 [129] suggests that MQTT and HTTP are the most popular IoT protocols. Dizdarevic et al. [37] comment on the survey that the reason for this may be that MQTT and HTTP are comparably mature and stable standards. Even though MQTT has many benefits, a connectionless communication protocol, such as UDP, could also be used in many cases.

Table 5.4 summarizes the consequences of collecting log data onto a centralized server with centralized topic-based publish-subscribe instead of request-response. Some of the items are not present in every logging system since, for example, not every system sends analysis results back to data sources or needs one-to-many communication. The consequences are discussed in the following paragraphs.

Data sources and servers need to subscribe to relevant messages. By contrast, request-response requires the server to be accessible for pushing and pulling data. In the context of this thesis, this means that data sources sending 360-degree video logs need to subscribe to topics where analysis results are published by the log server, and the log server needs to subscribe to log data published by the log sources.

In request-response, data sources (or anyone who needs analysis results) need to poll the server for analysis results, whereas with publish-subscribe, the data sources do not need to poll for results because the log server can publish the results when they are available. Publication PV presents a solution where a centralized log server

Table 5.4 Comparison of publish-subscribe and request-response when collecting log data onto a centralized server.

Publish-subscribe	**Request-response**
Server subscribes for logs	Server accepts requests with log data
Data sources subscribe for analysis results	Data sources poll for updates
Data sources publish log data	Data sources send data by making requests
Server can publish log analysis results	Data sources poll for analysis results
One-to-many communication is supported	No one-to-many communication
Responses are not usually needed	A response is sent to every request
Possibility to use wildcards in topic-based publish-subscribe	No wildcards available
Decoupling	Coupling

uses publish-subscribe for sending analysis results to data sources for visualization.

Publish-subscribe does not include responses to published messages so it may be more efficient if there is no need for responses [87]. By contrast, request-response makes a response to every request. In the context of this thesis, it is expected that single log messages are not so important as to require responses. Since 360-degree video user logging is often done at a relatively fast pace, and as there may be multiple simultaneous users, omitting the unnecessary responses can be a significant efficiency improvement.

Many topic-based publish-subscribe implementations offer an option to use wildcards for publishing or subscribing to a topic that matches a given set of keywords [41]. Request-response does not generally have a similar concept of wildcards so, for example, polling of multiple resources may be needed instead of one collective poll. Wildcards help with one-to-many communication from a server to multiple data sources and could significantly reduce the number of messages published by the server. For example, a log server could publish a message to a topic hierarchy related to analysis results of a video 'video/videoID/viewsession/viewSessionID/analysis' using a topic with a '+' wildcard of MQTT: 'video/videoID/viewsession/+/analysis'. The message would reach all those who have subscribed to the topic with any viewSessionID so the log server would not need to have a list of all the interested subscribers and publish messages separately to each one of them. (Publication PV includes some discussion related to designing topics.)

Publish-subscribe systems are decoupled in three dimensions: time, space, and synchronization [41]. Such decoupling allows asynchronous communication, which can be considered more effective than synchronous request-response. In the context of this thesis, it means that using publish-subscribe does not require the data sources and the logging server to be available at the same time, they do not need to wait for responses before continuing execution, and the communicating parties do not need to know each other's addresses.

5.5.3 Database

Gray et al. [54] list user-defined functions, stored procedures (in other words, prepared reusable subroutines available to users of RDBMSs), declarative data analysis interfaces and concurrent visualization to be beneficial features of databases. These can be useful in 360-degree video logging and analysis systems as well as, for example, stored procedures were used in Publication PV and a declarative data analysis interface combining concurrent visualizations was used in Publication PVI. Gray et al. [54] state that moving code to data is essential for performance. It means that the code is executed near the data so that the movement of data can be minimized. Moving code to data is important because moving usually bigger data close to code that would require more resources because the size of the code is usually smaller than that of the data. Databases can provide such functionality, for example, via SQL that helps performing the queries inside the database without no a need to move big chunks of data for every query. This may be essential in some real-time 360-degree video user analysis use cases. For example, fetching 50,000 rows of log data from the database to compute the most popular tile would require more network resources than sending 20–30 rows of SQL code to the database and getting back the result that is only one row. In addition, databases have traditionally been optimized for write-intensive workloads [58]. Optimizations for write-intensiveness can be deemed beneficial in 360-degree video logging with real-time analysis requirements because the sampling rate can be relatively fast and there can be multiple simultaneous users.

The type of data needs to considered when choosing a data storage solution. 360-degree video user logs are typically structured data suitable for RDBMSs. In addition, the database schema required for the implementations in Publication PIII and PV was relatively simple so there was no need for complex relational design. While NoSQL databases are generally suitable for non-structured data, they support structured data as well [93]. From that perspective, any database with support for structured data could be used in the context of this thesis.

In the solutions presented in Publications PIII and PV, there was a need for three database tables, as can be seen in Figure 5.2. The data types used are basic types such as integers, strings, or real numbers. Designing such a schema does not require much knowledge of relational design. Joining with a complex schema can be time-consuming [75]. However, it was not an issue during the research since the schema was simple and there was no need to perform joins. Of course, some functionality could require joins. Since 360-degree video user logs can often be presented in a relatively simple numerical and text format, they are suitable for database storage. 360-degree video user logs do not typically include large binary files or complex N-dimensional data. Such data types have traditionally caused challenges with databases [54, 127]. In addition, logs are not expected to contain a lot of NULL values that could lead to a waste of disk space [75].

In this thesis it was enough to process only a moderate (around 50,000 rows) amount of data. Even though the performance testing conducted in Publication PV suggests that an RDBMS with a moderate amount of data can provide sufficient performance for many use cases, RDBMSs have challenges considering the anticipated growth in the amount of data and simultaneous users. SQL may have a poor performance reputation due to the traditional design of ACID transactions, multi-threading, and disk management in RDBMSs [135]. Explicitly, the overhead is not a due to SQL but the (R)DBMS design [135]. It is good to remember that RDBMSs do not constantly enforce strong ACID semantics and NoSQL does not completely remove the need for transactionality [94]. The disadvantages of SQL include pre-processing of data into tabular format [88] and difficulty in making advanced analysis [111]. In general, joining many tables can make SQL queries inefficient, but the efficiency also depends on the database design and analysis needs – many queries can be accomplished without joining, and queries can be optimized [119]. Despite the disadvantages, SQL remains a popular tool, and many modern platforms suitable for log analysis support SQL [88].

RDBMS is, as far as the author knows, only used in Publications PIII, PV, and by Nguyen et al. [101]. Nguyen et al. [101] use an "SQL database" for logging at

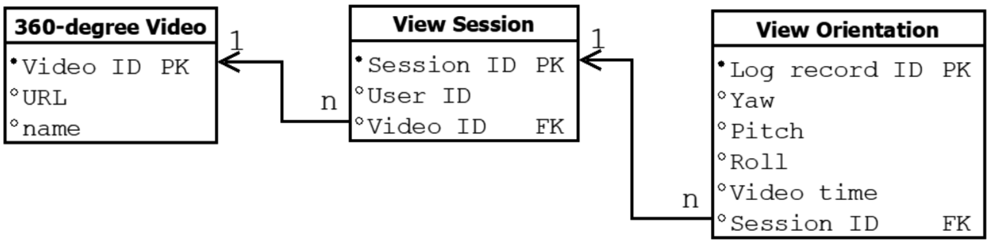

Figure 5.2 An example of relational schema for 360-degree video user logs.

least some user activities. However, it is not clear if it has been used for all logging. Many details are missing, so it is difficult to draw firm conclusions, but they seem to have a relatively advanced system. Note that this thesis does not prefer RDBMS as a database solution even though Publications PIII and PV use one.

RDBMSs traditionally support SQL well, but usage of SQL does not nowadays require a traditional RDBMS. SQL is a preferred solution for data analysis because, for example, of a common set of needed functions [5], maturity [122], portability [100], the importance of using popular standards [14], the preference for declarative queries [5], developer accessibility [5], extendibility [5], and optimization [54].

Four of the publications in Table 5.2 use a solution based on MongoDB [17, 107, 95, 108]. The missing support of SQL in MongoDB can make analysis more challenging as can be seen in the comment by Oliver et al. [107]: "For complex reporting, all relevant statements can be moved into a data warehouse to be processed later." MongoDB also works as a component of the data integration platform used in Publication PVI. The platform uses MongoDB so that it stores only the latest value, which is not useful for any time-based analysis. Luckily, the platform offers a way to integrate MongoDB with the NewSQL database CrateDB making efficient log storage and time-based analysis using SQL possible.

In general, a NoSQL database could be an effective solution [81]. Usually the performance of NoSQL databases is compared to RDBMSs if there is a relatively large amount of data [114]. However, in the context of this thesis, there may be a moderate amount of data. The performance differences between RDBMS and NoSQL may not be as big when there is only a moderate amount of data [114]. Further, if a query includes aggregates or searches to non-key values, the performance may be even weaker in NoSQL solutions [114]. Thus, the performance is partly dependent on the analysis needs, but 360-degree video user analysis may well require usage of aggregates or searches to non-key values. Even though the research conducted for this thesis entailed processing only a moderate amount of data, the author acknowledges that it is good to prepare for growing amount of data that is usually well supported in NoSQL solutions via scalability.

While a time series database could be an option, the data discussed in this thesis are not the most typical kind of time series data. Usually time in time series databases is expressed in milliseconds starting from 1970-01-01 [7]. However, in 360-degree video logging, the notion of time may come from the video time [26, 43, 84, 145, 99, 38]. In other words, the time scale starts from the beginning of the video and ends at the end of the video. For example, if a video is 10 seconds long, then the time scale can be from zero to 10,000 milliseconds. Thus, it could be possible to fit the video time into the real-world timestamp range of a time series database, although the solution would be somewhat artificial. Another characteristic feature of time series data is that they are heterogeneous unstructured data [18]. Having such data could be possible, for example, if a wide range of equipment with different measurement units were logged, but such a situation was not encountered during the study.

Still, a time series database could be a good option for 360-degree video logging since every log record can contain a real-world timestamp, each record makes a new row, and old values are basically never updated [47]. Moreover, built-in time-based partitioning, time related functions, and integrated machine learning algorithms could help with data analysis [18]. In Publication PVI, the author used CrateDB which is a NewSQL database that can be considered a time series database. It combines the easiness of SQL, real-time performance, and scalability. Such a combination makes log collection and analysis flexible and efficient.

Six of the publications in Table 5.2 do not explicitly mention having a database. There may be many reasons for not using databases. Gray et al. [54] encountered the following reasons: the cost of learning new tools, poor visualization tools, preference for using programming languages, lacking support for needed data types and access patterns, slowness, legacy applications, and a need for expensive database administrators. While some of the reasons may be still relevant, Gray et al. [54] consider that some are based on experiences with early databases that were not as mature as when they wrote their article. About 15 years has passed since their article so we can expect databases to be even better nowadays in 2023.

5.5.4 Conclusions on Architectural Comparison

This subsection concludes the section by comparing the architectures listed in Table 5.2 to the assumedly relatively efficient log server solution implemented by the author. The solution (after iterations) presented in Publication PV has a log server, a database with SQL support, and MQTT as a communication protocol. It is assumed to be relatively efficient since it has a customized lightweight log server, asynchronous low-latency communication, one-to-many communication support, and a single database (as opposed to the multiple databases in Publication PVI).

A group of publications is based on a log server solution with a NoSQL database and HTTP and WebSocket as communication protocols [17, 107, 95, 108]. One of the authors' reasons for using NoSQL is most likely that they wanted to use a lightweight open source data store called lxHive that is implemented with MongoDB. A consequence of the database selection is that they cannot use SQL because MongoDB does not support it. This can be also seen in the comment by Oliver et al. [107] about moving log data into data warehouses, that usually support SQL, for more complex analysis. However, they got the other benefits of MongoDB, such as improved performance, scalability, and schemaless data. The consequences of their communication protocol selection are that they get the benefits of request-reply, such as reliability via expressive replies, standardized return codes, general maturity, and software support of the protocol. With WebSocket they can combat against the disadvantages of HTTP by supporting real-time communication better, minimizing overheads, and enabling publish-subscribe [72]. However, they do not report if they used publish-subscribe [17, 107, 95, 108]. If they did not, they lost full decoupling and one-to-many communication.

Thus, the benefits of the solution using SQL and MQTT implemented for this thesis, compared to solutions with NoSQL and a lightweight communication protocol (but not explicitly publish-subscribe), are the following. *SQL:* better portability, usage of popular standards, better accessibility for developers, and usage of a mature technology with strong community support. *MQTT:* decoupling by design, one-to-many communication, and options provided by wildcards. Many of the same benefits are available by using publish-subscribe in general, but it is possible that not all publish-subscribe implementations support wildcards.

Nguyen et al. [101] have a log server, some client-server protocol, and an RDBMS.

The consequences of the communication protocol selection are that their traffic is likely heavier, and they lose the benefits of publish-subscribe. Duanmu et al. [38] have a log server, an unknown communication protocol, and possibly no database. It is difficult to make comparisons due to the missing information, but since they most likely do not have a database, they lose the benefits of databases, such as effective distributed data management. Qian et al. [117] use UDP protocol, but other relevant information is missing. The consequence is that they get the benefits of a connectionless protocol, such as simplicity and less used resources but reliability is compromised (that most likely does not matter in their research).

There is a relatively small amount of architectural information available in the related literature. Therefore, a proper architectural comparison is difficult. However, at least some aspects are quite clear: using publish-subscribe for 360-degree video user logging is a relatively novel idea, and log servers or databases have not been used much in the domain, despite their general advantages known in the IT field.

As a personal experience, integrating a RDBMS, a log server implemented with Node.js, and MQTT was relatively straightforward as discussed in Publications PIII and PV. The same can be said about integrating a data integration platform and MQTT in Publication PVI, but the author did not need to worry about setting up or maintaining the data integration platform, which would have been a laborious task. Setting up a data integration platform for only one special purpose would very likely be overkill. In addition, there were some restrictions in the platform that the developer just had to accept, since correcting the issues would have been too laborious in a complex system.

6 VISUALIZATION OF 360-DEGREE VIDEO USER LOGS

This chapter discusses the visualization needs of the companies in the 360VI project and reports the visualizations presented in the related literature. The chapter also presents the visualizations implemented during the research. The chapter concentrates on two different real-time visualizations – visualizations over 360-degree videos and traditional chart visualizations. By traditional chart visualizations the author means traditional charts, such as histograms, bar charts, pie charts, and line charts, that are not traditionally drawn over 360-degree video playback. The aim of the chapter is to answer the third research question (RQ3: How to efficiently visualize 360-degree video user logs?). The motivation for the visualization comes from the need to develop software for 360-degree video domain. Visualization of logs provides a way to improve software or to change the content of the video application based on user analysis. For example, it is possible to gain insights about the most popular parts of a video via visualization of logs.

RQ1 asked about the reasons of logging 360-degree video users, whereas RQ3 asked how to visualize the logs. A developer first needs a reason for logging after which it is possible to choose between the various visualization approaches. Some of the reasons for logging presented in Chapter 4 had real-time requirements. The real-time requirements link visualizations to the aspects of RQ2, which was about the efficiency of logging. The efficient architecture of RQ2 enables real-time visualization of logs collected from geographically distributed users.

6.1 User Visualization in Company Cases

The companies on the 360VI project had different analysis needs. As a reminder, the interests of the companies were the following. *A-UI:* improving user experience for a UI of a 360-degree video media library application. *B-SPORTS:* transferring

Table 6.1 Visualization approaches used to meet the analysis needs of the companies.

Company	Real-Time	Chart Visualizations	Visualizations over Video
A-UI	No	No	Yes
B-SPORTS	Yes	No	Yes
C-TRAFFIC	Yes	Yes	Yes
D-LEARNING	Yes	Yes	Yes

only the essential parts of 360-degree videos in high quality while watching sports events. *C-TRAFFIC:* analyzing if users have had the essential objects in their field of view while watching a 360-degree video in a traffic safety education application. *D-LEARNING:* various user analysis needs related to using a learning environment with 360-degree videos.

The author responded to the needs by developing visualizations. Table 6.1 categorizes and summarizes the visualization approaches used in each company case. Company E-ALGORITHMS do not appear in the table because they were not interested in user analysis. However, their interest in 360-degree video analysis algorithms led to considering combining video analysis and user analysis. By combining video analysis and user analysis it is possible, for example, to automatically recognize what objects are located in the field of view. That would, for example, help automating the user analysis of the traffic education application of company C-TRAFFIC.

6.1.1 Real-time Visualizations

This subsection elaborates the "Real-Time" column of Table 6.1. The interest of company C-TRAFFIC was in ascertaining if the user had seen the essential objects of the video, and the interest of company D-LEARNING was in analyzing the users of a learning environment with 360-degree videos. In those cases, a real-time logging and analysis system would allow fast visualization and, thus, fast response. Real-time visualizations are needed, for example, to determine what is happening currently when multiple distributed users are using the application.

Company B-SPORTS had an explicit need for a more advanced analysis when compared to other companies. For example, there was a need to compose analysis results based on multiple simultaneous users by summarization and classification, and the analysis result had more strict real-time requirements because of analyzing viewers of live streamed sports events, in a case where the analysis results have an effect on the quality of the stream. There was a need to calculate the most popular tile, for example, for viewport-adaptive streaming, that was in the interests of the company.

The mentioned real-time needs are met in the publications in the following way. First, Publication PIII presents a distributed real-time system for logging. Publication PV extends that work by presenting real-time visualization examples, and finally Publication PVI presents another, more general-purpose, distributed real-time logging and visualization system.

6.1.2 Chart Visualizations

This subsection elaborates the "Chart Visualizations" column of Table 6.1. Companies C-TRAFFIC and D-LEARNING were interested in behavioral analysis. For example, they needed to form an overall picture of the behavior of multiple users, to observe unusual behavior, and to obtain detailed information in special cases. One approach for meeting such needs is to use traditional chart visualizations that can be implemented without drawing over 360-degree videos. There was no need to present the visualizations directly to the end users over 360-degree videos so visualizations could be shown on an admin dashboard.

One need of company C-TRAFFIC was to gain an overall picture of how users behaved while using their traffic education application. The needs of company D-LEARNING were related to getting an overall picture of students' behavior while watching educational videos. Their analysis needs can be solved by producing real-time chart visualizations that allow quick feedback (using the approach in Publication PVI). Despite the need for an overall picture, there may be needs related to seeing detailed data. Luckily, chart visualizations also support a detailed view.

6.1.3 Visualizations over 360-Degree Video

This subsection elaborates the "Visualizations over Video" column of Table 6.1. In the case of company A-UI, it would be possible to see afterward how users acted while using the UI by visually tracing them. In the case of company C-TRAFFIC, it would be possible to estimate if a user had seen the essential objects of the video by visually following the user trace. In the case of company D-LEARNING, visual user tracing would help analyze users in general. In all the cases, user tracing could be needed for multiple simultaneous users or for only a single user. Publication PIII presents a visualization method over 360-degree videos in the spherical mode, that allows such visual trace analysis. The spherical mode enables to see the video immersively as if the viewer was in the middle of the 360-degree sphere as opposed to traditional mode that shows the video as if the video was a traditional video. The visualization uses the yaw, pitch, and timestamp fields of the log to draw a trace over the video without programmatic analysis.

Company B-SPORTS had also other interests than viewport-adaptive streaming. There was a need to classify users into clusters according to their view port centers. From the visualization point of view, the need was to implement visualization based on programmatic analysis results into end-user applications over 360-degree video. This entails, for example, superimposing various graphics, such as tiles, numerical analysis results, and textual information, over the most popular areas of the video. Although only company B-SPORTS had an explicit interest in analysis related to the most popular tiles and view point clusters, such an analysis could also be beneficial for the other companies. An approach to create such a visualization is discussed in Publication PV. Similarly to the user trace visualization of Publication PIII, the visualization of Publication PV uses the yaw, pitch, and timestamp fields of the log to calculate the most popular tiles and for the cluster analysis.

In the case of companies C-TRAFFIC and D-LEARNING, there was a need to show visualizations in the spherical mode. With such an approach it would be possible to show the visualization for the end users themselves using the same application the user had used while being logged, and they could learn from their previous viewing sessions. Using the same application is important so that the user need not to take off the HMD used while logging to see the visualization on a different device. In the case of company A-UI, visualizations over 360-degree videos in the spherical mode are not as essential since user traces are meant only for the developers so the traces can be visualized over 360-degree videos in the rectangular mode. However, visualizations over 360-degree videos in the spherical mode were an adequate solution for the research prototype.

Table 6.2 Visualization approaches in the related literature.

Publication	Real-Time	Chart Visualization	Visualizations over Video
[27]	No	No	No
[108]	No	No	No
[9]	No	Yes	No
[26]	No	Yes	No
[136]	No	Yes	No
[145]	No	Yes	Yes
[38]	No	Yes	Yes
[99]	No	Yes	Yes
[85]	No	Yes	Yes
[49]	No	Yes	Yes
[117]	Yes	Yes	No
[17]	Yes	Yes	No
[107]	Yes	Yes	No
[95]	Yes	Yes	No
[86]	Yes	Yes	No
[101]	Yes	Yes	Yes

6.2 Visualizations in Literature

This section discusses the visualizations presented in the related scientific literature. The publications are selected from Table 4.1 (page 56) listing the reasons for logging users in 360-degree video domain. Many of the publications in Table 4.1 do not explicitly discuss user analysis and have been excluded from this section. The publications included can be seen in Table 6.2. It is good to note that Table 4.1 may include multiple instances of the same publication, whereas Table 6.2 does not, which may make it appear relatively short.

Many of the publications discuss a system with real-time requirements. However, in many cases it is not entirely clear if visualizations are possible in real-time. Hence most of the publications have "No" in the "Real-Time" column. Some publications mention a wish for real-time analysis or present interactive programmatically generated traditional charts [17, 107, 95, 85, 86]. Nearly all the publications include traditional chart visualizations, but in most of the cases the visualizations are not generated automatically, but only for the figures of the publication. As a special detail, Wu et al. [145] made a 3D chart visualization.

A publication has "Yes" in the "Visualizations over Video" column if the publication contains a figure of a visualization over a 360-degree video. As can be seen, a few publications present visualizations over 360-degree videos, but only two of them present visualizations in the spherical playback mode [85, 101]. Again, in most of the cases, it seems that the visualizations were simply made up for the publication rather than presenting working real-time software. Some publications present graphics over 360-degree videos, but these are not visualizations made for user analysis [17, 107].

Table 6.2 suggests that there are not many real-time visualizations over 360-degree videos in the related literature. Only one publication ([101]) is considered to have such a visualization, and thus it is especially interesting from the viewpoint of this thesis. However, traditional chart visualizations are clearly much used, and there are many cases of non-real-time visualizations over 360-degree videos. The shortage in the presence of real-time visualizations over 360-degree videos, for its part, emphasizes the novelty of the work accomplished for this thesis.

6.3 Visualizations Implemented

This section presents the visualizations implemented during the research work. The selection of each visualization approach is justified and possible alternatives are discussed.

6.3.1 Visualizations over 360-Degree Video

Visualizations over 360-degree videos were implemented due to company needs. This is an illustrative and immersive visualization method.

User tracing was implemented because it allows a detailed inspection of user behavior. It is especially useful in situations where the analysis is difficult to automate, for example, if the analyst does not know beforehand what to look for. User tracing visualizations can be implemented in various ways. The author's solution to user tracing was to visualize the center point of a logged user's viewport over 360-degree

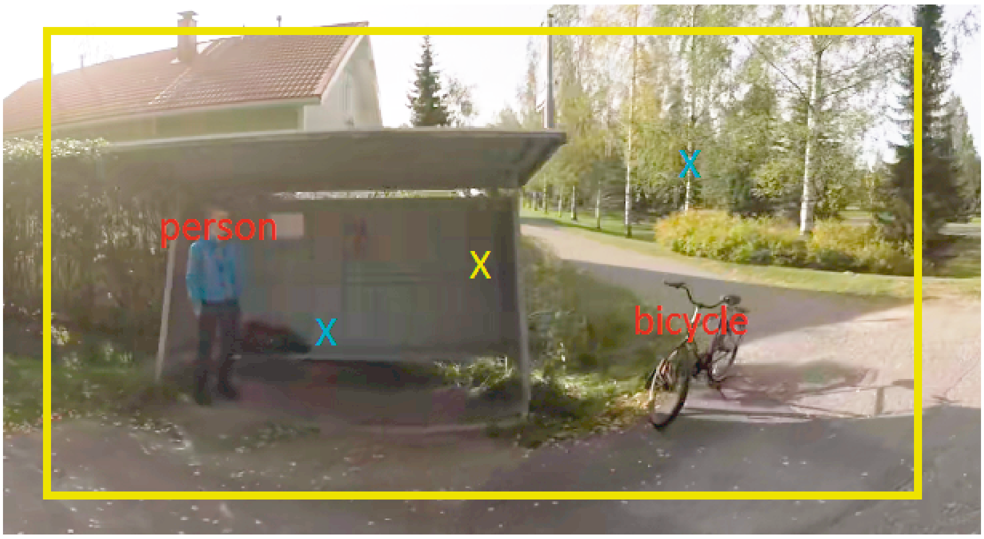

Figure 6.1 An example of user traces marked with X showing the view orientations of previous view sessions. (Object detection metadata annotations are also visible.)

video in the spherical playback mode. The author drew user traces over videos without connecting the dot sequences, which differs from the way in which scanpaths are usually implemented, but showing only the closest view orientation in relation to current video time. An example can be seen in Figure 6.1. The reason for not visually connecting dot sequences was that our videos were recorded from the viewpoint of a moving object and showing connected scanpaths would have been confusing when the video content changes fast. Visible paths connecting the dots can be more useful if the video is recorded by a camera that stays in place.

Visual traces are not always needed for studying user behavior in 360-degree videos, but it is possible in some cases to automate the analysis. For example, the solution to the need to analyze the objects seen in a traffic video by company C-TRAFFIC is somewhat trivial with the architecture used – if there are log data with user ids, it is possible to programmatically analyze the user's actions based on the log. For example, it is easy to check that the viewport center of the user was close enough to yaw coordinate 10° and pitch coordinate 10°, where the object of interest is located at the time of one second. A more interesting automation task would be, for example, to identify the essential objects in the video using video analysis algorithms and to combine that information with user logs. However, this is only briefly discussed in Publication PIII.

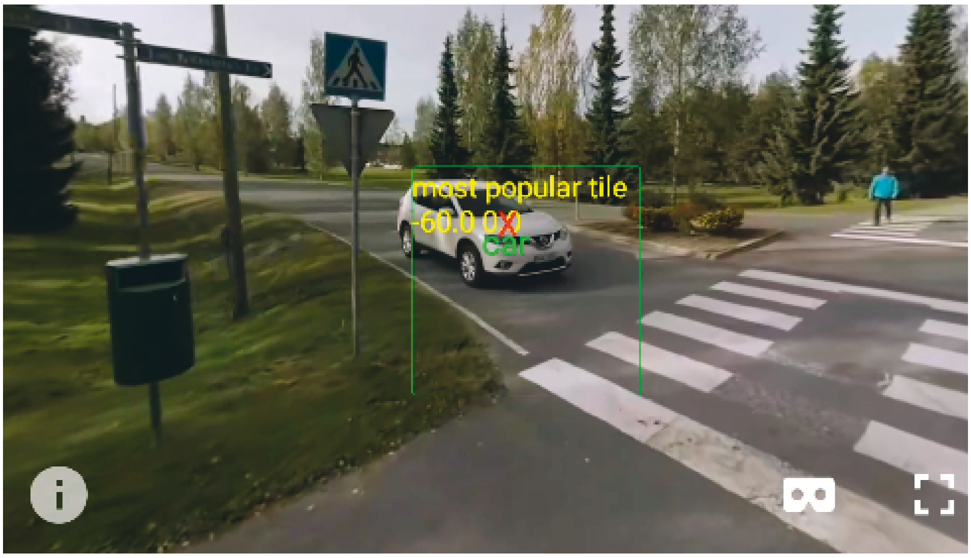

Figure 6.2 An example of the most popular tile visualization.

In Publication PV, the author drew simple tiles over the video to represent analysis results as seen in Figure 6.2. The numbers in the figure show the yaw and the pitch angle of the tile. The computation is executed by the analysis component of the logging and analysis architecture, and the results are published via MQTT to multiple clients for real-time visualization.

In contrast to Publication PIII (Figure 6.1), where the data for the user traces were queried using a single database query, in Publication PV the database needed to be queried continuously at short intervals due to the stricter real-time needs as the log data were collected continuously and were thus continuously changing. The visualizations therefore needed to be updated accordingly. In Publication PIII it was possible to query trace data with a single query because it sufficed to show traces of previous viewing sessions that were already over and no longer changing. Continuous querying increases the load of the database and the amount of network traffic. However, MQTT helped by delivering the analysis results to all interested subscribers in one-to-many style so that not all of them needed to do make continuous separate requests. Further, database load can be reduced by limiting the amount of processed data if the results stay accurate enough with a smaller amount of data.

SQL is not considered an efficient and feasible language for advanced analysis [111]. However, it is possible to implement an efficient K-means clustering with SQL [110], and there is an example of implementing K-means with SQL [96]. K-means clustering was adapted in this study from to the cited example. With the algorithm it was possible to experiment the user clustering interests of the company B-SPORTS. The cluster centers were visualized using the same method as user traces in Publication PIII, and their location was calculated once a second, which required continuous database communication.

6.3.2 Traditional Chart Visualizations

Traditional chart visualizations were implemented because they support some of the company cases and are often relatively easy to implement and understand. Visualization can be implemented with the help of SQL, as presented in Publications PV and PVI. Figure 6.3 shows a timeline graph based on an SQL query. The graph is a part of a web dashboard consisting of multiple graphs. The graph can be updated continuously and allows user interaction. It was implemented using a popular web dashboard tool Grafana. Other similar dashboard tools could have been used, but Grafana was chosen due to its popularity and because it can be integrated with CrateDB used by the data integration platform that was tried out as a log server.

By default, Grafana has a visualization tool named Graph for making traditional chart visualizations. Graph makes it necessary to use real-world timestamps on x-axis, so it is mostly useful for real-world time related analysis needs as opposed to video timestamp related analysis if video timestamps have been stored, for example, as integers. Such needs include analysis of live stream video watchers or analysis that is only related to the user's actions in real-world time. For example, with Graph it is possible to see that the user's yaw and pitch are close to 0° at some point in time, but it is be more difficult to see at what time in the video that happened. Since Graph visualization is suitable for user analysis of live stream videos, it supports the analysis needs of company B-SPORTS. The visualization in Figure 6.3 was implemented with Plotly plugin by Grafana. This is better than Graph in that it allows more data types on the x-axis. It allows see where the users were looking at a given second in video time.

Besides view orientations, there may be other visualization needs. One example is the need to visualize the most watched seconds of a video. Graph visualization by Grafana has a histogram mode, with which it is possible to show the most watched seconds as bars of different heights. This can be seen in Figure 6.4. While such a

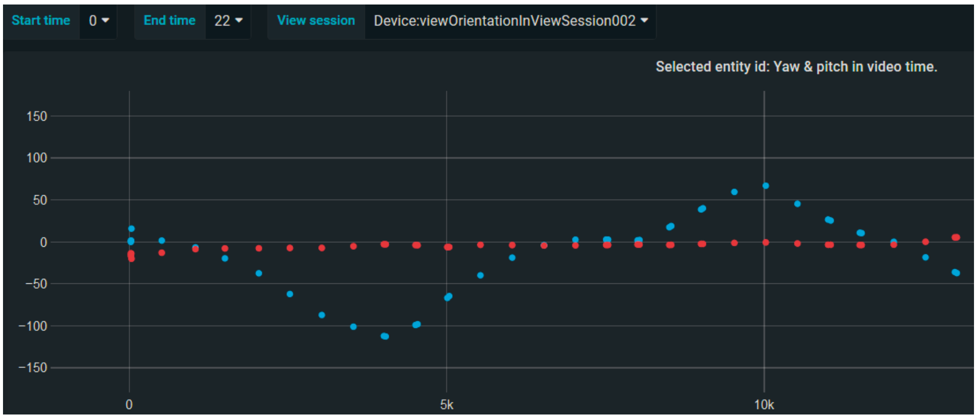

Figure 6.3 Example of a view orientation graph based on an SQL query.

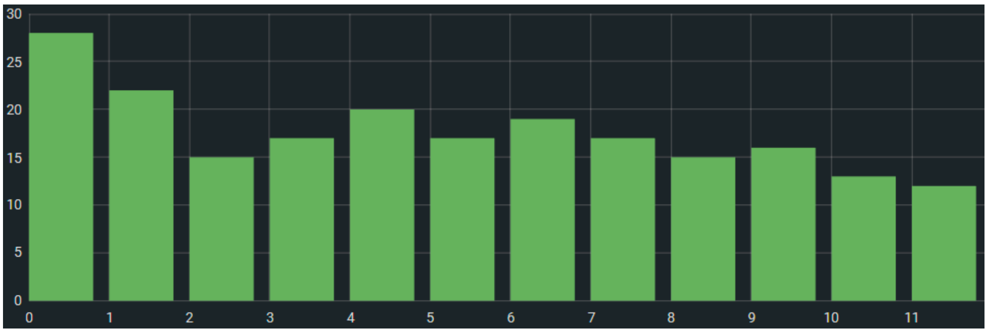

Figure 6.4 Example of a histogram showing the number of watchers for every second of the video.

visualization was not explicitly needed by the companies, it could provide them all with helpful related insights into the users' behavior.

Grafana also has some presumably rarely used polar visualizations, such as radar and wind rose visualizations, but the implementations were so buggy the author was unable to make proper use of them. However, such polar visualizations could be useful for 360-degree video related analysis since the spherical space fits well into polar view.

Since the traditional chart visualizations made for this thesis were not intended for end users, there was no need to query data from the database by the user front ends. Queries are only needed for the admin dashboard, which reduces the amount of network traffic because there are usually fewer admins than end users. Moreover, since there is no need to visualize data over videos, there is no need to synchronize visualizations in relation to video playback time. Nor is there any need to estimate what is an adequate pace for updating the visualizations over videos.

6.3.3 Comparison of Visualizations over Videos and Traditional Visualizations

As presented in this chapter, the author found two approaches for visually analyzing 360-degree video user logs. The first approach is by drawing visualizations over 360-degree videos, and the second approach is by using traditional chart visualizations. The chapter concludes with a comparison of the two approaches. In general, the usefulness of an analysis approach is related to the analysis needs.

Visualizations over 360-degree can directly illustrate video footage related details, that are not as easily visualized with traditional chart visualizations. Likewise, traditional charts are good for presenting overall information, that is not as easily visualized over 360-degree videos. In addition, creating traditional charts can be relatively easy with the available tool support, and it is possible for a viewer to follow a few real-time chart visualizations simultaneously. As a comparison, watching two 360-degree videos with visualizations simultaneously with a HMD would most likely be a disturbing experience.

In both the approaches, visualizations can be confusing if too much data is displayed at once. Thus, in general, both can benefit from approximation and summarization. Tile-based approximation can be used with both the approaches but a need for realistic and accurate visualization of tiles over 360-degree videos may increase the development difficulty. Tile-based analysis with chart visualizations also requires further investigation and it was not experimented with in this thesis work. In both the approaches, efficient usage of tiles may require database-level decisions, such as whether to preprocess and save tile-related information to a database already in the logging phase, or deal with more complex and performance-intensive but flexible database queries when fetching the data for display.

When drawing visualizations over 360-degree videos with a custom solution, much of the user experience development may need to be done from the scratch. In contrast, at least some basic usability features can be ready-made when implementing traditional charts with popular tools, such as Grafana.

7 RELATED WORK

This chapter presents the related work. The aspects included are public user logging datasets, real-time analysis of 360-degree video logging and analysis in VR and AR, and distributed user logging architectures.

As a brief introduction to the related work, Rong et al. [123] conducted a systematic review of logging practice in software engineering. Nine development contexts for logging were identified: fault tolerance, security, open source projects, distributed systems, complex systems, concurrent systems, embedded systems, multicore systems, and network [123]. Unfortunately, publications on user behavior logging, log analysis, and usage of logs were excluded [123]. A noteworthy finding was that logging practices lack formal specifications and systematic design [123]. The study is relevant in the context of this thesis because their findings are applicable to 360-degree video user logging. In addition, the logging development context "distributed systems" is also relevant for this thesis. Of course, a systematic review of *user logging* would have been a more relevant study, but the author is not aware of such studies. However, user analysis is possible without logging as can be seen in the studies of 360-video users conducted with questionnaires and interviews [44, 86].

7.1 Public User Logging Datasets

Several public user logging datasets for 360-degree videos have been published recently [84, 26, 145, 99, 38, 49]. In addition to publishing open data, the studies present examples of how to do analysis using their datasets. The author of this thesis considers the studies to be relevant work even though public datasets were not published in the studies conducted for this thesis.

Lo et al. [84] used open source user tracking components for logging and analyzed user logs by making saliency and motion maps. They logged yaw, pitch, roll, timestamp, and viewer position (x-, y-, and z-coordinates) [84]. Compared to the data logged by the author of this thesis, they have additionally the viewer position, which was not available when using Google VR SDK. Their log dataset can be used for optimizing existing 360-degree video streaming applications and novel applications [84]. For more detailed examples, their dataset is useful for predicting the attractive parts of videos, computing the most common field of views, developing interfaces, and identifying the essential elements for gaining viewers' attention [84].

Corbillon et al. [26] used an open source VR platform for logging and created example statistics about users' navigation patterns. They logged timestamp, video frame number, and user's head position in quaternion format [26]. Compared to the data logged by the author of this thesis, they used quaternion format, whereas this thesis used yaw, pitch, and roll format, and the logged video frame number in Publication III was only an approximation. The dataset can be used for studying and understanding 360-degree video consumption [26]. Corbillon et al. [26] think that "The prediction of navigation patterns is a cornerstone of the new generation viewport-adaptive streaming systems for 360-degree content".

Wu et al. [145] used Unity for logging users to explore their behavior and present related visualizations and statistics. They logged local timestamp, video playback time, view orientation in quaternion format, and viewer position (x-, y-, and z-coordinates) in Unity space [145]. Compared to the data logged by the author of this thesis, they included local timestamp and viewer position, and used quaternion format. With the local timestamp they have the option to analyze data as a function of local time. For example, they can analyze how a video was watched on January 1, 2022 at 3 p.m. With the viewer position in Unity space they can propbably do some Unity-specific analysis, or use the information for replay. By using quaternion format, they get some of the general benefits of quaternions, such as protection for gimbal lock. Their dataset can be useful for exploring user behavior patterns [145].

Nasrabadi et al. [99] used Unity for logging and analyzed logs by making heatmaps and studying viewport patterns with a clustering algorithm. They logged timestamp, head orientation in quaternion format, and vector from the video sphere center to a viewport center (x, y, z) [99]. Compared to the data logged by the author of this thesis, they included the viewport center vector and used quaternion format. Their reason for logging users was to analyze viewport patterns [99].

Duanmu et al. [38] made a dataset by logging computer users instead of HMD users by using a local server. They logged timestamp, yaw, and pitch and analyzed data with heatmaps and trajectory analysis [38]. Their data seems to be relatively similar to the data logged by the author of this thesis. Their reason for logging users is to explore the viewer behavior when watching 360-degree videos to see how it correlates with the video content [38].

Fremerey et al. [49] developed a tool for the logging and statistical analysis of head orientation data that they logged in yaw, pitch, and roll format. The logged data can be used, for example, as part of a viewport-adaptive streaming solution [49].

All these studies are related to this thesis because they are clear examples of 360-degree video logging and include the reasons for doing it. However, none of them discuss logging with distributed architecture or real-time visualizations.

7.2 Real-Time Analysis

Some studies address real-time user analysis [9, 117, 43, 136, 98, 27]. They may include some architecture related discussion, but usually the essential details from the viewpoint of this thesis are missing, hence comparing them to the work accomplished for this thesis is relatively difficult. Real-time user analysis is not always explicit in them, and therefore why some of them have been categorized to be without real-time analysis in Table 6.2 (page 81). However, even if the paper does not explicitly discuss real-time analysis, the solution proposed should be applicable to real-time usage. They are therefore discussed in this "Real-Time Analysis" subsection. This subsection is written more from the viewpoint of logging architectures, whereas Table 6.2 concentrates on the visualization capabilities.

Bao et al. [9] predict 360-degree video users based on user logs. They estimate that their method for motion prediction is feasible within 100–500 ms timeframe, but they do not provide many logging related details [9].

Qian et al. [117] studied optimizing 360-degree video delivery over cellular networks with the help of user tracking. They used open source user tracking components and sent the sensor readings from HMD to a laptop using UDP over Wifi for short latency [117]. While they have some kind of network, it is described only briefly and simultaneous logging of multiple users is not discussed [117].

Fan et al. [43] studied real-time view orientation prediction using neural networks with an environment consisting of open source user tracking components. The study does not explicitly mention a distributed architecture with a log server [43]. Sun et al. [136] also predict field of view with linear regression, but many logging details are missing. Nasrabadi et al. [98] figure an architecture for viewport-adaptive 360-degree video streaming without providing many logging related details.

Corbillon et al. [27] present a viewport-adaptive 360-degree video streaming system where an HMD client sends the current head orientation to a server requesting new video segments. However, they do not log users since they use a ready-made dataset [27].

7.3 Logging and Analysis in VR and AR

Given the paucity of detailed publications on 360-degree video logging, the author lists some user logging and analysis examples from VR and AR domains. By and large, these studies may include logging activities similar to those addressed in this thesis but there is no real-time analysis, some relevant logging details may be missing, or the architecture is non-distributed.

Eishita et al. [40] logged players' in-game performance using Unity. Orientation data were logged for every drawn frame [40]. The study includes some statistical log analysis [40].

Gandrud and Interrante [51] logged head and eye orientations for predicting destination during movement of HMD. The study does not report real-time analysis, but includes statistical log analysis [51].

Hodgson and Bachmann [61] logged users to study the challenges of movement in VR because of the limitations of real world (i.e., redirected walking). The logged data includes timestamp, user's position, and view orientation [61]. A statistical log analysis is included [61].

Koulieris et al. [76] studied users' accommodation and comfort with HMDs with a special equipment that measures vision problems. Log data were captured with the help of Unity and stored into an SQL database, but many logging details are missing [76]. A statistical analysis is included [76].

IPSViz by Raij and Lok [118] is an example of VR system with diverse logging features. It can log various inputs, such as head, hand, and body movements in addition to capturing speech and video [118]. It can produce 3D visualizations and scalable timelines with social information, such as verbal and non-verbal communication [118]. Despite the logging features of the system, the results of the study are based on interviews [118].

7.4 Distributed User Logging Architectures

A group of publications by Bibiloni et al., Oliver et al., and Molino et al. [17, 107, 95, 108] discuss a distributed 360-degree video user logging and analysis solution implemented with a NoSQL database, that supports visualizations in a web UI using a data store called lxHive. The solutions presented in this thesis differ from their solution; they use a NoSQL database, WebSocket, and HTTP, whereas the author of this thesis implemented one solution with MQTT and an RDBMS and another with MQTT and a combination of a NoSQL database and a time-series database. The similarities of this thesis and their work include distributed architecture and a support for real-time analysis. Thus, they have also understood the importance of distribution, databases and lightweight communication, but they do not justify their implementation of a distributed architecture using the techniques mentioned.

If lxHive had been implemented with an RDBMS, the author of this thesis can see no particular reason why the mentioned researchers ([17, 107, 95, 108]) could not use it and benefit from SQL support. They would probably lose some of their scalability and performance, but it is difficult to say how important these features were for them. MQTT could be suitable for them because they present no particular reasons for choosing HTTP or WebSocket. However, changing HTTP (and WebSocket) to MQTT would naturally need some re-factoring, as proposed in Publication PIV.

Nguyen et al. [101] built a distributed collaboration system for 360-degree videos, which includes a client-server architecture and an SQL database. The solutions presented in this thesis differ from their solution as they do not use MQTT and thus lose the benefits of MQTT. In general, the solutions presented in this thesis are in line with their solution since the components used seem to be somewhat similar. By using SQL, they obviously get the general benefits of SQL, but they offer no particular reasons for choosing a client-server protocol, an SQL database, or a distributed architecture.

All the publications in this subsection have been discussed in detail in the architectural comparison of Sections 5.4 and 5.5. However, they are also discussed in this chapter since they are probably the most relevant related work available.

Since publications on logging 360-degree video users that also discuss distributed architectures are relatively rare, some examples from VR, AR, and Mixed Reality (MR) domains are introduced next. For example, Brown et al. [21] present a detailed architecture of an AR combat simulator, where log data is transferred via a network. However, it seems to be a local network, and the authors present no real-time analysis [21].

A study by Hagiwara [57] on multi-modal VR data collection discusses a cloud-based solution. However, the architecture or logging details are not well explained [57].

Kobayashi et al. [74] logged humanoid robots with the help of VR technology. The paper includes a detailed explanation of a distributed logging architecture, where log data are transferred over a local network [74]. The publication presents some real-time features but no examples of real-time analysis [74].

Okuma et al. [105] logged user activity in a MR museum and stored the data in an RDBMS for behavioral analysis. However, it seems that the system does not have real-time requirements, and many logging details are missing [105].

8 RESULTS AND DISCUSSION

This chapter revisits the research questions and summarizes the answers to them. There is also a section for discussing the limitations of the thesis and ideas for future work.

8.1 Logging of 360-degree Video Usage

Chapter 4 contains the answer to RQ1: what is 360-degree video user logging and why do it? The first part of the question in answered by studying the scientific literature. The answer is summarized by forming the definition for user logging in 360-degree video domain. The second part is answered by categorizing example reasons for logging found in the scientific literature and received from the companies involved in the 360VI project. The most popular reasons for logging and analyzing 360-degree video users seem to be "viewport-adaptive streaming, transmission optimization, and saving bandwidth" and "exploring and predicting user behavior".

8.2 Efficient Distributed Logging and Analysis

Chapter 5 aims to answer RQ2: how to support 360-degree video user logging via development of efficient architecture? The answer presents a reference architecture and example implementations based on it. The answer compares the example implementations to those found in the related literature and enumeratates the benefits of the reference architecture.

A reference architecture is designed to fulfill the requirements derived from the reasons for logging 360-degree video users (which is a part of the answer of RQ1). The reference architecture is designed as a logging architecture where log data are sent via a network to a log server connected to a database enabling efficient data storage and analysis. A distributed architecture makes logging and analysis of distributed users possible and is beneficial in the development work, for example, because of supporting re-use and changing requirements.

Example implementations of 360-degree video logging and analysis systems are presented in Publications PIII, PV, and PVI. The example implementations are compared to those found in the related scientific publications from the perspective of distribution, databases, and network protocols. The comparison suggests that there are not many architecturally detailed publications available and even though distribution, database usage, and publish-subscribe communication could provide benefits, these are not widely used in the related research.

Publication PIII uses HTTP for sending user log records to the back end. Due to the limitations of HTTP, Publication PIV studies how to replace HTTP with MQTT in IoT setting, and Publication PV replaces HTTP with MQTT in a 360-degree video logging system. Publication PVI also presents a 360-degree video logging system that uses MQTT as a communication protocol. The author suggests that the efficiency of 360-degree video user logging and analysis architecture can improved by using a lightweight publish-subscribe protocol such as MQTT. The author of this thesis is not aware of publish-subscribe protocols being used for 360-degree video user logging by other researchers.

Despite their general advantages, the comparison conducted suggests databases are not widely used in scientific 360-degree video logging and analysis context so the thesis encourages their use. The author concentrated more on RDBMSs during the study, but they do not meet every need. Other database solutions can be useful as well. Publications PIII and PV had an RDBMS, while Publication PVI had a combination of a NoSQL database and a NewSQL database. Further, in addition to suggesting using databases, the author suggests using SQL for analysis, for example, because of better portability, usage of popular standards, better developer accessibility, and maturity.

8.3 Visualization of 360-degree User Logs

Chapter 6 aims to answer RQ3: how to efficiently visualize 360-degree video user logs? The question is answered in three parts. The first part discusses how the visualizations implemented meet the needs of the companies involved in the 360VI project. The second part compares the visualization solutions implemented to those found in the related literature. The third part presents the visualizations implemented during the study with the emphasis on describing the technical implementation, not the visual details.

The visualization approaches chosen in the company cases include real-time chart visualizations and visualizations over 360-degree videos. In the related literature, the need for real-time analysis is not explicit but chart visualizations are popular and there are some examples of visualizations over 360-degree videos.

The visualizations over 360-degree videos developed include a user trace visualization and a most popular tile visualization. The chart visualizations developed include a view orientation graph and a histogram showing the number of watchers of every second in the video. What distinguishes the visualizations developed from most of the related research is that they work in real time and visualizations over 360-degree videos in the spherical mode are not very common.

8.4 Discussion

While the thesis does not discuss big data, it is good to prepare for the growing amount data. Analysis can be started with a small amount of log data so that it is easier to prepare for the future of 10x–1000x log data [24]. In this thesis, simultaneous logging and analysis was tested with a few users at a time without unexpected performance problems. Naturally, there is a limit at which the performance starts to suffer, but the author expects that the solutions proposed (Publications PV and PVI) should cope at least with tens of simultaneous users and that these solutions can be optimized. In addition, more quantitative data on the effectiveness of different databases and data processing algorithms would be useful. That would help to decide when there is a need for big data solutions. An interesting question is when to move from databases to big data processing platforms. Big data processing platforms with MQTT support could provide feasible solutions for the expected increase in users and data.

Some of the discussion about MQTT is rather outdated since the work was done using version 3.1.1, and 5.0 has already been published. For example, 5.0 provides better support for request-response. It was not used during the study because the MQTT implementations used did not support it during the hands-on research. Still, using MQTT in the context of 360-degree video user logging and analysis is an innovative approach. Some arguments have been put forward for using other lightweight protocols, such as Constrained Application Protocol (CoAP), instead of MQTT. CoAP supports both request-response and publish-subscribe, thus it would be efficient in many situations [71]. However, while CoAP tries to be something between HTTP and MQTT, it fails in adaptation and support [143]. In addition, secure one-to-many communication is not well supported by CoAP [71, 33]. Moreover, distributed publish-subscribe solutions could have been investigated.

Advanced user analysis was left with little attention. Publication PV discusses an implementation of K-means clustering, but implementing and comparing other algorithms would have been interesting as well as thinking about use cases for them. Still, the implementation of K-means serves as an advanced data mining example.

It would be interesting to study what kind of visualizations over 360-degree videos are the most useful. The author only used simple tiles and texts, but it is expected there are more user-friendly visualizations for different purposes. This also applies to traditional charts, which are not drawn over 360-degree videos. While Publication PVI studied traditional charts to some extent, a more thorough comparison of different kinds of charts could be beneficial, and the practical work would not be very challenging because implementing traditional charts is relatively easy.

The implementations carried out for this thesis used only non-streamed local videos. However, streamed videos are important for many applications, so it would be interesting to study logging and analysis while using streamed videos. This could be done, for example, in the context of viewport-adaptive streaming.

9 CONCLUSIONS

This doctoral thesis discusses aspects related to logging and analyzing 360-degree video users. It comprises on the basis of six publications. The introductory part is not only an overview of the publications, but also presents further findings and discussion (as discussed in Section 3.4). The challenges of logging and real-time analysis of 360-degree video users are related to the continuously updating video frame and geographically distributed users that require an efficient distributed system. In addition, 360-degree video user logging is a relatively new domain where good practices have not yet evolved.

The work presents a reference architecture for efficient distributed logging and analysis of 360-degree video users. The reference architecture is based on a log server with a database. While server architectures and databases are not novel solutions in general, they are not well adopted in the domain. Part of the technical novelty of this thesis comes from the usage of a publish-subscribe protocol for transferring 360-degree video user logs and log analysis results in a network.

In addition to practical work, questions like "what is 360-degree video user logging, and why do it" were investigated. By studying scientific publications, a definition for 360-degree video user logging was formulated, and the author suggests that the most popular reasons for logging are related to "viewport-adaptive streaming, transmission optimization and saving bandwidth" and "exploring and predicting user behavior".

Further, as many of the benefits of logging are gained via log analysis, the thesis presents real-time visualizations for analyzing 360-degree video users using two approaches. The first approach is by drawing visualizations over 360-degree videos and the second approach is by making chart visualizations.

The main contributions of this thesis are the following:

- The first definition of user logging in 360-degree video domain, and the first categorization of the reasons of 360-degree user logging.

- A suggested reference architecture for distributed 360-degree video logging and analysis. This includes a novel publish-subscribe approach in the 360-degree video user logging and analysis context discussed.
- Implemented real-time visualizations of 360-degree user logs using two approaches. Visualizations were implemented by drawing over 360-degree videos and by making traditional charts. The visualizations are enabled by the used architecture and the used technical solutions.

REFERENCES

[1] Patrice Rondao Alface, Jean-François Macq, and Nico Verzijp. "Evaluation of bandwidth performance for interactive spherical Video". In: *Multimedia and Expo (ICME), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–6.

[2] A.R. Al-Ali, Imran A. Zualkernan, Mohammed Rashid, Ragini Gupta, and Mazin AliKarar. "A smart home energy management system using IoT and big data analytics approach". In: *IEEE Transactions on Consumer Electronics* 63.4 (2017), pp. 426–434.

[3] Mathias Almquist, Viktor Almquist, Vengatanathan Krishnamoorthi, Niklas Carlsson, and Derek Eager. "The prefetch aggressiveness tradeoff in 360 video streaming". In: *Proceedings of the 9th ACM Multimedia Systems Conference*. 2018, pp. 258–269.

[4] Vasileios Anastopoulos and Sokratis K Katsikas. "A methodology for building a log management infrastructure". In: *2014 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE. 2014, pp. 000301–000306.

[5] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. "Spark sql: Relational data processing in spark". In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 2015, pp. 1383–1394.

[6] Ronald T Azuma. "A survey of augmented reality". In: *Presence: Teleoperators & Virtual Environments* 6.4 (1997), pp. 355–385.

[7] Andreas Bader, Oliver Kopp, and Michael Falkenthal. "Survey and comparison of open source time series databases". In: *Datenbanksysteme für Business, Technologie und Web (BTW 2017) - Workshopband* (2017), pp. 249–268.

[8] Andrew Banks and Rahul Gupta. *MQTT Version 3.1.1.* OASIS standard. Oct. 2014.

[9] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos". In: *2016 IEEE International Conference on Big Data (Big Data).* IEEE. 2016, pp. 1161–1170.

[10] Giuseppe Barbieri and Augusto Celentano. *Multimedia technology: a companion to art visitors*. IGI Global, 2011, pp. 393–410.

[11] Titus Barik, Robert DeLine, Steven Drucker, and Danyel Fisher. "The bones of the system: A case study of logging and telemetry at microsoft". In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C).* IEEE. 2016, pp. 92–101.

[12] Richard Baskerville. "What design science is not". In: *European Journal of Information Systems* 17.5 (2008), pp. 441–443. DOI: 10.1057/ejis.2008.45.

[13] Cüneyt Bayılmış, M Ali Ebleme, Ünal Çavuşoğlu, Kerem Küçük, and Abdullah Sevin. "A survey on communication protocols and performance evaluations for Internet of Things". In: *Digital Communications and Networks* 8.6 (2022), pp. 1094–1104. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2022.03.013.

[14] Edmon Begoli and James Horey. "Design principles for effective knowledge discovery from big data". In: *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*. IEEE. 2012, pp. 215–218.

[15] Paolo Bellavista and Alessandro Zanni. "Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP". In: *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*. IEEE. 2016, pp. 1–6.

[16] Jennifer Romano Bergstrom and Andrew Schall. *Eye tracking in user experience design*. Elsevier, 2014, pp. 1–374.

[17] Toni Bibiloni, Antoni Oliver, and Javier del Molino. "Automatic collection of user behavior in 360 multimedia". In: *Multimedia Tools and Applications* 77.16 (2018), pp. 20597–20614.

[18] Ledion Bitincka, Archana Ganapathi, Stephen Sorkin, Steve Zhang, et al. “Optimizing Data Analysis with a Semi-structured Time Series Database”. In: *SLAML* 10 (2010), pp. 1–9.

[19] Jose Borges and Mark Levene. “Evaluating Variable-Length Markov Chain Models for Analysis of User Web Navigation Sessions”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.4 (2007), pp. 441–452. DOI: 10.1109/TKDE.2007.1012.

[20] Abbie Brown and Tim Green. “Virtual reality: Low-cost tools and resources for the classroom”. In: *TechTrends* 60.5 (2016), pp. 517–519.

[21] Dennis G Brown, Joseph T Coyne, and Roy Stripling. “Augmented reality for urban skills training”. In: *IEEE Virtual Reality Conference (VR 2006)*. IEEE. 2006, pp. 249–252.

[22] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamas Sarlos. “Are web users really markovian?” In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 609–618.

[23] Anton Chuvakin and Gunnar Peterson. “How to do application logging right”. In: *IEEE Security & Privacy* 8.4 (2010), pp. 82–85.

[24] Anton Chuvakin, Kevin Schmidt, and Chris Phillips. *Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management*. Newnes, 2012.

[25] Xavier Corbillon. *Enable the next generation of interactive video streaming*. Doctoral Dissertation. Ecole nationale supérieure Mines-Télécom Atlantique, 2018.

[26] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. “360-Degree Video Head Movement Dataset”. In: *Proceedings of the 8th ACM on Multimedia Systems Conference*. EPFL-CONF-227447. ACM. 2017, pp. 199–204.

[27] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. “Viewport-adaptive navigable 360-degree video delivery”. In: *2017 IEEE international conference on communications (ICC)*. IEEE. 2017, pp. 1–7.

[28] Maxime Cordeil, Tim Dwyer, Karsten Klein, Bireswar Laha, Kim Marriott, and Bruce H Thomas. "Immersive collaborative analysis of network connectivity: Cave-style or head-mounted display?" In: *IEEE transactions on visualization and computer graphics* 23.1 (2016), pp. 441–450.

[29] Jules S Damji, Brooke Wenig, Tathagata Das, and Denny Lee. *Learning spark: lightning-fast big data analysis*. 2nd Edition. " O'Reilly Media, Inc.", 2020.

[30] Erwan J David, Jesús Gutiérrez, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. "A dataset of head and eye movements for 360 videos". In: *Proceedings of the 9th ACM Multimedia Systems Conference*. 2018, pp. 432–437.

[31] Monika Dávideková and Michal Gregu Ml. "Software Application Logging: Aspects to Consider by Implementing Knowledge Management". In: *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE. 2016, pp. 102–107.

[32] *DBMS popularity broken down by database model - Number of systems per category, February 2019*. https://db-engines.com/en/ranking_categories. Online; accessed 23-April-2019. 2019.

[33] Niccolò De Caro, Walter Colitti, Kris Steenhaut, Giuseppe Mangino, and Gianluca Reali. "Comparison of two lightweight protocols for smartphone-based sensing". In: *2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*. IEEE. 2013, pp. 1–6.

[34] *Definition of log*. https://www.merriam-webster.com/dictionary/log. Online; accessed 23-April-2019. 2019.

[35] *Definition of log*. https://en.oxforddictionaries.com/definition/log. Online; accessed 23-April-2019. 2019.

[36] Fahad Taha AL-Dhief, Naseer Sabri, NM Abdul Latiff, NNNA Malik, Musatafa Abbas, Abbood Albader, Mazin Abed Mohammed, Rami Noori AL-Haddad, Yasir Dawood Salman, Mohd Khanapi, et al. "Performance comparison between TCP and UDP protocols in different simulation scenarios". In: *International Journal of Engineering & Technology* 7.4.36 (2018), pp. 172–176.

[37] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. "A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration". In: *ACM Computing Survey* 51.6 (Jan. 2019), pp. 1–29. ISSN: 0360-0300. DOI: 10.1145/3292674. URL: https://doi.org/10.1145/3292674.

[38] Fanyi Duanmu, Yixiang Mao, Shuai Liu, Sumanth Srinivasan, and Yao Wang. "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers". In: *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2018, pp. 1–6.

[39] Andrew T Duchowski, Margaux M Price, Miriah Meyer, and Pilar Orero. "Aggregate gaze visualization with real-time heatmaps". In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM. 2012, pp. 13–20.

[40] Farjana Z Eishita, Kevin G Stanley, and Alain Esquivel. "Quantifying the differential impact of sensor noise in augmented reality gaming input". In: *2015 IEEE Games Entertainment Media Conference (GEM)*. IEEE. 2015, pp. 1–9.

[41] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. "The many faces of publish/subscribe". In: *ACM computing surveys (CSUR)* 35.2 (2003), pp. 114–131.

[42] Phil Factor. *Statistics in SQL: Simple Linear Regressions*. https://www.red-gate.com/simple-talk/blogs/statistics-sql-simple-linear-regressions/. Online; accessed 23-April-2019. 2017.

[43] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. "Fixation prediction for 360 video streaming in head-mounted virtual reality". In: *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM. 2017, pp. 67–72.

[44] Stefano Fibbi, Lucio Davide Spano, Fabio Sorrentino, and Riccardo Scateni. "Wobo: Multisensorial Travels Through Oculus Rift". In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM. 2015, pp. 299–302.

[45] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. *Hypertext transfer protocol–HTTP/1.1*. Tech. rep. 1999.

[46] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*. Vol. 7. Doctoral Dissertation. University of California, Irvine Irvine, USA, 2000.

[47] Mike Freedman. *Time-series data: Why (and how) to use a relational database instead of NoSQL*. https://blog.timescale.com/time-series-data-why-and-how-to-use-a-relational-database-instead-of-nosql-d0cd6975e87c/. Online; accessed 23-April-2019. 2017.

[48] Paul Fremantle. "A reference architecture for the internet of things". In: *WSO2 White paper* (2014).

[49] Stephan Fremerey, Ashutosh Singla, Kay Meseberg, and Alexander Raake. "AVtrack360: An open dataset and software recording people's head rotations watching 360-degree videos on an HMD". In: *Proceedings of the 9th ACM Multimedia Systems Conference*. 2018, pp. 403–408.

[50] Zoltán Gál, Gergely Kocsis, Tibor Tajti, and Robert Tornai. "Performance evaluation of massively parallel and high speed connectionless vs. connection oriented communication sessions". In: *Advances in Engineering Software* 157 (2021), pp. 1–20. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2021.103010.

[51] Jonathan Gandrud and Victoria Interrante. "Predicting destination using head orientation and gaze direction during locomotion in vr". In: *Proceedings of the ACM Symposium on Applied Perception*. ACM. 2016, pp. 31–38.

[52] Ramin Ghaznavi Youvalari. *Encoding and Streaming Solutions for Immersive Virtual Reality Video*. Doctoral Dissertation. Tampere University, 2021.

[53] Joseph H Goldberg and Anna M Wichansky. *Eye tracking in usability evaluation: A practitioner's guide*. Elsevier, 2003, pp. 493–516. ISBN: 9780444510204. DOI: 10.1016/B978-044451020-4/50027-X.

[54] Jim Gray, David T Liu, Maria Nieto-Santisteban, Alex Szalay, David J DeWitt, and Gerd Heber. "Scientific data management in the coming decade". In: *Acm Sigmod Record* 34.4 (2005), pp. 34–41.

[55] Eduard Groen and Matthias Koch. “How Requirements Engineering can benefit from crowds: Driving innovation with crowd-based techniques”. In: *Requirements Engineering Magazine* (June 2016), pp. 1–12.

[56] NR Haddaway, Paul Woodcock, Biljana Macura, and Alexadra Collins. “Making literature reviews more reliable through application of lessons from systematic reviews”. In: *Conservation Biology* 29.6 (2015), pp. 1596–1605.

[57] Yoshinobu Hagiwara. “Cloud based VR system with immersive interfaces to collect multimodal data in human-robot interaction”. In: *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE. 2015, pp. 256–259.

[58] Stavros Harizopoulos, Velen Liang, Daniel J Abadi, and Samuel Madden. “Performance tradeoffs in read-optimized databases”. In: *Proceedings of the 32nd international conference on Very large data bases*. 2006, pp. 487–498.

[59] Hamid M Hasan and SA Jawad. “IoT Protocols for Health Care Systems: A Comparative Study”. In: *International Journal of Computer Science and Mobile Computing* 7.11 (2018), pp. 38–45.

[60] Alan R Hevner. “A three cycle view of design science research”. In: *Scandinavian journal of information systems* 19.2 (2007), pp. 87–92.

[61] Eric Hodgson and Eric Bachmann. “Comparing four approaches to generalized redirected walking: Simulation and live user data”. In: *IEEE transactions on visualization and computer graphics* 19.4 (2013), pp. 634–643.

[62] Andreas Holzinger. “Usability engineering methods for software developers”. In: *Communications of the ACM* 48.1 (2005), pp. 71–74.

[63] Mohammad Hosseini and Viswanathan Swaminathan. “Adaptive 360 VR video streaming based on MPEG-DASH SRD”. In: *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2016, pp. 407–408.

[64] Mohammad Hosseini and Viswanathan Swaminathan. “Adaptive 360 VR video streaming: Divide and conquer”. In: *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2016, pp. 107–110.

[65] Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. “6-DOF VR videos with a single 360-camera”. In: *2017 IEEE Virtual Reality (VR)*. 2017, pp. 37–44. DOI: 10.1109/VR.2017.7892229.

[66] Yongqiang Huang and Hector Garcia-Molina. "Publish/subscribe in a mobile environment". In: *Wireless Networks* 10.6 (2004), pp. 643–652.

[67] Juhani Iivari. "A Paradigmatic Analysis of Information Systems As a Design Science". In: *Scandinavian Journal of Information Systems* 19.2 (2007), pp. 39–64.

[68] E Douglas Jensen. "Eliminating the hard/soft real-time dichotomy". In: *Embedded Systems Programming* 7.10 (1994), pp. 28–35.

[69] Einar Broch Johnsen and Olaf Owe. "An asynchronous communication model for distributed concurrent objects". In: *Software & Systems Modeling* 6.1 (2007), pp. 39–58.

[70] Jetendra Joshi, Vishal Rajapriya, SR Rahul, Pranith Kumar, Siddhanth Polepally, Rohit Samineni, and DG Kamal Tej. "Performance enhancement and IoT based monitoring for smart home". In: *2017 International Conference on Information Networking (ICOIN)*. IEEE. 2017, pp. 468–473.

[71] Syed Rameez Ullah Kakakhel, Tomi Westerlund, Masoud Daneshtalab, Zhuo Zou, Juha Plosila, and Hannu Tenhunen. "A qualitative comparison model for application layer IoT protocols". In: *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE. 2019, pp. 210–215.

[72] Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate. "A survey on application layer protocols for the internet of things". In: *Transaction on IoT and Cloud computing* 3.1 (2015), pp. 11–17.

[73] Barbara Kitchenham. "Procedures for performing systematic reviews". In: *Keele, UK, Keele University* 33.2004 (2004), pp. 1–26.

[74] Kazuhiko Kobayashi, Koichi Nishiwaki, Shinji Uchiyama, Hiroyuki Yamamoto, and Satoshi Kagami. "Viewing and reviewing how humanoids sensed, planned and behaved with mixed reality technology". In: *2007 7th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2007, pp. 130–135.

[75] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W Godfrey. "Mining modern repositories with elasticsearch". In: *Proceedings of the 11th working conference on mining software repositories*. 2014, pp. 328–331.

[76] George-Alex Koulieris, Bee Bui, Martin S. Banks, and George Drettakis. "Accommodation and Comfort in Head-Mounted Displays". In: *ACM Transactions on Graphics* 36.4 (July 2017), pp. 1–11. ISSN: 0730-0301. DOI: 10.1145/3072959.3073622. URL: https://doi.org/10.1145/3072959.3073622.

[77] Jay Kreps, Neha Narkhede, Jun Rao, et al. "Kafka: A distributed messaging system for log processing". In: *Proceedings of the NetDB*. Vol. 11. 2011, pp. 1–7.

[78] KRPano.com. *krpano XML Reference*. https://krpano.com/docu/xml/. Online; accessed 23-April-2019.

[79] Evgeny Kuzyakov and David Pio. *Next-generation video encoding techniques for 360 video and VR*. https://code.fb.com/virtual-reality/next-generation-video-encoding-techniques-for-360-video-and-vr/. Online; accessed 23-April-2019. 2016.

[80] Valerie Lampkin, Weng Tat Leong, Leonardo Olivera, Sweta Rawat, Nagesh Subrahmanyam, Rong Xiang, Gerald Kallas, Neeraj Krishna, Stefan Fassmann, Martin Keen, et al. *Building smarter planet solutions with mqtt and ibm websphere mq telemetry*. IBM Redbooks, 2012.

[81] Neal Leavitt. "Will NoSQL databases live up to their promise?" In: *Computer* 43.2 (2010), pp. 12–14.

[82] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, and Hongtaek Ju. "Correlation analysis of MQTT loss and delay according to QoS level". In: *The International Conference on Information Networking 2013 (ICOIN)*. IEEE. 2013, pp. 714–717.

[83] Xing Liu, Qingyang Xiao, Vijay Gopalakrishnan, Bo Han, Feng Qian, and Matteo Varvello. "360 innovations for panoramic video streaming". In: *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM. 2017, pp. 50–56.

[84] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. "360-degree Video Viewing Dataset in Head-Mounted Virtual Reality". In: *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM. 2017, pp. 211–216.

[85] Thomas Löwe, Michael Stengel, Emmy-Charlotte Förster, Steve Grogorick, and Marcus Magnor. “Visualization and analysis of head movement and gaze data for immersive video in head-mounted displays”. In: *Proceedings of the Workshop on Eye Tracking and Visualization (ETVIS)*. Vol. 1. 2015.

[86] Hao Luo, Tien-Szu Pan, Jeng-Shyang Pan, Shu-Chuan Chu, and Bian Yang. “Development of a three-dimensional multimode visual immersive system with applications in telepresence”. In: *IEEE Systems Journal* 11.4 (2015), pp. 2818–2828.

[87] Mohamed Anis Mastouri and Salem Hasnaoui. “Performance of a publish/-subscribe middleware for the real-time distributed control systems”. In: *Computer Science and Network Security* 7.1 (2007), pp. 313–319.

[88] Ilias Mavridis and Helen Karatza. “Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark”. In: *Journal of Systems and Software* 125 (2017), pp. 133–151.

[89] Nenad Medvidovic and Richard N Taylor. “Software architecture: foundations, theory, and practice”. In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. Vol. 2. IEEE. 2010, pp. 471–472.

[90] Bernd Werner Meister, Philippe A Janson, and Liba Svobodova. “Connection-oriented versus connectionless protocols: A performance study”. In: *IEEE transactions on computers* 100.12 (1985), pp. 1164–1173.

[91] Paul Milgram and Fumio Kishino. “A taxonomy of mixed reality visual displays”. In: *IEICE TRANSACTIONS on Information and Systems* 77.12 (1994), pp. 1321–1329.

[92] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. “Automatic personalization based on web usage mining”. In: *Communications of the ACM* 43.8 (2000), pp. 142–151.

[93] Mohamed A Mohamed, Obay G Altrafi, and Mohammed O Ismail. “Relational vs. nosql databases: A survey”. In: *International Journal of Computer and Information Technology* 3.03 (2014), pp. 598–601.

[94] C Mohan. “History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla”. In: *Proceedings of the 16th International Conference on Extending Database Technology*. 2013, pp. 11–16.

[95] Javier Molino, Antoni Bibiloni, and Antoni Oliver. “Keys for successful 360-degree hypervideo design: A user study based on an xAPI analytics dashboard”. In: *Multimedia Tools and Applications* 79 (Apr. 2020), pp. 22771–22796. DOI: 10.1007/s11042-020-09059-2.

[96] *Multi-dimensional Clustering Using K-Means in Postgres SQL*. https://www.sisense.com/blog/multi-dimensional-clustering-using-k-means-postgres/. Online; accessed 09-June-2021. N.D.

[97] Nitin Naik. “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP”. In: *2017 IEEE international systems engineering symposium (ISSE)*. IEEE. 2017, pp. 1–7.

[98] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. “Adaptive 360-degree video streaming using scalable video coding”. In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 1689–1697.

[99] Afshin Taghavi Nasrabadi, Aliehsan Samiei, Anahita Mahzari, Ryan P McMahan, Ravi Prakash, Mylène CQ Farias, and Marcelo M Carvalho. “A taxonomy and dataset for 360-degree videos”. In: *Proceedings of the 10th ACM Multimedia Systems Conference*. 2019, pp. 273–278.

[100] Ameya Nayak, Anil Poriya, and Dikshay Poojary. “Type of NOSQL databases and its comparison with relational databases”. In: *International Journal of Applied Information Systems* 5.4 (2013), pp. 16–19.

[101] Cuong Nguyen, Stephen DiVerdi, Aaron Hertzmann, and Feng Liu. “CollaVR: Collaborative in-headset review for VR video”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 2017, pp. 267–277.

[102] Stephen Nicholas. *Power profiling: HTTPS long polling vs. MQTT with SSL, on android*. http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https. Online; accessed 23-April-2019. 2012.

[103] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[104] Sangyoon Oh, Jai-Hoon Kim, and Geoffrey Fox. “Real-time performance analysis for publish/subscribe systems”. In: *Future Generation Computer Systems* 26.3 (2010), pp. 318–323.

[105] Takashi Okuma, Masakatsu Kourogi, Nobuchika Sakata, and Takeshi Kurata. "Reliving museum visiting experiences on-and-off the spot". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2007, pp. 279–280.

[106] Adam Oliner, Archana Ganapathi, and Wei Xu. "Advances and Challenges in Log Analysis". In: *Commun. ACM* 55.2 (Feb. 2012), pp. 55–61. ISSN: 0001-0782. DOI: 10.1145/2076450.2076466. URL: https://doi.org/10.1145/2076450.2076466.

[107] Antoni Oliver, Javier del Molino, and Antoni Bibiloni. "Automatic View Tracking in 360-degree Multimedia Using xAPI". In: *Iberoamerican Conference on Applications and Usability of Interactive TV*. Springer. 2017, pp. 117–131.

[108] Antoni Oliver, Javier del Molino, and Antoni Bibiloni Coll. "Palma360: A 360-degree Web-based Multi-device Hypervideo". In: *ACM International Conference on Interactive Media Experiences*. 2020, pp. 165–169.

[109] Raymond Opdenakker. "Advantages and disadvantages of four interview techniques in qualitative research". In: *Forum qualitative sozialforschung/forum: Qualitative social research*. Vol. 7. 4. 2006.

[110] Carlos Ordonez. "Integrating K-means clustering with a relational DBMS using SQL". In: *IEEE transactions on Knowledge and Data engineering* 18.2 (2005), pp. 188–201.

[111] Carlos Ordonez. "Programming the K-means clustering algorithm in SQL". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 823–828.

[112] Cagri Ozcinar and Aljosa Smolic. "Visual attention in omnidirectional video for virtual reality applications". In: *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2018, pp. 1–6.

[113] Elia Palme, Chuan-Hoo Tan, Juliana Sutanto, and Chee Wei Phang. "Choosing the smart phone operating system for developing mobile applications". In: *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*. ACM. 2010, pp. 146–152.

[114] Zachary Parker, Scott Poe, and Susan V Vrbsky. "Comparing nosql mongodb to an sql db". In: *Proceedings of the 51st ACM Southeast Conference*. 2013, pp. 1–6.

[115] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. "A design science research methodology for information systems research". In: *Journal of management information systems* 24.3 (2007), pp. 45–77.

[116] Adam Philpot, Maxine Glancy, Peter J Passmore, Andrew Wood, and Bob Fields. "User experience of panoramic video in CAVE-like and head mounted display Viewing Conditions". In: *Proceedings of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video*. ACM. 2017, pp. 65–75.

[117] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. "Optimizing 360 video delivery over cellular networks". In: *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. ACM. 2016, pp. 1–6.

[118] Andrew B Raij and Benjamin C Lok. "Ipsviz: An after-action review tool for human-virtual human experiences". In: *2008 IEEE Virtual Reality Conference*. IEEE. 2008, pp. 91–98.

[119] Ch VN Sanyasi Rao and Tiruveedula Gopi Krishna. "SQL Performance Tuning". In: *International Journal of Science, Engineering and Computer Technology* 4.12 (2014), pp. 344–346.

[120] Sean Rhea, Eric Wang, Edmund Wong, Ethan Atkins, and Nat Storer. "Littletable: A time-series database and its uses". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. 2017, pp. 125–138.

[121] James M Ritchie, Raymond CW Sung, Heather Rea, Theodore Lim, Jonathan R Corney, and Iris Howley. "The use of non-intrusive user logging to capture engineering rationale, knowledge and intent during the product life cycle". In: *Management of Engineering & Technology, 2008. PICMET 2008. Portland International Conference on*. IEEE. 2008, pp. 981–989.

[122] Julian Rith, Philipp S Lehmayr, and Klaus Meyer-Wegener. "Speaking in tongues: SQL access to NoSQL systems". In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. 2014, pp. 855–857.

[123] Guoping Rong, Qiuping Zhang, Xinbei Liu, and Shenghiu Gu. "A Systematic Review of Logging Practice in Software Engineering". In: *2017 24th Asia-Pacific Software Engineering Conference (APSEC).* IEEE. 2017, pp. 534–539.

[124] Silvia Rossi, Francesca De Simone, Pascal Frossard, and Laura Toni. "Spherical clustering of users navigating 360 content". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2019, pp. 4020–4024.

[125] Vladimir Rubin, Irina Lomazova, and Wil MP van der Aalst. "Agile development with software process mining". In: *Proceedings of the 2014 international conference on software and system process.* Citeseer. 2014, pp. 70–74.

[126] Yuya Sasaki and Tetsuya Yokotani. "Performance evaluation of MQTT as a communication protocol for IoT and prototyping". In: *Advances in Technology Innovation* 4.1 (2019), pp. 21–29.

[127] Russell Sears, Catharine Ingen, and Jim Gray. "To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?" In: *Computing Research Repository - CORR* (Jan. 2007).

[128] Takashi Shibata. "Head mounted display". In: *Displays* 23.1 (2002), pp. 57–64. ISSN: 0141-9382. DOI: https://doi.org/10.1016/S0141-9382(02)00010-0.

[129] I Skerrett. "IoT Developer Survey 2016". In: *Eclipse IoT Working Group, IEEE IoT and Agile IoT* (2016), pp. 1–39.

[130] Dejan Skvorc, Matija Horvat, and Sinisa Srbljic. "Performance evaluation of Websocket protocol for implementation of full-duplex web streams". In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* IEEE. 2014, pp. 1003–1008.

[131] Aljoscha Smolic and Peter Kauff. "Interactive 3-D video representation and coding technologies". In: *Proceedings of the IEEE* 93.1 (2005), pp. 98–110.

[132] Benfano Soewito, Fergyanto E Gunawan, I Gede Putra Kusuma, et al. "Websocket to support real time smart home applications". In: *Procedia Computer Science* 157 (2019), pp. 560–566.

[133] Maarten van Steen and Andrew S Tanenbaum. "A brief introduction to distributed systems". In: *Computing* 98.10 (2016), pp. 967–1009.

[134] Jonathan Steuer. “Defining virtual reality: Dimensions determining telepresence”. In: *Journal of communication* 42.4 (1992), pp. 73–93.

[135] Michael Stonebraker. “SQL databases v. NoSQL databases”. In: *Communications of the ACM* 53.4 (2010), pp. 10–11.

[136] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. “Multi-path multi-tier 360-degree video streaming in 5G networks”. In: *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM. 2018, pp. 162–173.

[137] KR Suneetha and Raghuraman Krishnamoorthi. “Identifying user behavior by analyzing web server access log file”. In: *IJCSNS International Journal of Computer Science and Network Security* 9.4 (2009), pp. 327–332.

[138] Sampo Suonsyrjä, Kari Systä, Tommi Mikkonen, and Henri Terho. “Collecting Usage Data for Software Development: Selection Framework for Technological Approaches.” In: *Proceedings - SEKE 2016: 28th International Conference on Software Engineering and Knowledge Engineering*. 2016, pp. 114–119.

[139] Chengzhou Tang, Oliver Wang, Feng Liu, and Ping Tan. “Joint Stabilization and Direction of 360-degree Videos”. In: *ACM Transactions on Graphics (TOG)* 38.2 (2019), pp. 1–13.

[140] Yisheng Tang, Fengjie Wu, Zuolian Liu, and Weijian Mai. “Research on NAT Traversal Communication based on MQTT”. In: *2021 9th International Conference on Communications and Broadband Networking*. 2021, pp. 186–191.

[141] Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. “Performance evaluation of MQTT and CoAP via a common middleware”. In: *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*. IEEE. 2014, pp. 1–6.

[142] Minoru Uehara. “A case study on developing cloud of things devices”. In: *Complex, Intelligent, and Software Intensive Systems (CISIS), 2015 Ninth International Conference on*. IEEE. 2015, pp. 44–49.

[143] Mehar Ullah, Syed Rameez Ullah Kakakhel, Tomi Westerlund, Annika Wolff, Dick Carrillo, Juha Plosila, and Pedro HJ Nardelli. "IoT protocol selection for smart grid applications: Merging qualitative and quantitative metrics". In: *43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE. 2020, pp. 993–998.

[144] Vijay K. Vaishnavi and William L. Kuechler. "Design Science Research in Information Systems". In: *Association for Information Systems* (Jan. 2004), pp. 1–45. ISSN: 02767783. DOI: 10.1007/978-1-4419-5653-8.

[145] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. "A Dataset for Exploring User Behaviors in VR Spherical Video Streaming". In: *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM. 2017, pp. 193–198.

[146] Tetsuya Yokotani and Yuya Sasaki. "Comparison with HTTP and MQTT on required network resources for IoT". In: *Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016 International Conference on*. IEEE. 2016, pp. 1–6.

[147] Alireza Zare, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. "HEVC-compliant tile-based streaming of panoramic video for virtual reality applications". In: *Proceedings of the 24th ACM international conference on Multimedia*. ACM. 2016, pp. 601–605.

[148] S Zepeda and A Nuñez. "Asynchronous Communication for Improved Data Transport in The Network". In: *Journal of Physics: Conference Series*. Vol. 1828. 1. IOP Publishing. 2021, pp. 1–7. DOI: 10.1088/1742-6596/1828/1/012078.

PUBLICATIONS

PUBLICATION
I

Towards Framework for Choosing 360-degree Video SDK

A. Luoto

DOI: 10.5220/0006470600810086

Towards Framework for Choosing 360-degree Video SDK

Antti Luoto
Pervasive Computing, Tampere University of Technology, Korkeakoulunkatu 10, FI-33101 Tampere, Finland
antti.l.luoto@tut.fi

Keywords: 360-degree video, SDK, Software Development Kit

Abstract: 360-degree videos are gaining popularity among consumers. Still, software developers are early adopters of technology so it is important to map their needs for 360-degree video development. They use software development kits that help creating software on the 360-degree video software domain. We want to find out which factors developers need to take into account when choosing these software development kits. In this position paper we describe a preliminary 360-degree video SDK choosing criteria, based on literature and our own experiences, which we plan to evaluate with a survey.

1 INTRODUCTION

360-degree videos, also known as spherical videos, are getting more popular (Alface et al., 2011). Applications for 360-degree videos can be found for example from entertainment, industry, surveillance, and robotics. One of the reasons for popularity is that head mounted displays (HMD) supporting 360-degree video and virtual reality (VR) have become easily available for consumers. HMD is a wearable display device which consists of an optical system in a helmet with displays located in front of user's eyes creating an illusion of depth (Shibata, 2002). They are applicable for presenting interactive spatial information such as 360-degree videos or VR worlds. HMDs include dedicated display devices like Oculus Rift and HTC Vive in addition to mobile phone devices attached to headsets like Google Cardboard/Daydream and Gear VR. 360-degree videos are also getting more popular in web applications. For example both Youtube and Facebook support 360-degree videos.

360-degree video domain often requires reading the sensors of HMDs and calculating sphere mathematics. Luckily, there are Software Development Kits (SDK) that help developers with such tasks. We define SDK by using the definition by (Palme et al., 2010): "The SDK is as a set of development tools that allows a developer to create applications for a certain software package and hardware platform. SDKs differ in terms of their programming code languages, their libraries and API support." Thus, 360-degree video SDK is an SDK that allows developing 360-degree video applications. A 360-degree video SDK can, for example, offer an API that helps creating a video player which can detect the head movements in spherical videos both in monoscopic or stereoscopic mode. Examples of 360-degree video SDKs include Google VR SDK, OZO player SDK and KRPano. Comparing those SDKs and finding the best one for development work can require a considerable amount of time.

360-degree video SDKs offer many useful features but naturally they have different specifications and features, and they cannot offer everything for everybody. For example, our experience is that some 360-degree video SDKs for mobile devices do not allow easy development of user interface (UI) elements such as user interaction points. However, some of those 360-degree video SDKs can be integrated with a popular game engine called Unity 3D. With the help of Unity 3D or similar game engines, UI elements can be added more easily on top of 360-degree videos. Still, using complex tools such as Unity 3D can require more resources. For example, (Linowes and Schoen, 2016) state that an empty Unity 3D scene for Android requires a much bigger application package than a simple native code application, which has an increasing effect on memory and battery consumption.

The motivation for the work is that we have use cases for 360-degree videos so we need to choose a convenient SDK to work with. Further, we noticed that there seems to be relatively few scientific publications about choosing 360-degree video SDKs. The study aims to gain knowledge about the growing field, and the topic is significant for developers who need a scientifically structured criteria for choosing 360-degree video SDKs, for example for inspecting development tools.

Our research questions are:

- Which criteria is important when software developers choose SDKs for 360-degree video application development?
- What features software developers hope for 360-degree video SDKs to have?

To answer these questions we present a preliminary criteria for choosing an SDK from the developer's viewpoint. Our study can be seen as a contribution to developer experience studies.

The structure of the paper is as follows. Section 2 presents the scientific background for our work. Section 3 describes the identified criteria for choosing a 360-degree video SDK. Section 4 analyses the criteria and planned survey from a critical point of view. Section 5 concludes the paper and describes our on-going and planned research.

2 BACKGROUND

Our work can be seen as a part of developer experience studies. According to (Fagerholm and Münch, 2012) developer experience "consists of experiences relating to all kinds of artifacts and activities that a developer may encounter as part of their involvement in software development." They define three developer experience categories: development infrastructure, feelings about work and the value of one's own contribution. Our study relates mostly to the first one since SDKs are a part of development infrastructure.

Since 360-degree video software development has similarities to VR software development, (Bierbaum and Just, 1998) offers the most related background for our work. According to them, the primary requirements for VR development environment include performance, flexibility, and ease of use. On a more detailed level they list required capabilities and factors such as cross-platform development, support for VR hardware, high-level and low-level interfaces, programming languages, user interaction, minimal limitations, and choosing between commercial and open solutions, all of which we also included in our criteria. They discuss about the whole development environment where as we concentrate more on choosing only the SDK. The following background describes research about how developers choose their SDKs on other domains (than 360-degree video domain) and what they need to consider when doing that.

(Palme et al., 2010) propose a six-dimension benchmark (security, individual and organization buyer choice, market growth, ease of implementation and net revenue) for choosing a smart phone operating system (OS) for mobile application development. The dimensions are based on their own opinion about the most critical ones. They take SDK related viewpoints into account for example by saying that the OS SDK for Android is related to ease of development and market availability. They also note that the license of an SDK can have an effect on the usage decision. We did not include security in our criteria because we think that security is not among of the most essential characteristics for a 360-degree video SDK. If there are security related requirements, the other software components should offer the solution for most of them.

(Nykaza et al., 2002) interviewed developers about their needs for SDK documentation. They studied one particular documentation and give a detailed analysis of useful documentation features such as necessary content, taking into account the target audience and prerequisite knowledge, etc. Though documentation is an important part of using an SDK, we are not interested only about documentations. Documentation is however related to "ease of implementation" found in our criteria.

(Dalmasso et al., 2013) acknowledge the problem of the variety of different platforms and SDKs. They formed a classification and a comparison for different cross platform development tools for mobile platforms. The classification is based on general desirable requirements identified by themselves. The criteria they use has some similar aspects to our work. For example, they included the SDK's ability to be used for multiple platform development (also mentioned in our interview) as a part of their criteria. Additionally, they discuss performance in terms of CPU, memory and power usage, which we approach from the low resource consumption point of view.

(Argyriou et al., 2016) discuss about the challenges of designing UI in mixed 360-video and game environment using Google VR SDK and Unity 3D. Though they do not concentrate on creating choosing criteria of different SDKs, they are interested about user interaction which is also taken into account in our criteria.

3 CRITERIA FOR CHOOSING 360-DEGREE VIDEO SDK

We present a criteria about the factors software developers need to concider when choosing a 360-degree video SDK. The criteria is based on scientific literature, an interview with a 360-degree video application developer from industry (with seven years of 360-degree video application development experience), our own experiences (six months with Google VR SDK for Android, Nokia OZO SDK and Unity 3D). The interview with an expert was helpful especially from the web development perspective. Additional background for the criteria comes from our discussions with industry partners that have made us interested in some particular use cases such as user logging and user interaction.

3.1 Platforms, Domains and SDKs

As Table 1 presents, there are multiple platform alternatives for 360-degree video applications. Every SDK can not support every platform but for example web browser applications can be run in different kind of environments and Cardboard/Daydream applications can be run on different mobile devices (Android and iOS). There are also many fields for 360-degree video applications such as education, entertainment, industry, and research, which can have an effect on the desired characteristics of an SDK. Further, Table 2 lists different 360-degree video SDKs. We might be missing some SDKs but the list works as an example of different alternatives and it reflects the difficulty of choosing the best one for development work from many options.

Table 1: Example platforms for 360-video applications

Platform
Web browser
Cardboard/Daydream
Windows
Linux
macOS
HTC Vive
Gear VR
Oculus Rift

3.2 Features and Characteristics of SDKs

The combination of different features and characteristics of SDK can be the most important reason for choosing one. Table 3 lists the features and characteristic that are based on our own development experience, knowledge gained from an interview with an expert from industry and aspects found from the research literature. There are naturally other features and characteristics as well but either we have experience or we have found scientific background for the chosen ones. The features and characteristics in the list are elaborated in the following paragraphs.

User interaction. For immersive experience, a convenient way to implement user interaction, for example by adding an embedded UI on top of 360-degree video, can be an important factor. Probably the development of UI requires some kind of graphical API. However, not every SDK offers a possibility to add user interaction points easily.

Minimal limitations. The usage of an SDK should not be restricted only to the ready-made features. A skillful programmer should be able to extend the functionality if needed.

Performance. While even modern mobile devices are powerful enough for showing 360-degree videos, the performance requirements can increase for high resolution videos with a high refresh rate including other computation. An SDK should provide sufficient performance for comfortable immersive experience. Some SDKs include performance monitors but it is probably not the most essential feature for a 360-degree video SDK if the development environment otherwise includes a performance monitor.

VR hardware support. HMDs can be integrated to different kinds of devices that help for example with navigation and user interaction. For example Google Daydream supports a decicated controller that can be used for pointing and clicking.

Low-level API. In addition to high-level interface, an SDK can offer a low-level interface. With low-level source code the developers can make applications that perform faster or use less resources. For example, Google VR SDK for Android comes with a native developer kit (NDK) that is less restricted than the Java SDK but requires knowledge about C and C++. On the other hand, development work's abstraction level can be even increased for example with Unity 3D integration.

Programming language. The programming language of an SDK can have an effect on the usage decision. Some programmers are more familliar with some languages or the platform can require a certain language. For example a high-level language can support cross-platform development better or Javascript can be needed for web development.

Multiple platforms. Often creating a application for a single platform is not enough but it the implementation is needed for other platforms as well. An example of multiple platform support is Google VR environment that is provided for Android and iOS in addition to integration with Unity 3D and Unreal game engines and support for web applications.

Table 2: Example 360-degree video SDKs

SDK	Developer	Platform
Google VR SDK	Google	Android, iOS, Unity 3D, Unreal, Web
OZO player SDK	Nokia	Android, iOS, Oculus Mobile, SteamVR
OpenVR SDK/SteamVR SDK	Valve	Multiple vendors
OSVR SDK	Open source	Open Source VR headset
VR One SDK	Zeiss	VR One
krpano	krpano	Web
Pano2VR	Garden Gnome Software	Web, Cardboard
Marzipano	Open source	Web
Oculus SDKs	Oculus VR	Rift, Gear VR, Unity 3D, Unreal, Web, PC

Table 3: Different characteristics or features of 360-degree video SDKs with background

Feature or characteristic
User interaction (Argyriou et al., 2016)
Minimal limitations (Bierbaum and Just, 1998)
Performance (Bierbaum and Just, 1998)
VR hardware support(Bierbaum and Just, 1998)
Low-level API (Bierbaum and Just, 1998)
Programming language (Bierbaum and Just, 1998)
Multiple platforms (Dalmasso et al., 2013)
Low resource usage (Dalmasso et al., 2013)
Content management (Interview)
Web support (Interview)
Access to sensor data (LaValle et al., 2014)
Viewport tracking (LaValle et al., 2014)
Multiple 360 video formats (Own experience)
360 video format detection (Own experience)
DRM protection support (Own experience)
Free / Open source license (Palme et al., 2010)
Ease of implementation (Palme et al., 2010)
Market situation (Palme et al., 2010)

Low resource usage. Playing 360-degree videos can require relatively much computing power for mobile devices. With low resource usage we mean the SDK's ability to keep CPU, memory and power usage on minimal level for example for saving battery resources.

Content management. Some 360-degree video use cases are related to content management. For example a 360-degree video can be a part of an educational web page managed with a content management system. Thus, SDKs could support embedding 360-degree videos in varying content environments. On the other hand, content management inside 360-degree videos can be important as well. For example, sometimes it would be useful to add text on top a video or highlight a part of the it according to associated metadata.

Web support. While many 360-degree video applications are made for mobile devices, a support for web applications can be more important in the future when 360-degree videos become more popular in web.

Access to sensor data. An SDK can support different ways to access sensor data for head tracking. For example, the the head orientation can be retrieved in many formats such as euler angles, quaternions or matrix data. In addition, accessing accelerometer can be needed.

Viewport tracking. With viewport tracking we mean the video player's ability to automatically adapt to user's head orientation. For HMD usage this is basically a required ability but for web applications it can be preferable to navigate with mouse dragging.

Multiple 360 video formats. 360-degree videos come in multiple formats so an SDK should support as many as possible. They can be monoscopic (each frame is monocular equirectangular panorama) or stereoscopic (two vertically-stacked equirectangular panoramas) or the video be can stream, MPEG-4, webm etc.

360 video format detection. In addition to be able to play different 360-degree videos, it would be helpful for an SDK to detect the video format. For example, when using the class VRVideoView of Google VR SDK for Android, it is required to set the video format (monoscopic or steroscopic) in the program code because the player can not detect the video format by itself.

DRM protection support. As traditional videos, 360-degree videos can be protected with digital rights management (DRM) techniques. Not all SDKs offer playback for DRM protected videos.

Free / Open source license. The license of an SDK can affect the usage decision. For example, an individual developer getting familiar with field might want to start with a completely free SDK while a company might want to pay for non-restricted usage.

Ease of implementation. When choosing tools for software development work, the ease of implementation can be an important aspect. Ease of implementation includes things such as good documentation, familliar technologies, the quality of an API, etc. We include the easiness of integration with other components such as OS's and game engines under this category.

Market situation. Market situation can have an effect on the SDK usage decision. For example new devices can have new features that have the charm of novelty. In addition, the organization's strategy can determine the used platform.

4 DISCUSSION

When starting to work with 360-degree video development, we realized that choosing an SDK for 360-degree video application development is not an easy task. Additionally, we could not find a proper scientific criteria for choosing an SDK. Since 360-degree videos are a growing phenomena, a criteria for choosing the tools for the development work is beneficial for multiple parties.

To evaluate our criteria, we plan to conduct a survey. We chose the method because it is inexpensive, we hope to reach a large respondent group and the responses are anonymous. Disadvantages in using surveys include the inflexibility of the survey form and the lack of human interaction. Piloting the survey beforehand is important.

We plan to make an online survey with Google Forms. Primarily, we will call for participants from an association called Virtual Reality Finland that supports the development of the VR and AR ecosystem in Finland. The survey is planned to be lightweighted and not requiring much time to answer (10-15 minutes). According to plan, the survey has 10 questions divided to four sections: five multiple choise questions, four open field questions and one grid question with a 5-step answer scale. We aim to keep the questions short and simple.

A good survey should be clear, easy to follow, and provide enough information for respondents. There is a danger that some respondents do not understand what we mean with the questions. Answering to the open text questions can be more difficult for some respondents, so is it possible that they will leave the open field empty, even though open text answers could give the most interesting insight for us.

Our own development experience was limited to working with Google VR SDK for Android, Nokia OZO SDK and Unity 3D with simple applications. For that reason, we wanted to interview experts from the industry. However, we only managed to conduct one interview because it turned out to be difficult to get interviews from industry experts. That is one of the reasons we try to reach for a larger respondent group with an online survey.

The group of survey respondents can be expected to be quite exclusive since only developers with experience about 360-degree video SDKs are able to answer. Not any software developer can give proper insight on the topic. Therefore, getting a large enough response set for making meaningful research is not an easy task.

The answers will be analyzed statistically. Open answers naturally require more preparation for analysis but at first we intend to categorize them for further quantitative analysis. We also hope to get enough material for qualitative analysis.

Our criteria reflects our own interests to some extent. We are most interested in some particular use cases (like user logging and user interaction). However, we did not want to restrict the criteria only to those topics but we wanted to gain more wide view on the field, and we found support from the literature for many aspects. On the other hand, we assume that some interesting and important aspects were not included. Therefore, we hope that the planned open questions in the survey will give insight on those factors.

While the criteria and the survey is not the main goal of our research project, it is an important first step to gain knowledge about the field. We realize that there is a gap in current research not providing enough knowledge about developer experience in 360-degree video development. Studying software developers is important because, for example, (Yucel and Edgell, 2015) state that software developers invent uses for devices popular in future and they act as early adopters of technology, so their preferences can have effects on early market advantages.

5 CONCLUSIONS

In this position paper we presented a preliminary criteria for choosing a 360-degree video SDK. The criteria is based on research literature, our own experiences and an interview with an expert on the domain. To evaluate our criteria, we plan to conduct an online survey for software developers working in the field of 360-degree videos. Our eventual goal is to find out on which criteria software developers choose 360-degree video SDKs and what features are expected from them.

The motivation for the work comes from the need to sort out the field for further development of 360-

degree video applications. We need to choose a proper SDK for our use cases which include user interaction and user logging. In addition we hope that we can identify functionality gaps in the current SDKs.

The upcoming survey will be significant because 360-degree videos are gaining popularity among consumers, the developers are early adopters of technology and there are relatively few scientific publications about choosing 360-degree video SDKs.

This work is an initial study for a research project called 360 Video Intelligence. The purpose of the project is to create a 360-degree video platform which provides an easy way to run different kinds of analysis, for example object detection algorithms, on 360-degree videos. The videos with added metadata will be then played on 360-degree video player application. However, the player will not only play the video with visualized metadata but it will also gather user log for further analysis. Practical use cases for user logging include view port prediction that can be used for example on providing better video resolution only to the field of view similarly to work presented in (Ochi et al., 2014). We will also need some kind of UI elements for visualizing the added metadata on 360-degree videos. With the knowledge gained from developing our criteria and the following survey, we can have a better understanding about developing such applications.

ACKNOWLEDGEMENTS

This study was made in a research project called 360 Video Intelligence. We would like to thank TEKES for funding the project.

REFERENCES

Alface, P. R., Macq, J.-F., and Verzijp, N. (2011). Evaluation of bandwidth performance for interactive spherical video. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE.

Argyriou, L., Economou, D., Bouki, V., and Doumanis, I. (2016). Engaging immersive video consumers: Challenges regarding 360-degree gamified video applications. In *Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), International Conference on*, pages 145–152. IEEE.

Bierbaum, A. and Just, C. (1998). Software tools for virtual reality application development. *Course Notes for SIGGRAPH*, 98.

Dalmasso, I., Datta, S. K., Bonnet, C., and Nikaein, N. (2013). Survey, comparison and evaluation of cross platform mobile application development tools. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 323–328. IEEE.

Fagerholm, F. and Münch, J. (2012). Developer experience: Concept and definition. In *Software and System Process (ICSSP), 2012 International Conference on*, pages 73–77. IEEE.

LaValle, S. M., Yershova, A., Katsev, M., and Antonov, M. (2014). Head tracking for the oculus rift. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 187–194. IEEE.

Linowes, J. and Schoen, M. (2016). *Cardboard VR Projects for Android*. Packt Publishing Ltd.

Nykaza, J., Messinger, R., Boehme, F., Norman, C. L., Mace, M., and Gordon, M. (2002). What programmers really want: results of a needs assessment for sdk documentation. In *Proceedings of the 20th annual international conference on Computer documentation*, pages 133–141. ACM.

Ochi, D., Kunita, Y., Fujii, K., Kojima, A., Iwaki, S., and Hirose, J. (2014). Hmd viewing spherical video streaming system. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 763–764. ACM.

Palme, E., Tan, C.-H., Sutanto, J., and Phang, C. W. (2010). Choosing the smart phone operating system for developing mobile applications. In *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, pages 146–152. ACM.

Shibata, T. (2002). Head mounted display. *Displays*, 23(1):57–64.

Yucel, I. H. and Edgell, R. A. (2015). Conceptualizing factors of adoption for head mounted displays: Toward an integrated multi-perspective framework. *Journal For Virtual Worlds Research*, 8(2).

PUBLICATION II

Systematic Literature Review on User Logging in Virtual Reality

A. Luoto

DOI: 10.1145/3275116.3275123

Systematic Literature Review on User Logging in Virtual Reality

Antti Luoto
Tampere University of Technology
Tampere, Finland
antti.l.luoto@tut.fi

ABSTRACT

In this systematic literature review, we study the role of user logging in virtual reality research. By categorizing literature according to data collection methods and identifying reasons for data collection, we aim to find out how popular user logging is in virtual reality research. In addition, we identify publications with detailed descriptions about logging solutions.

Our results suggest that virtual reality logging solutions are relatively seldom described in detail despite that many studies gather data by body tracking. Most of the papers gather data to witness something about a novel functionality or to compare different technologies without discussing logging details. The results can be used for scoping future virtual reality research.

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**; • **General and reference** → *General literature*;

KEYWORDS

Systematic Literature Review, User Logging

ACM Reference Format:
Antti Luoto. 2018. Systematic Literature Review on User Logging in Virtual Reality. In *Academic Mindtrek 2018 (Mindtrek 2018), October 10–11, 2018, Tampere, Finland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3275116.3275123

1 INTRODUCTION

Virtual reality (VR) has gained attention in the recent years because of the popularity of consumer affordable equipment. This has also increased interest in VR research and various kinds of user studies with data collection in VR has been conducted. The reasons for data collection varies from getting evidence about the functionality of a novel VR solution to analyzing users. User logging enables a way to do user analytics but it is difficult to say how popular it is in VR research. Related questions include what kind of systems are built for logging the data and how they are described in the literature.

In this systematic literature review, we study the role of user logging in VR research from software engineering perspective. We are especially interested in the following research questions:

- RQ1: How VR user logging appears in VR research?
- RQ2: How data collection has been made in VR research?
- RQ3: For which reasons the logging or data collection has been made in VR research?

First, we mapped the reviewed papers to the four VR data collection groups suggested by Steptoe et al. [79]: *body tracking* (kinesic behavior, verbal signals, proxemics), *questionnaires and interviews* (subjective sense and experience), *performance metrics* (action quantification) and *physiological response* (stimuli experienced in VR).

Then, we categorized the papers according to the reason of data collection. From the papers' content, we distilled the five following categories: "getting evidence about functionality", "how users act in application usage domain", "comparing VR-aided and non-aided", "comparison of different VR implementations" and "logging development".

After the categorization, we studied closely the papers that describe developing user logging systems in VR. It turned out that there are relatively few papers discussing the topic in detail even though we tried to include papers broadly rather than strictly so that we would not miss anything relevant. In addition to the detailed papers, there are a few papers briefly discussing topics such as logging features, log files, collaborative or distributed data collection, log analysis, and log visualization.

From the 638 papers found with the search terms, we included 78 papers for the final review. The outcome of this study can be used for getting insight on what VR research has been focused on, and for scoping future research. For example, our study suggests that body tracking is the most popular data collection method and there is a minority in studies collecting physiological data from VR users. In addition, the majority of the studies gather user data for getting functional evidence about a new VR solution. Still, our observations suggest that VR logging solutions are relatively seldom explicitly discussed even if some kind of (body) tracking is used in the study.

2 BACKGROUND

VR can be defined as "a real or simulated environment in which a perceiver experiences telepresence" [80]. VR often requires a use of special set of hardware, for example a head-mounted display (HMD). This special equipment in turn often includes a body tracking sensors and features so that VR environment can respond to user activities such as head movement. Via these sensors, it is possible to receive data about the user. Those data can be logged so that it is possible to do analysis based on the logs.

In general, software users have been tracked for various reasons. For example, in human-computer interaction work it can be used for getting statistics about the detailed use of the system, and it can be used after the release or during user testing [35]. Runtime traces have been analyzed for improving architecture, performance, design

Mindtrek 2018, October 10–11, 2018, Tampere, Finland
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6589-5/18/10…$15.00
https://doi.org/10.1145/3275116.3275123

and usability [74]. Multiple authors [8, 16, 62] discuss tracking and analyzing users in web.

VR developers have similar needs and reasons to log users. For example, Ritchie et al. [73] discuss the benefits of user logging within VR in the following way: it is an almost non-intrusive method of capturing a rich data source for analysis, it minimizes user interactions during the data capture, it has potential to reduce time overhead of the capturing process, and the captured data can also be reused. They also note that for achieving the full benefits of VR logging, capturing methods need to be researched. Our study contributes to identifying the capturing methods in need for more research.

Augmented reality (AR) and mixed reality (MR) are closely related concepts to VR and some logging techniques used for AR and MR can be applicable for VR as well. AR is something that combines real and virtual, is interactive in real time, and is registered in three dimensions [2]. MR is a broader term that combines AR and augmented virtuality [61]. While 360-degree videos are on the edge of being VR, we are also working with 360-degree videos and wanted to include papers from that domain. 360-degree videos are omnidirectional videos where user can control the direction of viewport in a spherical space and the controlling often happens with special equipment such as HMD. Similarities in logging 360-degree video users and VR users are related to logging information about head orientation and gaze direction. However, in 360-degree videos users seldom have a possibility to change location in the space, while there can be a user interface and user interactions similar to VR.

We have developed a user logging and visualization framework for 360-degree videos and we are interested in how our solution fits with the related research. On the other hand, we have a broader interest in what VR researchers need and use logging for, and we would like to see how popular is logging in relation to other data collection methods such as interviews. By conducting a systematic literature review, we aim to get insight on the role of logging in VR research.

We are not aware of other systematic reviews concentrating on user logging or data collection in VR systems. Still, VR related systematic reviews exist. For example, Santos et al. [19] identified usage of requirements engineering in development of VR systems which is related to our work in a sense that it has also a software engineering context. Berntsen et al. [5] present another systematic review where VR systems have been categorized to three categories according to their use: *health*, *exploration*, and *presentation and entertaiment.* For example, their *exploration* category overlaps partly with our *usage domain* category.

While it is not a systematic review, Zhao [93] surveyed VR domain and suggests different classifications for VR systems. For example, classification by system functions divides VR systems in three categories: *training & drill*, *planning & design*, and *presentation & entertainment* whereas classification by data flow makes four categories: *platform data* (metadata produced by computer system, network, etc.), *model data* (world, 3D scenes, etc.), *sense data* (output offered to user), and *control data* (input by users). We did not think that those classifications are useful to us as such, but we are mainly interested in the last aforementioned *control data.*

Furthermore, user logging appears in systematic reviews outside VR domain. For example, a systematic literature review by Velsen et al. [82] found that questionnaires seem to be the most popular method, followed by interviews and data log analysis in user-centered evaluation studies.

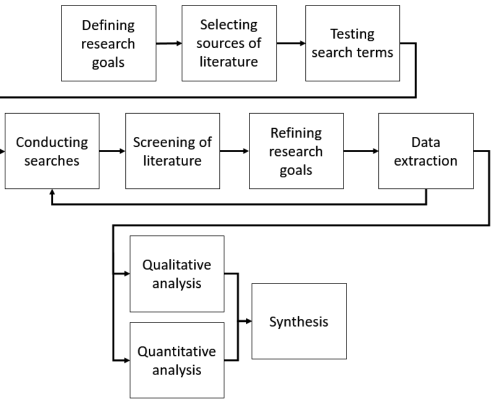

Figure 1: Flow of the review process.

3 REVIEW PROCESS

3.1 Planning

We created the search terms based on our RQs. VR is a bit difficult search term because VR can relate to immersive or non-immersive VR where immersion refers to illusion of being physically present in a simulated world [26]. We were mainly interested about immersive VR but included also papers about non-immersive VR. AR and MR are closely related concepts so we included them in our search terms since a logging system for them can be relevant to VR research as well.

- VR logging
- AR logging
- virtual reality logging
- augmented reality logging
- HMD logging
- head mounted display logging
- helmet mounted display logging
- mixed reality logging
- head orientation logging
- head orientation tracking

Figure 1 visualizes the flow used in the study. First, we defined preliminary research goals that were just to get an idea of popularity and role of logging in VR. Then we chose our electronic databases IEEE Xplore and ACM Digital Library because those are often used in the field of information technology. Testing of the search terms could have been a more accurate process, but we tested a few terms to see that we receive at least some relevant results. Testing the search terms was a bit problematic because we refined the research goals later. Before conducting the final searches, we created inclusion and exclusion criteria. The research was conducted during late 2017 and early 2018.

3.2 Inclusion and Exclusion Criteria

We aimed to get a large number of studies to get a fuller representation of the available research rather than a focused small number of studies that, on the other hand, could improve the quality of the papers. We tried to find HMD related logging but if it was not explicitly stated, other VR related data collection such as mobile or wearable devices was also allowed. It was not always easy to see if the paper used logging, so we tried to accept papers openly. We accepted papers starting from year 2000. The reason was that we think HMD based VR has progressed after 1990s and we thought that VR logging analytics does not have a long history before 2000. The search results did not include many papers done before 2000 so only few papers were not included because of the publishing year.

We excluded the paper if VR, AR, MR or 360-degree videos was not mentioned. In addition, some kind of data collection was required so, for example, the paper should not only describe a new technology without data collection. As an exception, a paper explicitly discussing logging development was not required to include data collection.

3.3 Identification

We executed the searches by starring with IEEE Xplore and then continued with ACM Digital Library. We used the default search field of both the database websites.

Table 1. Selection process of the literature.

Source	*All*	*Abstract*	*Year*	*Content*	*Included*
IEEE	328	74	72	37	37
ACM	310	94	90	41	41
Total	638	170	164	78	78

Table 1 summarizes the effect of inclusion and exclusion criteria on the amount of papers on different phases. Column *All* shows the full amount of found papers. The column is problematic in a sense that we only checked the 50 first results from the queries that produced hundreds of results. We did this because we discovered that the relevance decreases often fast after the first ten results. Column *Abstract* shows the amount of promising papers accepted after reading title and abstract. It can be seen that most of the papers were excluded at this point. Column *Year* shows the papers accepted by year.

We made data extraction after almost every single search and not only after all of the searches had been conducted. This is presented in Figure 1 with an arrow from *Data extraction* to *Conducting searches*. This gave us a possibility to make small adjustments to the research goals, data extraction form, and inclusion criteria during the searches.

3.4 Data Analysis

Since we had papers that include similar and comparable content, we had material for a quantitative analysis. On the other hand, since we were interested about the reasons for data collection, we had to identify those reasons as well. Thus, we distilled five categories according to the reason of data collection by reading the papers and categorized the papers to those groups. The five groups were:

(1) Papers that collected data to prove something about a novel technique or a product they have studied or developed.
(2) Papers that studied how users act in a certain domain, for example, in a museum.
(3) Papers that compared users' activities with and without VR gear.
(4) Papers that compared different technologies.
(5) Papers that explicitly discussed or studied VR logging.

Qualitative aspect of the analysis comes from the effort of identifying those categories and finding the role of logging. The papers discussing about logging solutions from technical point of view were not always comparable to those concentrating on user studies.

4 RESULTS

Table 2. Amount of publications per year.

Year	*ACM*	*IEEE*	*Total*
2000	0	1	1
2001	0	2	2
2002	0	0	0
2003	0	1	1
2004	1	0	1
2005	0	1	1
2006	2	1	3
2007	0	3	3
2008	0	2	2
2009	2	0	2
2010	1	0	1
2011	1	0	1
2012	2	4	6
2013	2	3	6
2014	5	1	6
2015	5	5	10
2016	10	7	18
2017	10	6	17

The amount of included publications per year can be seen in Table 2. It can be seen that the amount of related research has increased during the recent years. Figures 2 and 3 visualize our analysis results. One paper could fit to multiple categories which explains why the sum of papers in Figure 2 is bigger than *Total* in Table 1.

In Figure 2, the data collection categorization (body tracking, performance metrics, physiological response, and questionnaires and interviews) is based on suggestion by Steptoe and Steed [79]. It can be seen that body tracking was the most popular method. We included everything that collected data by tracking kinesics, verbal signals, oculesics or proxemics into this category. For example, usage of HMD as a data collection tool was categorized in this group. Table 3 lists the publication references.

Body tracking was the most used data collection method (49%) and it was more used in papers found from IEEE Xplore when compared to ACM Digital Library. Performance metrics was the second in the amount of papers (42%), also a few more found from in IEEE Xplore. In this category, we included papers that measured

Table 3. Included papers categorized by data collection method and source.

Source	*Body tracking*	*Perf. metrics*	*Physiol. response*	*Quest. and interv.*
IEEE	[21–23, 25, 28, 32, 34, 39, 42, 46, 55–57, 60, 66, 69–71, 73, 76, 79, 81, 91, 92]	[4, 9, 10, 23, 28, 41, 42, 44, 45, 55, 56, 60, 65, 66, 69, 75, 85, 91]	[17, 79]	[3, 13, 21, 44, 50, 55, 60, 69, 75, 92]
ACM	[7, 11, 14, 18, 20, 27, 29, 52, 53, 63, 68, 78, 84, 87]	[1, 15, 29, 38, 40, 47, 49, 51, 53, 54, 72, 78, 86, 88, 89]	-	[1, 6, 12, 15, 18, 24, 30, 31, 33, 36, 37, 48, 53, 58, 59, 64, 67, 68, 72, 77, 78, 83]

Table 4. Included papers categorized by reason for data collection and source.

Source	*Func. evidence*	*Usage domain*	*VR vs. non-VR*	*Tech. comp.*	*Logging devel.*
IEEE	[9, 13, 22, 32, 39, 41, 42, 45, 46, 55, 57, 70, 71, 76, 81]	[56, 73]	[3, 10, 17, 28, 85, 91]	[21, 34, 44, 50, 60, 69, 75, 92]	[4, 25, 65, 73, 79]
ACM	[1, 11, 14, 15, 24, 29, 31, 36, 54, 58, 59, 63, 64, 67]	[37, 40, 51, 66]	[6, 12, 20, 38, 49, 72, 77]	[7, 18, 27, 30, 33, 47, 48, 51, 53, 54, 68, 78, 84, 87–89]	[52]

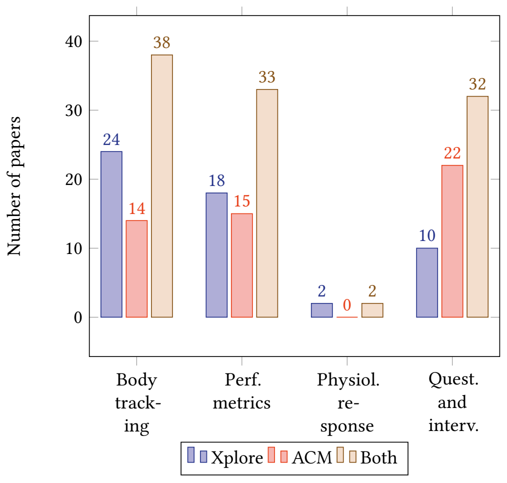

Figure 2. Categorization according to data collection method. The categories are body tracking, performance metrics, physiological response, and questionnaires and interviews.

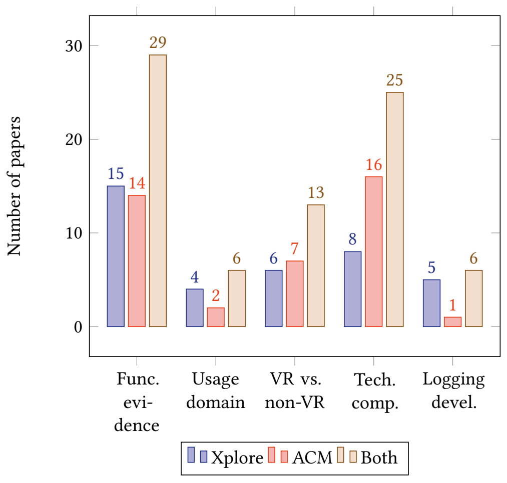

Figure 3. Categorization according to reason for data collection. The categories are getting functional evidence, collecting data in usage domain, comparing VR-aided and non-VR-aided, comparing technologies, and Logging development.

how users performed certain actions in the VR. For example, these papers measured how fast a test subject finished a certain operation in VR or how many times a test subject used certain interaction methods in VR.

The third biggest category (41%), questionnaires and interviews, was chosen if the paper mentioned using either of the methods. It was often relatively clear to see if the paper used this method. This category was more often found from ACM Digital Library and it was the only category where ACM Digital Library had a bigger set of papers when compared to IEEE Xplore.

Physiological response was chosen when the study measured phenomena like heart rate or blinking. Some papers studied physiological phenomena, for example cybersickness [83], using questionnaires but they were categorized to interviews and questionnaires group. We identified zero papers from ACM Digital Library and two papers (3%) from IEEE Xplore to this category [17, 79].

Figure 3 shows the classification to five categories according to the reason of data collection distilled from the papers' content. *Func. evidence* refers to getting evidence about the functionality of a VR solution, *Usage domain* refers to getting data about how users act in the application domain, *VR vs. non-VR* refers to comparisons between VR-aided and non-VR-aided usage, *Tech. comp.* refers to comparisons between different technology solutions, and *Logging devel.* refers to activities such as log analysis, log specification and log replay. One exception that did not fit to any of these categories was a paper about experiencing fun in VR despite cybersickness [83]. It could have been in a category called *Other* or *VR experience.* Table 4 lists the paper references according to reason for data collection.

Func. evidence is the most popular reason (37%) and it almost equally appeared in papers from both the sources. It was followed (32%) by *Tech. comp.* where ACM had more (21%) papers than IEEE (10%). The category *VR vs. non-VR* had almost equal amount of papers from both the sources (8% and 9%) and the total amount is clearly less when compared to the two biggest categories. *Logging devel.* shares the least amount of total papers (8%) with *Usage domain.*

Thus, the remaining six papers (8%) were categorized to *Logging devel.* group where we aimed to classify the papers focusing on logging. In those papers, the logged data was used for:

- Evaluating a logging format [73, 79]
- Analysing logs [25, 52, 73]
- Replaying VR events [4, 73]
- Log management [65]

To elaborate on the paper's content a bit more, Steptoe et al. [79] present a general multimodal data capture and analysis architecture that aims in log standardization. Ritchie et al. [73] demonstrate the potential of VR CAD tool logging by using an XML based log file and automated log analysis. Lo et al. [52] visually analyze head orientation of 360-degree video viewers and published an open head tracking dataset. Fitzgerald et al. [25] give a description of their body logging system with a motion storage and visual analysis. Belfore [4] describes a Java application using Virtual Reality Modeling Language which allows logging and restoration of VR sessions. Nakamura et al. [65] propose an AR behavior log management technique in which real world locations and objects are tagged with virtual cubes.

5 DISCUSSION

5.1 Role of Logging in VR Research

While we identified only six papers with logging as their primary topic, many other papers discussed logging aspects so that logging was not the main idea of the paper. We found the following topics to be discussed:

- Logging features or log files: [3, 9, 22, 34, 38, 39, 41, 48, 55, 60, 76, 81, 85, 87, 91].
- Collaborative or distributed data collection: [10, 28, 32, 46, 66, 68].
- Log analysis and visualization: [7, 20, 21, 23, 27, 45, 53, 56, 57, 70, 71, 88, 92].

Naturally, some kind of technical solution is needed for logging. Examples for such include: relational database [48], a multi-purpose logging module [9], Unity 3D tool [91], and logging server [46].

Orientation, position and time are common log entries in this domain. So, for example, Euler angles [39], rotational position and time [87], gyroscope, accelerometer, gaze [68], and magnetometer [55] have been logged. Less usual, and more domain specific, log data includes text entries [60], conversation [81], user's performance [25], trajectories [91], interactions [3], context [38], and activities [66].

Logging is a method for studying collaboration. It has been used, for example, for studying pair work in a combat simulator [10], interactive television event [28], humanoid robots' behavior [46], and general MR collaboration [68]. Despite that, the described logging architectures are more often local and focused on getting functional evidence about a single solution instead of being distributed platforms for long-term log analysis.

The eventual benefits of logging come from the log analysis. Examples of visual analysis include graphs, charts, plots [7, 20, 27, 45, 57], maps, trajectories and heat maps [23, 56, 70, 92], playback, and timelines [71]. An example of statistical analysis is game players' performance analysis [21].

When identifying the role of user logging in VR research, we came to conclusion that logging (only) head orientation and logging analysis are not very common (22%) [7, 20, 21, 27, 34, 44, 48, 53, 55, 60, 69–71, 76, 87, 88, 92]. Instead, many studies are interested in logging other wearable devices or sensors. Some of the logging is made without specific devices just by tracking the user's activities in virtual environment. The log can be also based on video motion capture.

Using body tracking equipment does not mean that the tracking information is explicitly logged or that the log is analyzed. Furthermore, logging details are not always outspoken even if rest of the VR system is explained in detail. However, if a study uses body tracking as a data collection method, explicit logging is more likely to be used.

While we accepted publications about 360-degree videos, only a few papers discussed them (for example [18, 24, 52, 55]). We could have found more by including them explicitly in the search terms.

5.2 Validity

Logging can be more popular than our results suggest – sometimes there is challenge to identify research that only tracks the user from research that also uses the tracking logs as research data by reading the paper.

One weakness in the study is that the review was completely done by one researcher. Guidelines for systematic reviews suggest that some decisions would be better to do with a support of another researcher to reduce the researcher bias [43].

The categorization by data collection method (Table 3) can be considered unbalanced and, as such, not well motivated. However, for other categories than physiological response it is relatively balanced, and it clearly shows that physiological response is not well represented in the past research. We expect this category to get more popular in the future since cybersickess has gained more attention recently.

Bookkeeping about duplicate papers found with the search terms was not thorough. For IEEE Xplore, this information was included more accurately but for ACM not every duplicate was marked especially in a situation when the query results only provided duplicates. However, the results do not include duplicate papers and we even excluded too similar papers (same authors, same topic, etc.) to reduce bias.

Snowballing is an alternative strategy to do a systematic review [90]. Snowballing to both directions together with the database searches could improve the validity and introduce more interesting papers.

Quality aspects of the included studies were not considered in depth. For the most cases, we required some kind of data collection, but for example, we could have assessed the study design to ensure a minimum level of quality [43]. On the other hand, then we could have had less material.

We used the default search field of the electronic libraries. The results could have been more accurate if we had used the advanced search options. For example, we could have used advanced search with Boolean expressions. Furthermore, we could have found more relevant papers with more and improved search terms. For example cave automatic virtual environment (CAVE) could have been included while our personal interests are on HMDs.

We chose year 2000 or later in our inclusion criteria. It is possible that there are relevant papers done before that. However, the VR technology evolves so that we think that studies that are more recent are more relevant. In addition, we think that there has been a relatively lot of VR research recently (after 2010) which can be also seen in our Table 2. In addition, there might be some studies published in 2017 that had not been yet added to the databases when we executed the queries.

One goal of a systematic review is to present a repeatable research process. Replication of this study is supported describing the process together with the inclusion and exclusion criteria. We assume that there are additional relevant publications available but our searches did not found them. However, we think that 78 papers is a representative amount of papers on which relatively valid conclusions can be made.

6 CONCLUSION

In this systematic literature review, we studied how user logging has been discussed in VR research. While identifying the role of logging, we categorized research papers according to data collection method and the reason for data collection.

Our observations suggest that publications about logging development in VR are relatively rare and logging details are not usually discussed even if logging has been made. The data collection categorization suggests that measuring physiological response is rare when compared to other data collection methods (body tracking, performance metrics and questionnaires and interviews). The categorization according to reason for data collection suggests that the most popular reasons for collecting data are to get functional evidence about a novel VR technology or comparing different VR technologies with each other.

According to this review, VR user logging development calls for more research, generalization and common practices. One way to improve the situation could be to encourage the VR researchers who log users to discuss the used logging procedure more in detail.

ACKNOWLEDGMENTS

The authors would like to thank Business Finland for funding the work.

REFERENCES

[1] Ryan Arisandi, Yusuke Takami, Mai Otsuki, Asako Kimura, Fumihisa Shibata, and Hideyuki Tamura. 2012. Enjoying virtual handcrafting with ToolDevice. In *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 17–18.

[2] Ronald T Azuma. 1997. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments* 6, 4 (1997), 355–385.

[3] Hani Bani-Salameh and Clinton Jeffery. 2015. Evaluating the Effect of 3D World Integration within a Social Software Environment. In *Information Technology-New Generations (ITNG), 2015 12th International Conference on*. IEEE, 255–260.

[4] Lee A Belfore and Suresh Chitithoti. 2000. An interactive land use VRML application (ILUVA) with servlet assist. In *Simulation Conference, 2000. Proceedings. Winter*, Vol. 2. IEEE, 1823–1830.

[5] Kristina Berntsen, Ricardo Colomo Palacios, and Eduardo Herranz. 2016. Virtual reality and its uses: a systematic literature review. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*. ACM, 435–439.

[6] Alberto Betella, Enrique Martínez Bueno, Wipawee Kongsantad, Riccardo Zucca, Xerxes D Arsiwalla, Pedro Omedas, and Paul FMJ Verschure. 2014. Understanding large network datasets through embodied interaction in virtual reality. In *Proceedings of the 2014 Virtual Reality International Conference*. ACM, 23.

[7] Kamran Binaee, Gabriel Diaz, Jeff Pelz, and Flip Phillips. 2016. Binocular eye tracking calibration during a virtual ball catching task using head mounted display. In *Proceedings of the ACM Symposium on Applied Perception*. ACM, 15–18.

[8] Jose Borges and Mark Levene. 2007. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Transactions on Knowledge and Data Engineering* 19, 4 (2007).

[9] Irina Branovic, Ranko Popovic, Nenad Jovanovic, Roberto Giorgi, Bosko Nikolic, and Miodrag Zivkovic. 2014. Integration of simulators in virtual 3D computer science classroom. In *Global Engineering Education Conference (EDUCON), 2014 IEEE*. IEEE, 1–4.

[10] Dennis G Brown, Joseph T Coyne, and Roy Stripling. 2006. Augmented reality for urban skills training. In *Virtual Reality Conference, 2006*. IEEE, 249–252.

[11] Xavier P Burgos-Artizzu, Julien Fleureau, Olivier Dumas, Thierry Tapie, François LeClerc, and Nicolas Mollet. 2015. Real-time expression-sensitive hmd face reconstruction. In *SIGGRAPH Asia 2015 Technical Briefs*. ACM, 9.

[12] Georgina Cárdenas-López, Perla Martinez, Giuseppe Riva, Ximena Duran-Baca, and Gonzalo Torres. 2015. Virtual reality environments as auxiliaries in the treatment of obesity. In *Proceedings of the 2015 Virtual Reality International Conference*. ACM, 1.

[13] Settachai Chaisanit, Napatwadee Sangboonnum Hongthong, Surachai Suksakulchai, and Chuchart Pinpat. 2012. Traditional musical Virtual Reality on M-learning. In *Internet Technology And Secured Transactions, 2012 International Conference for*. IEEE, 271–274.

[14] Jung-Woo Chang, Suk-Ju Kang, Min-Woo Seo, Song-Woo Choi, Sang-Lyn Lee, Ho-Chul Lee, Eui-Yeol Oh, and Jong-Sang Baek. 2017. Real-time temporal quality compensation technique for head mounted displays. In *SIGGRAPH Asia 2017 Posters*. ACM, 5.

[15] Zikun Chen, Wei Peng, Roshan Peiris, and Kouta Minamizawa. 2017. Thermo-Reality: thermally enriched head mounted displays for virtual reality. In *ACM SIGGRAPH 2017 Posters*. ACM, 32.

[16] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamas Sarlos. 2012. Are web users really markovian?. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 609–618.

[17] Tien-Yow Chuang, Chih-Hung Chen, Hwa-Ann Chang, Hui-Chen Lee, Cheng-Lian Chou, and Ji-Liang Doong. 2003. Virtual reality serves as a support technology in cardiopulmonary exercise testing. *PRESENCE: Teleoperators & Virtual Environments* 12, 3 (2003), 326–331.

[18] Tomás Dorta, Davide Pierini, and Sana Boudhraâ. 2016. Why 360 and VR headsets for movies?: exploratory study of social VR via hyve-3D. In *Actes de la 28ième conférence francophone sur l'Interaction Homme-Machine*. ACM, 211–220.

[19] Alinne C Correa dos Santos, Marcio Eduardo Delamaro, and Fatima LS Nunes. 2013. The relationship between requirements engineering and virtual reality systems: A systematic literature review. In *Virtual and Augmented Reality (SVR), 2013 XV Symposium on*. IEEE, 53–62.

[20] Andrew T Duchowski, Donald H House, Jordan Gestring, Robert Congdon, Lech Świrski, Neil A Dodgson, Krzysztof Krejtz, and Izabela Krejtz. 2014. Comparing estimated gaze depth in virtual and physical environments. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 103–110.

[21] Farjana Z Eishita, Kevin G Stanley, and Alain Esquivel. 2015. Quantifying the differential impact of sensor noise in augmented reality gaming input. In *Games Entertainment Media Conference (GEM), 2015 IEEE.* IEEE, 1–9.
[22] Marc Ericson, Takafumi Taketomi, Goshiro Yamamoto, Gudrun Klinker, C Santos, and Hirokazu Kato. 2015. [POSTER] Towards Estimating Usability Ratings of Handheld Augmented Reality Using Accelerometer Data. In *Mixed and Augmented Reality (ISMAR), 2015 IEEE International Symposium on.* IEEE, 196–197.
[23] Søren Eskildsen, Kasper Rodil, and Matthias Rehm. 2012. Visualizing learner activities with a virtual learning environment: Experiences from an in situ test with primary school children. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on.* IEEE, 660–661.
[24] Stefano Fibbi, Lucio Davide Spano, Fabio Sorrentino, and Riccardo Scateni. 2015. Wobo: Multisensorial travels through oculus rift. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems.* ACM, 299–302.
[25] Diarmaid Fitzgerald, John Foody, Dan Kelly, Tomas Ward, Charles Markham, John McDonald, and Brian Caulfield. 2007. Development of a wearable motion capture suit and virtual reality biofeedback system for the instruction and analysis of sports rehabilitation exercises. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE.* IEEE, 4870–4874.
[26] Laura Freina and Michela Ott. 2015. A literature review on immersive virtual reality in education: state of the art and perspectives. In *The International Scientific Conference eLearning and Software for Education*, Vol. 1. " Carol I" National Defence University, 133.
[27] Jonathan Gandrud and Victoria Interrante. 2016. Predicting destination using head orientation and gaze direction during locomotion in vr. In *Proceedings of the ACM Symposium on Applied Perception.* ACM, 31–38.
[28] Chris Greenhalgh, Steve Benford, and Mike Craven. 2001. Patterns of network and user activity in an inhabited television event. *Presence: Teleoperators & Virtual Environments* 10, 1 (2001), 35–50.
[29] Jan Gugenheimer, David Dobbelstein, Christian Winkler, Gabriel Haas, and Enrico Rukzio. 2016. FaceTouch: Touch Interaction for Mobile Virtual Reality. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems.* ACM, 3679–3682.
[30] Jan Gugenheimer, Evgeny Stemasov, Julian Frommel, and Enrico Rukzio. 2017. Sharevr: Enabling co-located experiences for virtual reality between hmd and non-hmd users. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* ACM, 4021–4033.
[31] Jan Gugenheimer, Dennis Wolf, Eythor R Eiriksson, Pattie Maes, and Enrico Rukzio. 2016. Gyrovr: Simulating inertia in virtual reality using head worn flywheels. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology.* ACM, 227–232.
[32] Yoshinobu Hagiwara. 2015. Cloud based VR system with immersive interfaces to collect multimodal data in human-robot interaction. In *Consumer Electronics (GCCE), 2015 IEEE 4th Global Conference on.* IEEE, 256–259.
[33] Daniel Hepperle and Matthias Wölfel. 2017. Do you feel what you see?: multimodal perception in virtual reality. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology.* ACM, 56.
[34] Eric Hodgson and Eric Bachmann. 2013. Comparing four approaches to generalized redirected walking: Simulation and live user data. *IEEE transactions on visualization and computer graphics* 19, 4 (2013), 634–643.
[35] Andreas Holzinger. 2005. Usability engineering methods for software developers. *Commun. ACM* 48, 1 (2005), 71–74.
[36] Felix Hülsmann, Julia Fröhlich, Nikita Mattar, and Ipke Wachsmuth. 2014. Wind and warmth in virtual reality: implementation and evaluation. In *Proceedings of the 2014 Virtual Reality International Conference.* ACM, 24.
[37] Wolfgang Hürst, Ferdinand de Coninck, and Xhi Jia Tan. 2016. Complementing Artworks to Create Immersive VR Museum Experiences. In *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology.* ACM, 34.
[38] Hendrik Iben, Hannes Baumann, Carmen Ruthenbeck, and Tobias Klug. 2009. Visual based picking supported by context awareness: comparing picking performance using paper-based lists versus lists presented on a head mounted display with contextual support. In *Proceedings of the 2009 international conference on Multimodal interfaces.* ACM, 281–288.
[39] Shahidul Islam, Bogdan Ionescu, Cristian Gadea, and Dan Ionescu. 2016. Full-body tracking using a sensor array system and laser-based sweeps. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on.* IEEE, 71–80.
[40] Naoya Isoyama, Tsutomu Terada, and Masahiko Tsukamoto. 2015. An evaluation on behaviors in taking photos by changing icon images on head mounted display. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers.* ACM, 985–990.
[41] Bret Jackson and Daniel F Keefe. 2016. Lift-off: Using reference imagery and freehand sketching to create 3d models in vr. *IEEE transactions on visualization and computer graphics* 22, 4 (2016), 1442–1451.
[42] Hyung-il Kim and Woontack Woo. 2016. Smartwatch-assisted robust 6-DOF hand tracker for object manipulation in HMD-based augmented reality. In *3D User Interfaces (3DUI), 2016 IEEE Symposium on.* IEEE, 251–252.
[43] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33, 2004 (2004), 1–26.
[44] Alexandra Kitson, Abraham M Hashemian, Ekaterina R Stepanova, Ernst Kruijff, and Bernhard E Riecke. 2017. Comparing leaning-based motion cueing interfaces for virtual reality locomotion. In *3D User Interfaces (3DUI), 2017 IEEE Symposium on.* IEEE, 73–82.
[45] Andreas Knote, Sarah Edenhofer, and Sebastian Von Mammen. 2016. Neozoa: An immersive, interactive sandbox for the study of competing ant species. In *K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR), IEEE Virtual Reality Workshop on.* IEEE, 5–10.
[46] Kazuhiko Kobayashi, Koichi Nishiwaki, Shinji Uchiyama, Hiroyuki Yamamoto, and Satoshi Kagami. 2007. Viewing and reviewing how humanoids sensed, planned and behaved with mixed reality technology. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on.* IEEE, 130–135.
[47] Takafumi Koike. 2014. Measurements of operating time in first and third person views using video see-through HMD. In *Proceedings of the 2nd ACM symposium on Spatial user interaction.* ACM, 139–139.
[48] George-Alex Koulieris, Bee Bui, Martin S Banks, and George Drettakis. 2017. Accommodation and comfort in head-mounted displays. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 87.
[49] Seunghae Lee. 2010. Understanding wayfinding for the elderly using VR. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry.* ACM, 285–288.
[50] Xia Sheng Lee, Mohd Faris Khamidi, Zi Siang See, Tim John Lees, and Changsaar Chai. 2016. Augmented reality for ndimensional building information modelling: Contextualization, Customization and Curation. In *Virtual System & Multimedia (VSMM), 2016 22nd International Conference on.* IEEE, 1–5.
[51] Jia-Wei Lin, Ping-Hsuan Han, Jiun-Yu Lee, Yang-Sheng Chen, Ting-Wei Chang, Kuan-Wen Chen, and Yi-Ping Hung. 2017. Visualizing the keyboard in virtual reality for enhancing immersive experience. In *ACM SIGGRAPH 2017 Posters.* ACM, 35.
[52] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. 360 Video Viewing Dataset in Head-Mounted Virtual Reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference.* ACM, 211–216.
[53] Paul Lubos, Gerd Bruder, Oscar Ariza, and Frank Steinicke. 2016. Ambiculus: LED-based low-resolution peripheral display extension for immersive head-mounted displays. In *Proceedings of the 2016 Virtual Reality International Conference.* ACM, 13.
[54] Andrés Lucero, Kent Lyons, Akos Vetek, Toni Järvenpää, Sean White, and Marja Salmimaa. 2013. Exploring the interaction design space for interactive glasses. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems.* ACM, 1341–1346.
[55] Hao Luo, Tien-Szu Pan, Jeng-Shyang Pan, Shu-Chuan Chu, and Bian Yang. 2015. Development of a Three-Dimensional Multimode Visual Immersive System With Applications in Telepresence. *IEEE Systems Journal* (2015).
[56] Claus B Madsen, Jacob Boesen Madsen, and Ann Morrison. 2012. Aspects of what makes or breaks a museum ar experience. In *Mixed and Augmented Reality (ISMAR-AMH), 2012 IEEE International Symposium on.* IEEE, 91–92.
[57] Jacob B Madsen and Claus B Madsen. 2013. An interactive visualization of the past using a situated simulation approach. In *Digital Heritage International Congress (DigitalHeritage), 2013*, Vol. 1. IEEE, 307–314.
[58] Diako Mardenbegi and Pernilla Qvarfordt. 2015. Creating gaze annotations in head mounted displays. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers.* ACM, 161–162.
[59] Florin Octavian Matu, Mikkel Thøgersen, Bo Galsgaard, Martin Møller Jensen, and Martin Kraus. 2014. Stereoscopic augmented reality system for supervised training on minimal invasive surgery robots. In *Proceedings of the 2014 Virtual Reality International Conference.* ACM, 33.
[60] Roderick McCall, Benoît Martin, Andrei Popleteev, Nicolas Louveton, and Thomas Engel. 2015. Text entry on smart glasses. In *Human System Interactions (HSI), 2015 8th International Conference on.* IEEE, 195–200.
[61] Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems* 77, 12 (1994), 1321–1329.
[62] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. 2000. Automatic personalization based on web usage mining. *Commun. ACM* 43, 8 (2000), 142–151.
[63] Florian Müller, Sebastian Günther, Azita Hosseini Nejad, Niloofar Dezfuli, Mohammadreza Khalilbeigi, and Max Mühlhäuser. 2017. Cloudbits: supporting conversations through augmented zero-query search visualization. In *Proceedings of the 5th Symposium on Spatial User Interaction.* ACM, 30–38.
[64] Alessandro Mulloni, Hartmut Seichter, and Dieter Schmalstieg. 2012. Indoor navigation with mixed reality world-in-miniature views and sparse localization on mobile devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces.* ACM, 212–215.

[65] Yuki Nakamura, Shinya Yamamoto, Morihiko Tamai, and Keiichi Yasumoto. 2013. Supporting daily living activities using behavior logs and Augmented Reality. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 658–663.

[66] Takashi Okuma, Masakatsu Kourogi, Nobuchika Sakata, and Takeshi Kurata. 2007. Reliving museum visiting experiences on-and-off the spot. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 279–280.

[67] Hiroyuki Osone, Takatoshi Yoshida, and Yoichi Ochiai. 2017. Optimized HMD system for underwater VR experience. In *ACM SIGGRAPH 2017 Posters*. ACM, 25.

[68] Thammathip Piumsomboon, Arindam Day, Barrett Ens, Youngho Lee, Gun Lee, and Mark Billinghurst. 2017. Exploring enhancements for remote mixed reality collaboration. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*. ACM, 16.

[69] Eric D Ragan, Siroberto Scerbo, Felipe Bacim, and Doug A Bowman. 2017. Amplified head rotation in virtual reality and the effects on 3d search, training transfer, and spatial orientation. *IEEE transactions on visualization and computer graphics* 23, 8 (2017), 1880–1895.

[70] Yashas Rai, Patrick Le Callet, and Philippe Guillotel. 2017. Which saliency weighting for omni directional image quality assessment?. In *Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on*. IEEE, 1–6.

[71] Andrew B Raij and Benjamin C Lok. 2008. Ipsviz: An after-action review tool for human-virtual human experiences. In *Virtual Reality Conference, 2008. VR'08. IEEE*. IEEE, 91–98.

[72] Mikko J Rissanen, Yoshihiro Kuroda, Naoto Kume, Megumi Nakao, Tomohiro Kuroda, and Hiroyuki Yoshihara. 2006. Audiovisual guidance for simulated one point force exertion tasks. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*. ACM, 365–368.

[73] James M Ritchie, Raymond CW Sung, Heather Rea, Theodore Lim, Jonathan R Corney, and Iris Howley. 2008. The use of non-intrusive user logging to capture engineering rationale, knowledge and intent during the product life cycle. In *Management of Engineering & Technology, 2008. PICMET 2008. Portland International Conference on*. IEEE, 981–989.

[74] Vladimir Rubin, Irina Lomazova, and Wil MP van der Aalst. 2014. Agile development with software process mining. In *Proceedings of the 2014 international conference on software and system process*. ACM, 70–74.

[75] Shyam Prathish Sargunam, Kasra Rahimi Moghadam, Mohamed Suhail, and Eric D Ragan. 2017. Guided head rotation and amplified head rotation: Evaluating semi-natural travel and viewing techniques in virtual reality. In *Virtual Reality (VR), 2017 IEEE*. IEEE, 19–28.

[76] Mohit Singh and Byunghoo Jung. 2017. High-definition wireless personal area tracking using AC magnetic field for virtual reality. In *Virtual Reality (VR), 2017 IEEE*. IEEE, 209–210.

[77] Richard HY So, KP Wong, SL Yuen, J Tang, H Yeung, and Junliang Liu. 2011. Virtual reality gaming for rehabilitation: An evaluation study with physio-and occupational therapists. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*. ACM, 503–506.

[78] Oleg Špakov, Poika Isokoski, Jari Kangas, Jussi Rantala, Deepak Akkil, and Roope Raisamo. 2016. Comparison of three implementations of HeadTurn: a multimodal interaction technique with gaze and head turns. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, 289–296.

[79] William Steptoe and Anthony Steed. 2012. Multimodal data capture and analysis of interaction in immersive collaborative virtual environments. *Presence: Teleoperators and Virtual Environments* 21, 4 (2012), 388–405.

[80] Jonathan Steuer. 1992. Defining virtual reality: Dimensions determining telepresence. *Journal of communication* 42, 4 (1992), 73–93.

[81] Wataru Sunayama, Yuki Shibata, and Yoko Nishihara. 2016. Continuation Support of Conversation by Recommending Next Topics Relating to a Present Topic. In *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*. IEEE, 168–172.

[82] Lex Van Velsen, Thea Van Der Geest, Rob Klaassen, and Michael Steehouder. 2008. User-centered evaluation of adaptive and adaptable systems: a literature review. *The knowledge engineering review* 23, 3 (2008), 261–281.

[83] Sebastian Von Mammen, Andreas Knote, and Sarah Edenhofer. 2016. Cyber sick but still having fun. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. ACM, 325–326.

[84] Eric Whitmire, Laura Trutoiu, Robert Cavin, David Perek, Brian Scally, James Phillips, and Shwetak Patel. 2016. EyeContact: Scleral coil eye tracking for virtual reality. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. ACM, 184–191.

[85] Stefan Wiedenmaier, Olaf Oehme, Ludger Schmidt, and Holger Luczak. 2001. Augmented reality (AR) for assembly processes-an experimental evaluation. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*. IEEE, 185–186.

[86] Peter Willemsen, Mark B Colton, Sarah H Creem-Regehr, and William B Thompson. 2004. The effects of head-mounted display mechanics on distance judgments in virtual environments. In *Proceedings of the 1st Symposium on Applied perception in graphics and visualization*. ACM, 35–38.

[87] Betsy Williams, Matthew McCaleb, Courtney Strachan, and Ye Zheng. 2013. Torso versus gaze direction to navigate a ve by walking in place. In *Proceedings of the ACM Symposium on applied perception*. ACM, 67–70.

[88] Betsy Williams, Gayathri Narasimham, Tim P McNamara, Thomas H Carr, John J Rieser, and Bobby Bodenheimer. 2006. Updating orientation in large virtual environments using scaled translational gain. In *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*. ACM, 21–28.

[89] Betsy Williams, Travis Rasor, and Gayathri Narasimham. 2009. Distance perception in virtual environments: a closer look at the horizon and the error. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization*. ACM, 7–10.

[90] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 38.

[91] Trinette Wright, Sandrine de Ribaupierre, and Roy Eagleson. 2017. Design and evaluation of an augmented reality simulator using leap motion. *Healthcare technology letters* 4, 5 (2017), 210.

[92] Catherine A Zanbaka, Benjamin C Lok, Sabarish V Babu, Amy Catherine Ulinski, and Larry F Hodges. 2005. Comparison of path visualizations and cognitive measures relative to travel technique in a virtual environment. *IEEE Transactions on Visualization and Computer Graphics* 11, 6 (2005), 694–705.

[93] Qinping Zhao. 2009. A survey on virtual reality. *Science in China Series F: Information Sciences* 52, 3 (2009), 348–400.

PUBLICATION III

Lightweight Visualization and User Logging for Mobile 360-degree Videos

A. Luoto, P. Heino, and Y. You

DOI: 10.1109/SEARIS44442.2018.9180230

Lightweight Visualization and User Logging for Mobile 360-degree Videos

Antti Luoto[*]
Tampere University of Technology

Pietari Heino [†]
Tampere University of Technology

Yu You [‡]
Nokia

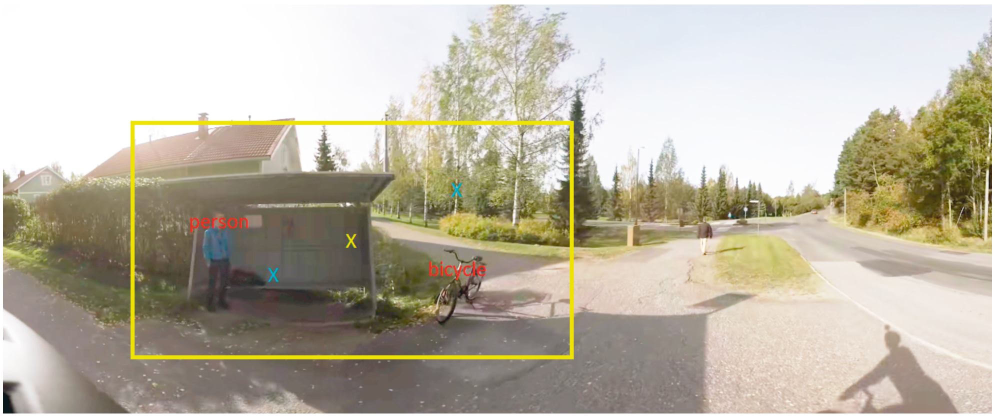

Figure 1: Example of added metadata and playback of previously logged view sessions on 360-degree video. Object detection metadata is shown in red text, current viewport is marked with the light green box, the current head orientation center is marked with the light green X and previously logged view sessions are the blue Xs.

Abstract

360-degree videos are getting more popular also in mobile domain. As the amount of viewers grow, it is beneficial to track what they are doing. We have 360-degree videos with object detection metadata that we want to visualize on the video. At the same time, we are interested in how the users act when watching the videos with added information. Logging the device orientation is one way to do that.

We present a study about a lightweight method to visualize information on top of 360-degree videos while logging the users. The proposed visualization technique is generic and can be used for example to visualize video content related metadata or logging results on top of 360-degree video. We evaluated the work by making a proof of concept and performance analysis, which shows that FPS starts to decrease after around 2000 simultaneous visualization objects. A comparison with other existing visualization solutions suggests that our approach is lightweight.

Index Terms: C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed applications I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Video analysis; D.2.8 [Software Engineering]: Metrics—Performance measures

1 Introduction

360-degree videos are getting more popular due to the recent low-price consumer hardware [9]. From the software developer's perspective, there are multiple SDKs and multiple platforms providing 360-degree video support. Finding out which platform and software development kit (SDK) offer the best combination for one's use cases can require a lot of effort [29]. Still, many developers share common requirements and needs. User logging in 360-degree video environment is an important feature for multiple parties and applications [13,27,28,38]. In addition, many 360-degree video application require only lightweight and simple graphics [19], meaning that usage of heavy 3D engines would be overhead.

In this paper, we present a study about lightweight way to add textual information on top of 360-degree videos taking into account the device orientation and logging it simultaneously. The visualized information can be, for example, a result of video content analysis or it can originate from user log analytics. The selected platform, Google VR Cardboard/Daydream, is a popular choice for 360-degree video applications and it offers an SDK for Android. It is also an example of cheap consumer platform that we are aiming at. Figure 1 shows an example of the application's functionality. Our visualization technique uses overlapping transparent nested Android layout elements. The technique is lightweight and without a need for writing low-level native code or using bloated graphics libraries or engines. Saving resources is important with mobile devices to save battery power. The solution includes a backend for storing the user logs and analyzing them.

Logging, in turn, is important because user tracking in 360-degree video domain has high demand. By gathering data of the users' behavior, one is able to do a statistical analysis. 360-degree video domain is also a relatively new research field so it is not obvious how masses of users actually behave when watching those videos.

The main reason for combining visualization and logging in this paper is that they are relatively important features and they can be used in our future work. The future applications for the presented ideas include automatic annotation of the regions of interest, predicting user's device rotation, broadcasting only the needed part of the video in high quality, heatmap analysis of the field of view, improv-

[*]e-mail: antti.l.luoto@tut.fi
[†]e-mail: pietari.heino@tut.fi
[‡]e-mail: yu.you@nokia.com

ing user experience, or adding collaboration to watching 360-degree videos.

Our hypothesis is that the proposed solution is lightweight, meaning that it can handle a large amount (thousands) of simultaneous visualization objects without significant performance reduction. The evaluation was done by implementing a proof of concept where we first logged users watching 360-degree video with visualized metadata, after which we visualized the session traces on top of the same video. Further evaluation was made by measuring the performance while increasing the amount of hotspots. The measurements show that FPS starts to decrease at around 2000 hotspots. Such amount of simultaneous hotspots should be enough for multiple visualization applications. When compared to Unity 3D, Google VR Web View and KRPano, our solution performs better. The comparison with Unity 3D suggests that our solution's CPU and memory usage are significantly lower.

The structure of the paper is as follows. Section 2 forms the theoretical background for the work. Section 3 presents selected related work from the viewpoints of visualization and logging. Section 4 tells the implementation details. Section 5 presents our demonstrations and performance measurements. Section 6 discusses the flaws of the solution. Section 7 explains our future plans especially from the log analysis perspective. Section 8 concludes the paper.

2 Background

360-degree videos, also known as spherical videos, are videos that show omnidirectional visual information about the surroundings of the observer as opposed to traditional videos that show information only from a single fixed direction. They have applications in multiple domains such as education, entertainment, industry and robotics. The video needs to be in planar format to be played with most of the video players and there are many proposed sphere-to-planar mappings: equirectangular projection, cubemap projection, tile segmentation scheme, rhombic dodecahedron map, etc. [16]. In this paper, we use equirectangular projection, which is widely supported by video players [16]. *Yaw, pitch and roll* [3] is one format of expressing rotations in three dimensional space, also used in 360-degree video domain [16].

Google VR [21] offers an SDK for developing 360-degree video applications for Google Cardboard and Daydream platforms. The platform is designed so that a mobile phone works as a display that is attached to a headband worn by the user. Together the phone and the headband form a head mounted display (HMD). When user rotates his/her head, the video will rotate accordingly using the phone's orientation sensors. However, 360-degree video applications can be used as handheld without a HMD as well which is also supported by the SDK.

Google VR SDK is freely available for multiple platforms (iOS, Unity 3D, Unreal, Web) but we concentrate on Android SDK which offers Java API. While fully commercial VR systems have their advantages [6], Google VR is a relatively popular and cheap solution [9].

Google VR SDK for Android comes with a native development kit (NDK), with which a skilled (C and C++) developer is less restricted. On the other hand, the abstraction level can be risen on a higher level since the SDK can be integrated to a popular Unity 3D game engine, which can help developing advanced applications. Google VR takes simplicity into account but does not restrict advanced users, which is seen as one of the requirements for VR development environments [6]. Still, Unity 3D is a relatively heavy and complex utility also requiring special knowledge and consuming more device resources [26]. The resource consumption should be taken into account when developing mobile applications because battery life has not increased as fast as power demand [14].

Hotspots can be used displaying different kinds of information in 360-degree videos [8]. For example, in Google VR Web View hotspots are interaction points often connected to some kind of functionality whereas in KRPano hotspots can be interaction points or plain textfields. One use for hotspots is to display short texts inside videos [8]. Unfortunately Google VR SDK for Android does not support such functionality (though hotspot support is provided in Google VR Web View). We use hotspots (text tags) to mark points of interest generated by video analysis algorithm. While the usage of hotspots might not be convenient in all our planned future work, so far it has been a useful way to prove the analysis metadata format functional and also to do various visualization experiments. Also, the placement of hotspots on a video as a function of time has been considered a challenge [8]. Such functionality can be supported with our approach.

In general, software users have been tracked for various reasons. For example, runtime traces have been analyzed for improving architecture and performance in addition to improving design and usability in agile software development style [35]. Multiple authors [7, 12, 30] have written about tracking and analyzing users in web. Mobasher et al. [36] categorize three types of data sources for web usage mining: server, proxy and client. While we are not only in the domain of web applications (we don't use browser), we log on the client side.

Watchers of videos can be naturally tracked for various reasons. For example, by recording the user activity in 360-degree video, we can conduct analysis on the user's behavior [28]. There are 360-degree video research cases where head tracking traces have been used for user prediction or video quality assessment [16]. Also, when logging 360-degree video usage, it is good to note the difference between logging the head orientation and eye orientation [28]. While head orientation provides information about the watched viewport, eye tracking can tell where the user is concentrated inside that viewport. We acknowledge that eye tracking has multiple applications, for example in usability development [20] and UX work [5], and it is getting increasingly important since the consumer devices are starting to support it. Still, our research only covers the information that comes from the head (or device) orientation because our focus is on platforms that primarily do not support eye tracking. Still, parts of our work are applicable to eye tracking as well.

3 Related Work

Recently there have been multiple studies about on publishing open 360-degree head movement datasets. While we are not providing a public dataset, we suggest an alternative method to collect user data with aim in simultaneous multi-user data collection and near real-time data analysis. Our logging has been aimed, so far, on handheld mobile usage instead of HMD usage.

There seems to be relatively few scientific publications about visualizing metadata on top of 360-degree videos. Therefore, from the visualization perspective, the related work presents topics such as adding hotpots in panoramic videos and using graphics in AR.

3.1 Logging

Lo et al. [27] offer public datasets made with their testbed based on Oculus Rift and open source libraries such as GamingAnywhere and OpenTrack. In contrast, our logging server architecture enables a way to collect datasets simultaneously with multiple users. As an addition to the data logged by them, we include accelerometer sensor values and viewport size.

Corbillon et al. [13] released a public 360-degree video head movement dataset. Their technology is based on Open-Source Virtual Reality (OSVR) HMD. They log head orientation using quaternion format whereas we use yaw and pitch. Similarly to our work, they generate a log entry when a new frame is drawn to the device.

Wu et al. [38] made a dataset for exploring user behavior in spherical videos. The technology is based on HTC Vive and Unity 3D. The head orientation is logged by using quaternions and the position of HMD in Unity space. They also present visualizations and statistics with their dataset. As opposed to Unity 3D usage, our approach is assumedly more lightweight and thus not consuming as much resources.

3.2 Visualization

Kwiatek and Woolner [25] present a panoramic video solution where hotspot information can be added using XML files. KRpano [24] also uses XML for adding hotspots in panoramic views. Chiang et al. [11] present a VR SDK with hotspot management where it seems that the hotspots are used for creating transitions between linked scenes. While we are not directly discussing about interaction with hotspots, our technique enables a way to implement that easily. Especially, with the help of object detection metadata, it is easy to create hotspots that follow the wanted location spatially and temporally without a manual work.

Gammeter et al. [19] present an AR solution that implements server-side object recognition and client-side object tracking. Their implementation uses the device's sensors and overlays text labels on live camera feed. Their challenge is to send footage from the client to the server for the object detection, while we can do the object detection before the video reaches the client. In addition, the tracking can be simpler for us because we only need to trust the coordinates coming from the object detection algorithm since the user is more restricted and cannot move freely in 3D space.

Wagner et al. [37] and Ferrari et al. [18] use overlaying 3D graphics in AR whereas our approach enables usage of simple and lightweight graphics. Our work is inspired by the thought, that while simple 2D graphics do not take 3D space fully into account, they come with the benefit of efficiency and they are sufficient for many applications [19].

4 Proof of Concept

Our framework consists of three main components: video analysis platform, video player and log server. The video analysis platform is a server capable of taking videos as input and running specified algorithms on the video. The video player is a mobile phone application capable of playing equirectangular 360-degree videos with visualized metadata in addition to collecting the relevant user data simultaneously. The logging server stores the data (log) received from the player and provides a way to analyze them. It also makes the logs and analysis results accessible so that the video player can use them. The architecture is distributed (all the components can be running on separate nodes) and the 360-degree video player can communicate with the other components via REST APIs using client-server model. Figure 2 summarizes and visualizes the components.

4.1 Video Analysis Platform

The video analysis platform has been reported in detail in [22]. In this paper, we concentrate on other parts of the framework. However, to summarize, it is a platform for executing analysis algorithms on videos so that user can send a video and specify which algorithms are executed. Then the analysis results – in other words metadata – may be retrieved via REST API in JSON format. The platform server itself is implemented with Node.js and the algorithms can be implemented basically with any programming language since they have a simple Node.js wrapper providing interface for internal communication.

The analysis results can be, for example, object detection metadata produced by YOLO9000 algorithm [34]. The metadata contains information such as rectangular location coordinates of the object, size of the object, timestamp, confidence score between zero and one, and name of the classified element such as 'car' and used classification dataset such as 'coco'. The following JSON snippet is an example of two simultaneous detections:

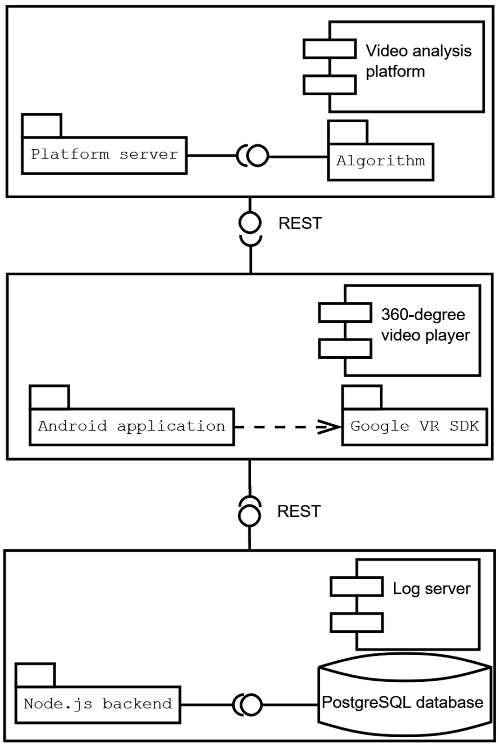

Figure 2: Architecture of the framework.

[{"timestamp":0,
"classification":{"name":"coco","class":"person"},
"score":0.602483,
"rect":{"x":3095,"y":1536,"width":1349, "height":678}},
{"timestamp":0,
"classification":{"name":"coco","class":"bicycle"},
"score":0.312036,
"rect":{"x":2597,"y":1531,"width":1044,"height":372}}]

Thus, the metadata can contain information that there is a person located in coordinates 100 on x-axis and 200 on y-axis with a recognition score around 0.75 at the time of one second. Therefore, it is possible to place a hotspot labeled 'person' in those coordinates on that time when playing the video in a spherically decoding player.

4.2 Video Player

We are using Google VR SDK (version 1.40.0) for Android and the SDK's class VrVideoView for adding a 360-degree video player in Android application. The SDK allows adding 360-degree video player as a view layout element and offers a basic interface for playing videos and managing the player. While the class is helpful in many ways, the SDK does not offer everything a developer might need. For example, it does not directly support adding hotspots on panoramic photos and videos. Because of that, we tried to find a lightweight way to visualize text or simple graphics on top of 360-degree videos. As our development device, we used Moto Z Droid mobile phone running Android operating system version 7.0. With the proof of concept, we used video files from local file system, but

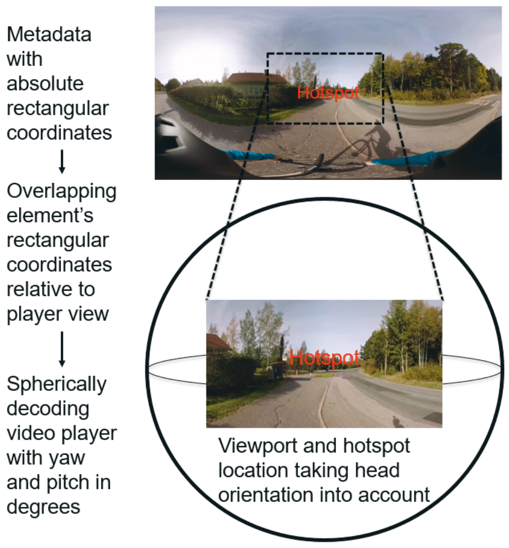
Metadata with absolute rectangular coordinates
Overlapping element's rectangular coordinates relative to player view
Spherically decoding video player with yaw and pitch in degrees
Hotspot
Hotspot
Viewport and hotspot location taking head orientation into account

Figure 3: Summary of the required coordinate system conversions.

they could be as well provided by the video analysis platform.

4.3 Visualization Technique

We use overlapping nested layout elements to add text and other simple graphics on 360-degree video. While the layout can be defined in a declarative way with XML files, our method primarily supports programmatical way. Adding a new TextView, which is a basic UI element for showing text, as a child to VrVideoView can be done easily and it allows creating multiple children overlapping with each other. The background of TextView is transparent which makes it convenient for overlapping.

VrVideoView retrieves the device's orientation in yaw and pitch format. Yaw is the vertical angle between -180° and 180°, and the pitch is the horizontal angle between -90° and 90°. Roll is not available via the API but we found a way to calculate it using the devices sensors.

The challenge of adding graphics, moving according to the device rotation, comes from the different coordinate systems in the metadata, in the video player, and in the 2D plane placed on top of the video. Figure 3 summarizes the needed conversions.

Since the video analysis metadata contains coordinates in rectangular pixel format, they need to be converted so that they are useful for the video player using degrees in spherical space. The conversion with monocular equirectangular panorama video can be calculated with Equations 1 and 2

$$\alpha_x = ((w/v_w)x_v - x_0) * 180°/(w/2) \tag{1}$$

$$\alpha_y = (y_0 - (h/v_h)y_v) * 90°/(h/2) \tag{2}$$

where w and h are the width and height of the video player, v_w and v_h are the width and height of the original video, x_v and y_v are the original rectangular coordinates, and x_0 and y_0 are the center point of the layout element overlaying the video player.

Now that the location is known in yaw (α_x) and pitch (α_y) format, the corresponding location of the overlapping element (TextView in this case) on the 2D plane can be calculated by using the length of arc for both the angles. The equations are

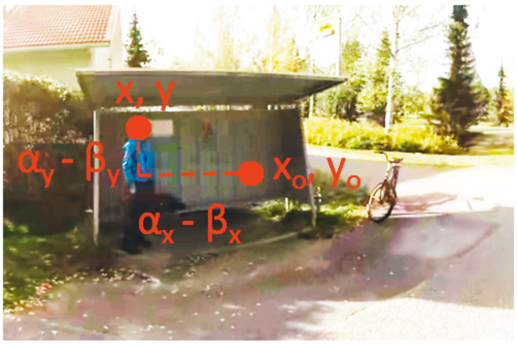

Figure 4: Calculating the location of the hotspot (x,y) when it's pitch (α_x) and yaw (α_y) are known. The person acts as a hotspot in the figure. The viewer is looking at point β_x and β_y which is located in the center of the layout element marked by the point x_0, y_0. Thus, the difference of the hotspot location and viewport center can be calculated.

$$x = x_0 + 2\pi R * ((\alpha_x - \beta_x)/360°) \tag{3}$$

$$y = y_0 - 2\pi R * ((\alpha_y - \beta_y)/360°) \tag{4}$$

where x_0 and y_0 are the center point of the video player screen, R is the depth of the video in the player, α_x are α_y are the angles for the location of the object, and β_x and β_y are the angles for the current device rotation. The resulting x and y define the hotspot location. Figure 4 visualizes the equations. In VrVideoView, R is half of the screen width when the video's aspect ratio corresponds the player's aspect ratio. The overlapping TextView element has an origin in the left upper corner of the video player screen so that x increases towards the right-hand side and y increases towards the bottom.

Equation 3 needs modifications when the hotspot appears on the other side of the vertical border of the rectangular video than the center of the user's viewpoint. Figure 5 visualizes the situation. In other words, if the user is looking between the yaw angles -180° and -135°, and the hotspot is located between the angles 135° and 180°, the equation needs to be in format

$$x = x_0 - 2\pi R * ((-\alpha_x - \beta_x + 360°)/360°) \tag{5}$$

or respectively, if the user is looking between 135° and 180°, and the hotspot is located between -135° and -180°, the format is

$$x = x_0 - 2\pi R * ((\alpha_x - \beta_x - 360°)/360°) \tag{6}$$

where the symbols have the same meaning as in Equation 3.

The timestamp in metadata enables a way to synchronize the metadata with the video. Metadata provides timestamp in milliseconds that is also the timestamp format available via Google VR SDK.

4.4 Logging

For every frame of the video, the application makes a log entry that contains data such as yaw, pitch, roll, video time, viewport (seen area) size, accelerometer measurement values and frame number. Video content metadata related information could be logged as well. Detailed specification about the most important logged records can be seen in Table 1.

Table 1: Logged records with descriptions and reasoning.

Log record	Origin	Description	Why
yaw	VrVideoView	Vertical angle between -180° and 180°.	Identifying device orientation.
pitch	VrVideoView	Horizontal angle between -90° and 90°.	Identifying device orientation.
roll	Sensors	Clockwise or counter clockwise rotation between -180° and 180°.	Identifying device orientation.
accelerometer	Sensors	X, y and z accelerometer values in m/s.	Predicting device orientation.
video time	VrVideoView	Current time of the video in milliseconds.	Storing the device orientation in time.
viewport size	VrVideoView	Width and height of the viewport in pixels.	Identifying the field of view.
frame number	Generated	Approximation of the current frame number according to a counter in the SDK's onNewFrame method.	Storing additional information to video time.
objects	Metadata	List of classified objects and their location (yaw, pitch) in the field of view.	(Example of) adding video content analysis to user log.

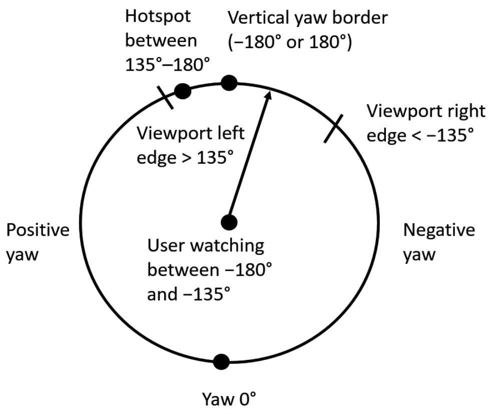
Hotspot between 135°–180°
Vertical yaw border (−180° or 180°)
Viewport left edge > 135°
Viewport right edge < −135°
Positive yaw
Negative yaw
User watching between −180° and −135°
Yaw 0°

Figure 5: A situation where the viewport center point is on the other side of the 180° border of negative and positive yaw, and there is a hotspot on the other side of the border.

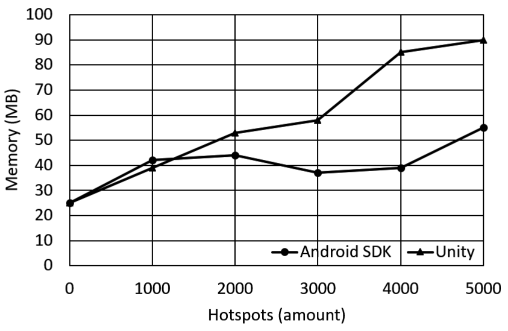

Figure 6: Memory consumption when adding hotspots with Google VR Android SDK and Unity 3D.

The entry is sent with HTTP to the log server that implements a REST interface implemented with Node.js. The data is stored in a relational PostgreSQL database. REST architecture enables a way to log multiple users simultaneously.

Logging the viewport data just once could be enough for many applications but if the viewport size changes during the view session, for example because of turning the handheld phone from portrait orientation to landscape orientation, storing it for each frame can be useful.

5 Evaluation

We evaluated the proof of concept with demonstrations and performance measurements. The used video was 22 seconds long, 20 MB-sized, monocular 360-degree video (MPEG-H Part2 H.265) with resolution 3840x1920, and 30 FPS. The video contains cycling footage. The idea in the demonstrations was that users' view sessions were logged and stored to the backend. After that, we were able to show the trace of the view session over the same 360-degree video. Simultaneously, when watching the video, object detection metadata was visualized on it. The successful demonstrations show that the visualization technique works, the logging is accurate enough, and the approach is promising for further development. The demonstrations and performance measurements were done using Moto Z Droid mobile phone (Quad-core 2x1.8 GHz Kryo & 2x1.6 GHz Kryo, 4 GB RAM) running Android operating system version 7.0.

To compare our solution with existing solutions, we experimented by adding static hotspots (3D texts) to Unity 3D with 360-degree video player in VR mode. We used Google's sample 360-degree video about gorillas. The size of the video used was about 9 megabytes, had resolution of 2048x2048, and 30 FPS (H264, MPEG-4).

Measurements with Unity 3D can be seen in Figures 7, 6 and 8. The Profiler tool of Unity does not show fully comparable results when compared to Android Monitor used with Android SDK. For example, total CPU usage is not visible, so we recorded the process PostLateUpdate.FinishFrameRendering that was the process having the highest CPU load. CPU usage was very volatile so we approximated the average. However, CPU usage was higher in all the measurements when compared to Android SDK. With 2000 hotspots the average FPS decreased clearly under 30.

Measurements with Android SDK can be also seen in Figures 7, 6 and 8.. The measurements were done using Android Monitor tool in Android Studio. CPU usage was very volatile so we recorded the highest peak. FPS, memory usage, and CPU usage reduced slower when compared to Unity 3D. We could not experience a difference in performance with Android SDK when the visualizations were disabled or when using up to 2000 simultaneous hotspots. It is not clear to us why CPU usage increases relatively slowly after 3000 hotspots. We do not know either why memory usage decreases after 3000 hotspots but then increases again. The reason might be something related to Android memory management.

In addition, we experimented with Google VR View for the Web. Adding only 1000 hotspots over 360-degree video running in

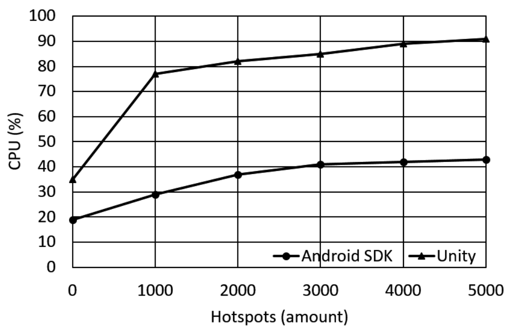

Figure 7: CPU usage when adding hotspots with Google VR Android SDK and Unity 3D.

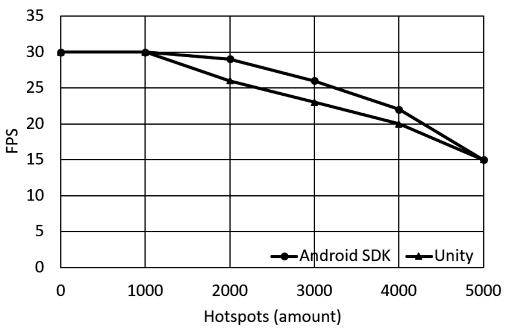

Figure 8: FPS when adding hotspots with Google VR Android SDK and Unity 3D.

Chrome browser on Android (Moto Z) resulted in 23 FPS. Another experiment was made with KRPano (using Textfield plugin) and it took well over two minutes for 500 static hotspots to even appear in a browser using a modern laptop (Lenovo W541).

Logging generates quite a lot of data because the video frame updates dozens of times in a second and the videos can be long. One second of logging generates about 12–13 kilobytes of log which makes over 700 kilobytes in a minute.

6 Discussion

Our technique of using nested layout elements for visualization on Android works with widget styled applications. Thus, we need to investigate if a similar technique can be used in the VR mode of Google VR to improve HMD support. The usage of TextView supports only adding simple graphics in text format but another type of view element could be used as well to provide other kind of graphics.

Using nested layout elements in Android might not be performance-wise if there are a lot of nested elements. However, in our case the hierarchy stays flat which does not weaken the performance significantly [1].

The accuracy of the visualization technique depends also on the amount of distortion in the video. The distortion increases when going away from the center point of the viewport. Naturally, a bigger sized hotspot allows more inaccuracy when compared to smaller one highlighting a smaller detail. Thus, the technique works if approximations are allowed. In addition, distortion can be a problem near the polar areas.

Occulusion naturally happens with hotspots on top of video. It could be reduced by using transparency or allowing user to enable or disable them. Text font, size and color could be improved for better readability. However, we have not investigated these aspects so far, while they are important usability and user experience aspects.

Another flaw of the visualization is the lack of taking roll angle into account. Now rolling of the device makes the hotspots move out from their correct place. The main reason for not taking the roll into account is that the SDK's class VrVideoView does not provide it at the moment and the proof of concept works relatively well without. While it is possible to calculate the roll by using the device's accelerometer and magnetometer sensors [2], a proper filtering is also needed since the signals coming directly from the sensors are often noisy [19]. On the other hand, logging the roll is simpler than using it with the visualization.

While our hotspots do not enable interaction, adding such functionality would be a relatively easy task. For example, keeping the center of the view near the hotspot long enough (gazing it), or a touch event via the screen, could trigger an interaction. In addition, the metadata coming with the videos could be supported better. For example, if the metadata contains information about the size of the detected object, the visualization could make use of it.

Analyzing the logs reveals that the video time for the logging events is different between the logging sessions. For example, the first three recorded times for a view session can be 21, 34 and 64 milliseconds while the next log has 21, 165 and 175 milliseconds. This happens because the VR SDK's onNewFrame (triggers a logging event) is executed only for drawn frames. Still, the times are in milliseconds that makes the error relatively small and allows using the logged values in various analytics. This makes the logged frame numbers approximations, because as seen in the example logs, the frame number two has a different timestamp (34 and 165). Therefore, the logged video time is more reliable than the frame number.

The meaningfulness of comparing Unity 3D and our solution can be questioned since they have a different set of features. Unity 3D is more general and complex while we work on relatively small Android domain with simple features. We chose Unity 3D as our primary comparison subject because it is popular in the field, it is relatively well documented, and it is easily available. We agree that there could be systems more close to our work but we did not have access to them. Also, comparing to the ones found in related work have quite similar problems as comparison with Unity 3D – they have a different set of features. Additionally, we made smaller experiments with Google Web VR and KRPano. To make the comparison more trustworthy, more videos and more experiments would be needed.

Further, we cannot think of many use cases for thousands of simultaneous hotspots. Therefore, the difference between 2000–3000 hotspots might not be very relevant. However, to see clearly the performance difference between different approaches, we decided to use a relatively great amount of hotspots in our experiments.

7 Future Work

Low resource usage can be seen as one of the essential aspects of 360-degree video development [29]. Message queueing telemetry transport (MQTT) [31] is a lightweight publish-subscribe protocol that uses less resources (for example memory and CPU) when compared to HTTP on Android [32]. The logging data could be sent with MQTT, because we do not necessarily need the responses of HTTP, which should help saving resources of mobile devices. MQTT is also a binary-based protocol so using it should use less resources when compared to text based HTML. In addition, MQTT helps communicating to servers behind NAT or vice versa [4]. Another way to save resources could be to send the logs as batches, which on the other hand, could weaken the collaborative and near real-time aim of the study.

Defining the depth of an object in the video is a challenge. Without a knowledge of the depth, it needs to be manually approximated when adding hotspots. Luckily, methods for approximating the depth exist [17, 23], and with such information, making a hotspot which size changes in relation to the distance would be possible.

So far, we have made only simple analysis of the logged data. However, in the future, we plan to use the data for predicting user's head movement for bandwidth usage optimization since 360-degree videos are relatively heavy. For example, the head movement in 360-degree video can be predicted with over 90% accuracy in the short term with methods such as linear regression [33].

Heatmap [15] could be one way to analyze 360-degree video usage. For example, in a heatmap, the parts of the video that are viewed often are highlighted with bright colors whereas the parts viewed seldom have lighter colors. The heatmap visualization can be implemented on the video player client or as service of the logging server having an analysis dashboard. One application for the heatmap analysis would be to place advertisements on the often viewed parts of the video. Heatmap could be more useful when combined with eye tracking, which would require adding eye tracking support to our framework.

Carlier et al. [10] use crowd sourced information to zoom and retarget traditional videos. The idea is that the viewing experience can be improved by automatically providing the most interesting parts of the video. While their implementation works for traditional videos, our framework could be applicable for a similar approach with 360-degree videos in the future.

8 Conclusions

We presented a proof of concept on Android for visualizing metadata on top of the 360-degree video while logging the users simultaneously. The chosen 360-degree video platform was Google VR for Android (Cardboard/Daydream). The technique for superimposing virtual objects is lightweight and works with native Android layout approach.

The proof of concept was evaluated by showing traces of user logs on 360-degree video while displaying object detection metadata on the video simultaneously. The evaluation shows that the presented visualization method works as expected and the framework is ready for further development.

Lightweightness of the technique was evaluated by monitoring memory, CPU, and FPS while increasing the amount of hotspots. With our setting, there was not a notable FPS loss with 2000 hotspots. Low resource usage can be a significant aspect with mobile devices since battery power is often consumed relatively fast.

The combination of logging and visualization in mobile 360-degree video domain has a lot of potential applications. For example, with the help of the proposed solution, it would be interesting to study how to support placing advertisements on the most watched parts of the 360-degree video by analyzing the logged data.

Acknowledgments

We would like to thank Business Finland for funding the work.

References

[1] Android. Optimizing layout hierarchies. https://developer.android.com/training/improving-layouts/optimizing-layout.html, No date. Accessed: 2017-05-03.

[2] Android. Position sensors - computing the device's orientation. https://developer.android.com/guide/topics/sensors/sensors_position.html, No date. Accessed: 2017-05-18.

[3] H. A. Ardakani and T. Bridges. Review of the 3-2-1 euler angles: a yaw-pitch-roll sequence. *Department of Mathematics, University of Surrey, Guildford GU2 7XH UK, Tech. Rep*, 2010.

[4] P. Bellavista and A. Zanni. Towards better scalability for iot-cloud interactions via combined exploitation of mqtt and coap. In *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*, pp. 1–6. IEEE, 2016.

[5] J. R. Bergstrom and A. Schall. *Eye tracking in user experience design*. Elsevier, 2014.

[6] A. Bierbaum and C. Just. Software tools for virtual reality application development. *Course Notes for SIGGRAPH*, 98, 1998.

[7] J. Borges and M. Levene. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Transactions on Knowledge and Data Engineering*, 19(4), 2007.

[8] F. Bota, F. Corno, and L. Farinetti. Hypervideo: A parameterized hotspot approach. In *ICWI*, pp. 620–623, 2002.

[9] A. Brown and T. Green. Virtual reality: Low-cost tools and resources for the classroom. *TechTrends*, 60(5):517–519, 2016.

[10] A. Carlier, V. Charvillat, W. T. Ooi, R. Grigoras, and G. Morin. Crowdsourced automatic zoom and scroll for video retargeting. In *Proceedings of the 18th ACM international conference on Multimedia*, pp. 201–210. ACM, 2010.

[11] C.-C. Chiang, A. Huang, T.-S. Wang, M. Huang, Y.-Y. Chen, J.-W. Hsieh, J.-W. Chen, and T. Cheng. Panovr sdk—a software development kit for integrating photo-realistic panoramic images and 3-d graphical objects into virtual worlds. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 147–154. ACM, 1997.

[12] F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos. Are web users really markovian? In *Proceedings of the 21st international conference on World Wide Web*, pp. 609–618. ACM, 2012.

[13] X. Corbillon, F. De Simone, and G. Simon. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, number EPFL-CONF-227447, pp. 199–204. ACM, 2017.

[14] S. K. Datta, C. Bonnet, and N. Nikaein. Android power management: Current and future trends. In *Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on*, pp. 48–53. IEEE, 2012.

[15] A. T. Duchowski, M. M. Price, M. Meyer, and P. Orero. Aggregate gaze visualization with real-time heatmaps. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 13–20. ACM, 2012.

[16] T. El-Ganainy and M. Hefeeda. Streaming virtual reality content. *arXiv preprint arXiv:1612.08350*, 2016.

[17] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pp. 834–849. Springer, 2014.

[18] V. Ferrari, T. Tuytelaars, and L. Van Gool. Markerless augmented reality with a real-time affine region tracker. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*, pp. 87–96. IEEE, 2001.

[19] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, and L. Van Gool. Server-side object recognition and client-side object tracking for mobile augmented reality. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 1–8. IEEE, 2010.

[20] J. H. Goldberg and A. M. Wichansky. Eye tracking in usability evaluation: A practitioner's guide. *The mind's eye: Cognitive and applied aspects of eye movement research*, 2003.

[21] Google. Google vr. https://developers.google.com/vr/, 2016. Accessed: 2017-05-03.

[22] T. Kalliomäki. Design and Performance Evaluation of a Software Platform for Video Analysis Service. Master's thesis., 2018.

[23] K. Karsch, C. Liu, and S. Kang. Depth extraction from video using non-parametric sampling. *Computer Vision–ECCV 2012*, pp. 775–788, 2012.

[24] KRPano.com. Optimizing layout hierarchies. https://krpano.com/docu/xml/, No date. Accessed: 2017-05-12.

[25] K. Kwiatek and M. Woolner. Embedding interactive storytelling within still and video panoramas for cultural heritage sites. In *Virtual Systems and Multimedia, 2009. VSMM'09. 15th International Conference on*, pp. 197–202. IEEE, 2009.

[26] J. Linowes and M. Schoen. *Cardboard VR Projects for Android*. Packt Publishing Ltd, 2016.

[27] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. 360 video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 211–216. ACM, 2017.

[28] T. Löwe, M. Stengel, E.-C. Förster, S. Grogorick, and M. Magnor. Visualization and analysis of head movement and gaze data for immersive video in head-mounted displays. In *Proceedings of the Workshop on Eye Tracking and Visualization (ETVIS)*, vol. 1, 2015.

[29] A. Luoto. Towards framework for choosing 360-degree video sdk. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017)*, pp. 81–86. SCITEPRESS, 2017.

[30] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.

[31] mqtt.org. Mqtt. http://mqtt.org/, No date. Accessed: 2017-05-31.

[32] S. Nicholas. Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android. http://stephendnicholas.com/archives/1217, 2012. Accessed: 2016-10-05.

[33] 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*, pp. 1–6. ACM, 2016.

[34] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[35] V. Rubin, I. Lomazova, and W. M. van der Aalst. Agile development with software process mining. In *Proceedings of the 2014 international conference on software and system process*, pp. 70–74. ACM, 2014.

[36] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *Acm Sigkdd Explorations Newsletter*, 1(2):12–23, 2000.

[37] D. Wagner, D. Schmalstieg, et al. First steps towards handheld augmented reality. In *ISWC*, vol. 3, p. 127, 2003.

[38] C. Wu, Z. Tan, Z. Wang, and S. Yang. A dataset for exploring user behaviors in vr spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 193–198. ACM, 2017.

PUBLICATION IV

Fighting Network Restrictions of Request-Response Pattern with MQTT

A. Luoto and K. Systä

IET Software 12.5 (2018), pp. 410–417
DOI: 10.1049/iet-sen.2017.0251

IET Software

Research Article

Fighting network restrictions of request-response pattern with MQTT

ISSN 1751-8806
Received on 29th September 2017
Revised 10th February 2018
Accepted on 29th March 2018
doi: 10.1049/iet-sen.2017.0251
www.ietdl.org

Antti Luoto[1] ✉, *Kari Systä*[1]

[1]*Laboratory of Pervasive Computing, Tampere University of Technology, Korkeakoulunkatu 10, FI-33101 Tampere, Finland*

✉ *E-mail: antti.l.luoto@tut.fi*

Abstract: As Internet-of-Things (IoT) devices become more powerful, they can also become full participants of Internet architectures. For example, they can consume and provide RESTful services. However, the typical network infrastructures do not support the architecture and middleware solutions used in the cloud-based Internet. The authors show how systems designed with RESTful architecture can be implemented by using an IoT-specific technology called message queuing telemetry transport (MQTT). Their example case is an application development and deployment system that can be used for remote management of IoT devices. To evaluate the proposed solution, they performed resource consumption experiments to compare HTTP and request-response usage of MQTT. The results suggest that MQTT uses less central processing unit time and memory.

1 Introduction

We assume that when the devices in the Internet of Things (IoT) become more powerful and capable of performing complex tasks, the execution will move towards the edge devices. This means the devices become programmable and participate in rich interactions with other peers on the Internet.

However, the network and system architectures of IoT systems are usually constrained and optimised for minimal consumption of energy and other resources. One typical constraint is that devices are connected to the local network and that network is connected to the Internet through a gateway. This kind of network configuration imposes constraints, for example, the devices are not directly accessible from services on the Internet.

In this article, we explore compatibility between IoT and Internet architectures. The research was inspired by our existing system [1] that uses the RESTful architecture for device management and the management system has to address devices directly. In this research, we show how the functionality of the original system can be preserved even if the network configuration imposes constraints.

Concretely, this study explores the issues between Internet architectures and constraints in IoT systems by showing how designs based on RESTful Internet architectures can be refactored on top of message queuing telemetry transport (MQTT). The work is based on earlier work where a web-based tool can be used for both development and remote deployment of applications to IoT devices. While evaluating and demonstrating this system, we realised that the devices behind a network gateway could not be accessed by other components. The original architecture of the system is based on RESTful style and uses HTTP in all communications. Depending on the task, any component – including devices – of the system may either offer or consume resources. This architecture could not work under the network constraints.

MQTT is a lightweight protocol designed for device-to-device communication in IoT environments. MQTT uses publish–subscribe pattern for the communication and a centralised broker handles all subscriptions and message deliveries. Due to that communication is limited to devices sending messages to the broker and the broker forwards the messages to active subscriptions. This design is convenient in the constraining network configurations since only the broker needs to be accessible by all the components. In addition, MQTT was assumed to use fewer resources than HTTP used by REST.

Our research question is that how a REST-based system can be refactored to use MQTT in network configurations where the real internet protocol (IP) addresses of all components are not visible to the other peers in the system. The sub-questions include how big changes are needed in the implementation and what are the performance characteristics of the two options. The core technical challenge is how to convert the request-response pattern assumed by REST to the publish-subscribe pattern of MQTT version 3.1.1.

The proposed solution uses separate response messages and a flexible MQTT topic hierarchy with specific request and response topics. The solution is designed for an IoT system consisting of multiple devices and it fits our case with a relatively small amount of code refactoring. Additionally, we compared the resource consumption of request-response styled MQTT usage with HTTP. The comparison suggests that MQTT uses less central processing unit (CPU) time and memory.

The rest of the paper is structured as follows. In Section 2, we describe the problem domain and introduce the basics of MQTT. In Section 3, we compare our work with the work performed by others. In Section 4, we discuss a solution to the mismatch of MQTT and REST and present our proof of concept. In Section 5, we evaluate the work with a source code example and describe the resource usage experiments. In Section 6, we discuss the usefulness of the solution. Finally, in Section 7, we provide some concluding remarks and thoughts for future work.

2 Background

2.1 Realistic IoT architectures

According to Datta *et al.* [2], a recent literature about IoT architectures fails to address real life problems related to 'device and service discovery, controlling endpoints from mobile clients and interaction with endpoints through standard protocol'. To overcome this challenge, Datta *et al.* proposed a concept of 'realistic IoT architecture'. It consists of machine-to-machine (M2M) devices and endpoints, a wireless gateway with web services and mobile clients. In such architectures, the network elements, like firewalls and network address translations (NATs), impose restrictions on applications. Almost every organisation and private home user connected to the Internet has some kind of firewall and NAT today.

Firewall is a security mechanism for filtering, validating and blocking the network traffic. It protects the devices by separating them from the other parts of the Internet. NAT is a method, implemented in firewall or gateway, which converts IP address to another to save IP address space and to increase security by hiding the local network from the public one. On the other hand, NAT also causes restrictions in cases where the services in the Internet or backend need to connect devices in the local network.

We experienced such problems with our case system that used RESTful HTTP-requests from server-side to devices. This happened, for example, when we demonstrated the system outside of our laboratory network. Some of the devices were located in our research network, but we needed to bring the development interface and some example devices to remote locations. There the connection was usually based on local wireless networks or portable 4G access points. Due to the limitations of those networks, some communications were not possible because our back-end servers could not access the devices at the demonstration locations with HTTP.

There are several ways to bypass the restrictions caused by NAT. The most well-known techniques are based on relay servers and hole punching [3]. For example, Lin *et al.* [4] discuss five NAT traversal techniques used by voice over IP applications. Different NAT traversal techniques also cause load on the gateway devices doing the work.

One option is MQTT [5] that is a common technology in the IoT domain. In this research, we wanted to see if MQTT can solve these problems and if our systems can be made MQTT compatible. Our initial studies also indicated that the changes to the original system would be minimal with MQTT. Thus, we wanted to port our system to use MQTT to ensure that our earlier research results can be applied under realistic network configurations.

2.2 Basics of MQTT

MQTT uses the publish-subscribe communication pattern [5]. This means that senders do not send messages directly to recipients. Instead, the messages are just published for possible receivers. Similarly, the receivers express interest by subscribing to forthcoming messages. MQTT includes a special *broker* component that manages the subscriptions and publishing of the messages. In MQTT, the subscriptions and published message are matched to each other with *topics* which may form hierarchical structures. Topics are constructed so that a slash character separates the different levels in the hierarchy. For example, if `abc` is the first-level topic then `abc/123` is a second-level topic. The subscriber can use wild cards to subscribe to multiple topics in the hierarchy. A special character + is used as a single-level wild card and # character is used as a wild card for multiple levels. For example, a subscriber of topic `abc/+` gets messages sent to topics `abc/123`, `abc/xy`, and to all other topics that start with abc/ and also have only one additional level.

2.3 Research challenges

MQTT is not just a different protocol since it also imposes a different style of distribution. While RESTful applications built on top of HTTP are based on a synchronous request-response paradigm, MQTT is based on the publish-subscribe pattern. Change from request-response to publish-subscribe may potentially lead to big changes in the architecture of the system.

HTTP is a relatively heavy protocol both computationally and from network traffic perspective [6]. MQTT has lower power consumption than HTTP in many cases [7]. This is important in the IoT domain with low-resource devices. Our hypothesis was that MQTT uses less CPU, memory, and energy than HTTP. However, simple porting of a REST-based system on top of MQTT might not use MQTT efficiently; for instance, the number of communication may increase.

With the growing number of IoT devices, scalability needs to be taken into account in the IoT domain. For example, more devices mean more problems when devices in varying networks need to communicate with each other. Since the scale is expected to grow, the IoT domain should consider preparing for open IP networks to prevent problems coming from NAT traversal and mobility [8]. For example, some present IoT platforms support local connectivity but not global connectivity [9]. Since secure global connectivity and open IP networks are not available yet in the large scale, NAT traversal with MQTT takes part in helping with the scalability issues. In addition, when traversing NATs, security should not be forgotten.

3 Related work

3.1 Relation of MQTT to request-response and NAT traversal

Request-response over publish-subscribe architecture is not a new idea and different needs for such applications have been reported [10–12]. However, our case system and the need to overcome problems caused by the network gateways motivated us to investigate the topic again. A draft document by Advancing Open Standards for Information Society (OASIS) [13] acknowledges the lack of built-in request-response pattern in MQTT. They state that it is needed when (i) IoT device reads data from the server or from other device or vice versa, and (ii) when the IoT device needs to set a value in the server or other device or vice versa with a confirmation that the operation was successful. Our case adds a new third need – the logic of the application should be radically changed for MQTT without request-response pattern.

A few studies about the relation of MQTT and REST exist. Collina *et al.* [14] studied a broker that bridges MQTT and REST by exposing MQTT topics as REST resources and vice versa so that it is possible to use MQTT via REST but they do not try to use MQTT similarly to REST. Chen and Lin [15] implemented an MQTT proxy in their REST architecture comparing latency and performance between the protocols. However, they do not discuss how to implement functionality similar to REST with MQTT. Neither of these studies tries to overcome the network architecture restrictions that rise because of NATs or firewalls.

Bellavista and Zanni [16] also acknowledge the usefulness of hierarchical MQTT topics in modelling IoT device trees and MQTT's ability to help with NAT problems. Their aim is to use a combination of MQTT and constrained application protocol (CoAP) to get the best benefits from both since CoAP uses few resources in certain situations.

Uehara [17] presents a general framework for IoT devices where MQTT is used as a communication protocol. They use MQTT for communicating beyond NAT and for request-response traffic. Unfortunately, the details of request-response implementation are a bit unclear.

3.2 Tool tutorials

Some tool tutorials [18–21] describe solutions that use request-response with MQTT. While the techniques presented in the tutorials could be useful for tackling the restrictions imposed by the network architectures, we did not see enough benefits to integrate the presented complex tools or use different programming languages since our aim was in minimal refactoring. However, we briefly summarise their request-response techniques. Reactive Blocks [20] (a visual Java programming environment) and Emitter.io example [21] use request identifiers, unique request-response topics and subscription of hierarchical topic structures with wild cards. Eclipse Kura IoT service gateway [18] instructs to use response codes similar to HTTP codes and places REST verbs to MQTT topic hierarchies. The tutorial of solace systems [19] (a complex messaging middleware) has a different approach of using general reply topics and correlation IDs in message payloads for connecting replies to requests.

3.3 Alternative techniques

Techniques to implement the functionality without MQTT exist as well. HTTP long polling could work but we wanted to use MQTT because it supports IoT better. HTTP is not designed for pushing data from the server to client [7] and HTTP long polling is considered inefficient [22]. Websockets [23] could be another alternative often used in web browsers to create two-way communication. However, WebSockets are not designed for constrained devices and they do not support IoT domain well [24]. The use of native Websocket library could still be one option.

Destination network address translation (DNAT) is an enhanced version of NAT that that allows remote peers to initiate a session with peers in the local network. It also improves network access and reliability issues [25]. To the best of our knowledge, the relation of DNAT and IoT has not been explicitly discussed in the scientific IoT literature. Gateways also need to be configured for DNAT which is not always possible. We needed a solution that works with any network and gateway.

Virtual private networks can be used for NAT traversal but it is not simple [26]. Universal plug and play have also been suggested for NAT traversal. However, it is often considered insecure (and thus disabled in gateway devices) and it is not supported by all NATs [27]. It is a relatively resource consuming technique to be used in constrained IoT devices [28] and it does not work well with nested NATs [29].

Srirama and Liyanage [30] propose a solution based on transmission control protocol hole punching [31] for mobile devices in 3G and 4G networks. The approach could also be applicable in the IoT domain. However, studies suggest that functionality of hole-punching depends on the NAT implementation [32]. Although the use of hole punching for this kind of IoT application calls for future research, in this research we decided to use MQTT instead.

Other IoT protocols than MQTT exist. For example, Open Mobile Alliance Lightweight Machine-to-Machine [33] is an IoT protocol that supports device and application management [34]. LWM2M uses CoAP [35] protocol. CoAP is a lightweight version of HTTP and it is assumed to have the same NAT problems as our original implementation. As with HTTP, port forwarding or particular connection requests can be used with CoAP [36] but several authors [16, 17, 22] mention the benefits of MQTT when communicating beyond NAT.

3.4 Summary

Better support for request-response pattern has been proposed to the next version (5.0) of MQTT by OASIS [37]. The solution combines three main ideas: 'reply-to' topics included in the request, correlation-ID to connect the requests to replies, and unique 'reply-to' topics, but unfortunately, the available material does not provide a detailed explanation about them yet. The built-in request-response seems promising but, in contrast, our solution works with MQTT 3.1.1.

As a summary, several other researchers have worked on the integration of HTTP/REST architectures to MQTT protocol. For example, many parties acknowledge the need of request-response in MQTT, MQTT is connected to HTTP systems in various ways and the benefits of MQTT when communications behind NAT are known. However, there seem to be relatively few scientific publications about using MQTT in request-response (or REST) style or design of MQTT topic hierarchies.

4 Proof of concept

4.1 REST-style request-response with MQTT

When a system that uses HTTP and follows RESTful architectural style is ported on top of MQTT, some problems need to be solved. In our example case, we recognised the following main challenges:

1. *Addressing of the resources*: While in HTTP, the addressing is uniform resource locator (URL)-based and in RESTful architectures, all resources have a unique URL, MQTT clients subscribe to topics.
2. *Request-response architecture*: In HTTP, all sent messages assume a response that includes information of about success or failure, and an optional payload. MQTT messages are one directional without a response.

The addressing problem is reasonable easy to solve since the naming conventions of URLs and MQTT topics are very similar. For example, a URL http://example.com/abc/123 can be mapped to a three-level topic `example.com/abc/123`.

There are a few alternatives to implement the responses. We need a mechanism to connect requests to corresponding replies. Also, the broker's ability to deliver messages efficiently should be taken into account. We reasoned a few options to implement responding. We give two examples in this study:

1. Before sending any request, the caller subscribes to a unique topic for the response to the request it will send next. In this case, the content of the response message consists of a status code and a payload assumed by the application. The topic hierarchies need to be designed so that the topic for the response message can be derived from the request automatically. Unsubscription from the response topic needs to be done so that the number of registered topics in a long-living system does not grow without a limit.
2. Each caller has a generic response topic that it subscribes to, and all responses to that caller are sent to that topic. The respondent needs to know the response topic, the response message needs to include request identification, and the client needs to match the response to the correct request. This option moves a part of the responsibility of directing a response to the correct requester from the broker to the client. If an application creates multiple requests – and they may be active simultaneously, then the matching of the response to correct topic becomes even more complicated. Compared with the previous option, a smaller number of topics is needed and they are not created and removed dynamically.

After analysing the options, we selected the first option since it requires the least modifications to our application code, it uses the broker more efficiently, and since the number of devices in IoT systems is expected to grow, a flexible topic structure is beneficial. The downside is the need for creating response topics dynamically and subsequently a need to remove them dynamically, too.

In the second option, the clients would need to do more work by parsing the payload to connect the requests to responses. Besides, a general topic is considered as an anti-pattern in a high throughput environment [38].

The status code, e.g. 200 indicates success, is core component of the HTTP protocol. In principle, it could be encoded to the topic hierarchy or added to the payload. In our case, adding it to the payload requires few changes to the original source code, but also keeps the topic hierarchy simpler.

4.2 Original system

The present work is based on our earlier research on programmable IoT devices [1]. In that work, we have developed a system for development and deployment of applications to IoT devices. The original system consists of three active components: integrated development environment (IDE), runtime environment and resource registry (RR). The IDE runs on a web browser, and it is used for programming, deploying and managing the applications on devices. The runtime environment is pre-installed on the participating IoT devices and it essentially makes devices small application servers. The runtime environment provides a representational state transfer (REST) API for installing, starting, stopping and removing applications [39]. For example, when a new application is installed on a device at address `example.iot`, the IDE sends a POST request to URL http://example.iot/APP and the payload of the request is the application package in TGZ format.

All devices and installed applications need to register themselves with a central registry RR. The RR also provides an API for the discovery of devices, device capabilities, installed applications and services provided by the applications. The architecture of the system is shown in Fig. 1. The arrows depict communication between the components: IDE deploys and manages applications in devices, devices register themselves and update their applications to RR, and IDE queries devices and installed applications from RR. All the communication in the original system is implemented using HTTP. Further information about the original system can be found in [1, 40].

It should be noted that the logical control-flow of the original system requires a response from the previous request before sending the next. For example, IDE should not query the updated state of the applications (six in Fig. 1) from the RR before it gets the acknowledgment of a deployment (4 in Fig. 1) from the device.

4.3 MQTT implementation

We use node.js [41], MQTT.js [42], and Mosquitto [43] MQTT broker which implements MQTT version 3.1.1. We implemented the communication from the IDE to devices and from devices to RR seen in Fig. 1 with MQTT. From the communication shown in Fig. 1 the communication between IDE to RR is still implemented with HTTP. The resulting architecture and communication are presented in Fig. 2. The right side on Fig. 3 shows how an MQTT broker is accessed by the other parties behind NATs. We think that communication from IDE to devices and from devices to RR is enough for a proof of concept to show MQTT working in our use case. However, we do not foresee any problem in implementing communication from IDE to RR with MQTT as well – if that is needed. Note that RR is a central component in the IoT deployment system whereas the MQTT broker only delivers MQTT messages.

From the different options [(1) complex topic and requestID in the topic, (2) general topic and requestID in message], to implement the request-response pattern with MQTT discussed in Section 4.1, we selected the one where a unique topic is created for every request and response. The topic hierarchies designed for our system are presented in Fig. 4. One notable detail in the hierarchies is the relation of replies to corresponding requests. In our solution, each topic mapped to a resource has request and reply subtopics followed by unique identifiers (rID). Device identifications (dID) are used as unique identifiers for devices.

The left hierarchy in Fig. 4 consists of the following structure. The first level 'device' is a topic describing the problem domain – we are deploying applications to devices. Each device in the system has a separate branch which is identified with a unique dID. For example, a device with an identification XX has a topic starting with `device/XX`. The devices manage their own branches of topic hierarchies for messages that are directed to them. Essentially this implements the addressing scheme discussed in Section 4.1: the beginning of the topic (`device/<DID>`) corresponds to IP address (domain name) and the rest corresponds to the path of URL. For details, see Table 1 where a mapping between some URLs of the system and MQTT topics is given. The level 'app' of each dID branch is for applications installed on the devices. The branches following from here are for requests and replies addressed to the applications. It would be easy to extend the hierarchy by adding a level of application IDs. For example, then it would be possible to address a certain application with a topic (`device/<ID>/app/<APPID>`).

The topics enabling communication with RR are seen on the right in Fig. 4. The first level indicates that the hierarchy is meant for RR. The topics `RR/devices/request and RR/devices/reply` are used for registering devices. Device identification is not used yet because the unregistered device does not have a dID yet. On the second level, each registered device has a separate branch identified by dID. 'Apps' branch is used for managing the applications running on the device or retrieving information about them. For example, it is used by devices to publish the current states of all its applications to RR using topic (`RR/devices/<DID>/apps/request/<rID>`). As with device hierarchy, if an individual application resource needs to be expressed, the hierarchy can be extended by adding appIDs after 'apps' level.

The sequence charts in Fig. 5 describe how an HTTP request-response relates to MQTT request-response. One request-response sequence between device and RR requires multiple communications between three components, device, MQTT broker and RR. The message sequence chart in Fig. 5 depicts the devices' registration both with HTTP and MQTT. For the RR to receive registrations, it needs to subscribe to a topic for RR related requests with a wild card (+). The device, in turn, subscribes to a topic where it knows to expect the response. Then the device sends a registration request to a topic that contains a unique identifier for the request and is also subscribed by RR with the wild card. RR will then send a response to the response topic subscribed by the device. The status code in the reply tells whether the registration was successful or not. After the registration, the device can unsubscribe from the response topic.

In Fig. 6, the situation is similar to the device registration case with the difference that the communication happens from the IDE to a device. In this case, a user deploys an application using the IDE. Initially, the device (that has already been registered and thus has a dID) must have subscribed to a topic that is used for deploying applications to that device. Before sending the deployment message, the IDE subscribes to a topic dedicated to the response sent by the device. During the deployment, the IDE first creates a unique rID, subscribes to a reply topic of that rID and publishes a message to the topic that the receiving device has subscribed to. After completion of the deployment, the device publishes the response to the unique topic expected by the requesting IDE. The response contains information whether the deployment was successful or not. The IDE can unsubscribe from the reply topic finally.

5 Evaluation

5.1 Amount of refactoring

One of our targets and success indicators was the number of required changes to the original system. We noticed that the changes were relatively small and local. The adaptation to MQTT was done by replacing the parts of the source code that sends the HTTP requests with a source code that publishes an MQTT message and subscribes to the reply topic.

The code examples in Fig. 7 show how the original JavaScript code snippet using HTTP is refactored for MQTT. The operation shown in the example is a device registering itself to RR. The original code simply sends an HTTP POST to a URL hosted by RR and saves the returned dID for later use. The handling of the response in refactored code is a bit more complex because it needs to be converted to a string and the response status and message body are not automatically parsed. An alternative and possibly a simpler way could be to include the status code in the topic hierarchy but we did not try that in practice.

While the snippets are an example from our case, the solution can be generalised so that for using MQTT in request-response style, dedicated request-response topics together with request IDs offer one flexible solution. The solution also provides the other benefits of MQTT such as lower resource consumption and ability to use a normal publish-subscribe pattern when needed.

5.2 Resource consumption

We did not discover any performance issue while testing. The maximum size of an MQTT message is about 256 MB and the size of the messages in our system has been under 10 kB. The requester removes temporary topics after receiving the response which increases the amount of traffic a bit but prevents the system from growing memory usage continuously. Our solution uses complex topics though simple and short topic structures could use fewer resources [38]. On the other hand, a flexible topic structure is important when adding new features. Still, MQTT is IoT optimised and using it should help to save the resources when compared with HTTP.

We conducted experiments to compare the resource consumption of the protocols with the request-response pattern. We compared two applications. The first sent 1000 requests and received replies for those with MQTT. The experiment setting was such that Raspberry Pi (our constrained IoT device) worked as a requesting MQTT client, the MQTT broker located in the OpenStack virtual machine, and a modern laptop had the replying MQTT client. The latency from Raspberry Pi to MQTT broker was 38 ms (throughput 14 Mbps), and from the laptop to MQTT broker was 2 ms (throughput 262 Mbps). The request workload was 'GET x' where x was a request ID integer between 0 and 999. The reply

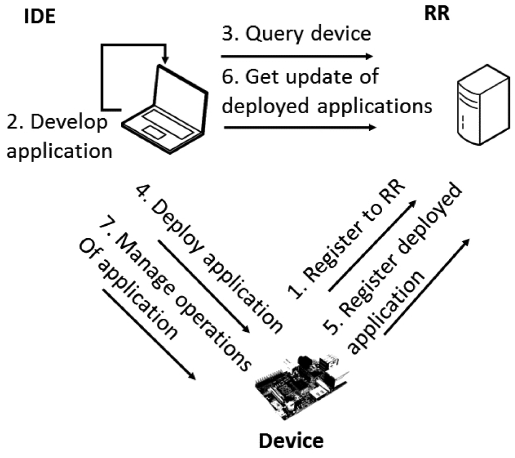
IDE
RR
3. Query device
6. Get update of deployed applications
2. Develop application
4. Deploy application
7. Manage operations Of application
1. Register to RR
5. Register deployed application
Device

Fig. 1 *Original framework with HTTP. Numbers indicate an order of a typical workflow*

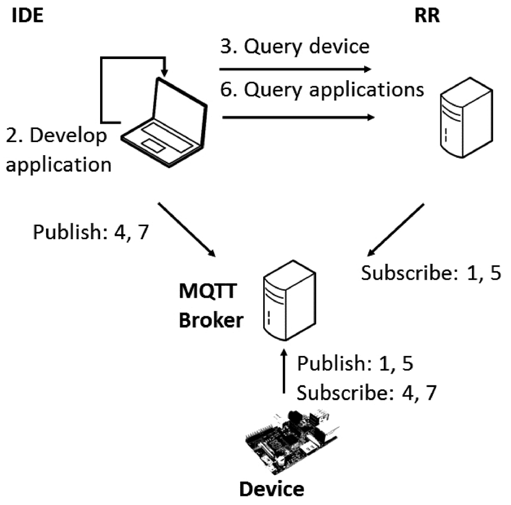

Fig. 2 *HTTP replaced with MQTT. Numbers correspond to the numbers seen in Fig. 1*

workload was ‘200 OK x’ where x was the corresponding request ID. MQTT payload was symmetrically encrypted with AES-256. The second application did the same thing with HTTP without encryption. The setting was such that Raspberry Pi (again as constrained IoT device) worked requesting HTTP client sending requests to an Apache HTTP server. The latency from the Raspberry Pi to the HTTP server was 20 ms. The reply workload was empty.

The requests were sent as fast as possible without extra delay between them. The experiments were conducted on the Rasperry Pi version 2 (total memory 927 MB) running Raspbian operating system and Node.js [41] version 5.4.0. The used MQTT library was mqtt.js version 3.1.1. The MQTT broker was Mosquitto version 1.4.10.

The measurements were done using process.memoryUsage() function of Node.js and Linux command line tool *time*. MemoryUsage function shows the total memory allocation and time shows the used CPU time of a process execution.

Table 2 shows the results of five CPU time and memory consumption measurements. MQTT used on average 10.3 s less CPU time and HTTP used more than double the CPU time in every measurement. On average MQTT used 1.6% points less memory.

6 Discussion

The proof of concept works as expected and it allows demonstrations outside of our lab. The original demo did not work if IoT devices were behind a firewall because IDE could not access the devices directly. Refactoring of the system on the source code level required a relatively small amount of work. The MQTT implementation has a centralised broker that is accessible from everywhere. Thus, only the broker consumes a public IP address, and the devices can stay behind NATs and firewalls. As a result, our demonstration worked as expected.

Since the broker needs a public IP address, our proposal does not work if it is located in a private IP space nor if the firewalls have default MQTT ports blocked from inside to outside traffic. In some cases, it might be possible to tunnel MQTT to other ports than the default port but we did not study it.

Note that we do not think that adding request/<rID> or reply/<rID> to the topic addressing a resource (for example, `example.com/resource123/request/<rID>`) is a practice used in REST. Actually, such a solution might be even argued to resemble remote procedure call more than REST. Instead, we try to describe a way to use MQTT efficiently and conveniently while being consistent with REST as far as it is feasible. While the MQTT version of the REST method (for example GET) uses to request and reply topics, the target of the operation is identified by the base topic `example.com/resource123`.

Our experiments with 1000 request-replies suggest that the MQTT uses considerably less CPU time and less memory than HTTP even with a request-response pattern. While 1000 request-replies done as fast as possible might not be a real-world use case, it gives hint about the resource usage between the protocols. Reducing CPU and memory consumption improves energy-efficiency, which is getting increasingly important. The differences seen in our experiments can be significant on constrained devices even if you do not need to worry about NAT traversal.

In HTTP-based systems, the clients get either a response or an error condition. In the current MQTT system conditions like network errors may lead to situations where the client never gets any notice and the requesting subsystem has no way to discover what went wrong. The quality of service (QoS) of MQTT could provide a partial solution but most probably, some additional logic needs to be added. The QoS level used by us is MQTT level ‘zero’ which means that there are no guarantees of delivery of the messages. However, we have not discovered any cases of undelivered messages in our tests.

If extra security is needed in REST, there are standard mechanisms for encryption, authentication, and authorisation. MQTT also provides security features such as authentication and authorisation but those were not used with the proof of concept. While they were not used, we think using the following features would be relatively straightforward:

- Authentication by adding username and password while connecting to broker [36].
- Authorisation by using Mosquitto broker's topic permissions [44].
- Payload encryption [45]. (Our MQTT experiments done with symmetric payload encryption suggest that resource usage still remains lower than HTTP.)

In addition, transport layer security could (and should) be used but it can be challenging with constrained devices [45]. Secure MQTT [46] is also one proposed solution to improve MQTT security.

7 Conclusions and future work

We presented a work where originally REST-based communication was substituted with MQTT communication maintaining the request-response behaviour from the application point of view. The use case of the work was an IoT development and deployment framework presented in [1]. The initial motivation for the work came from the problems of using HTTP to access devices that are

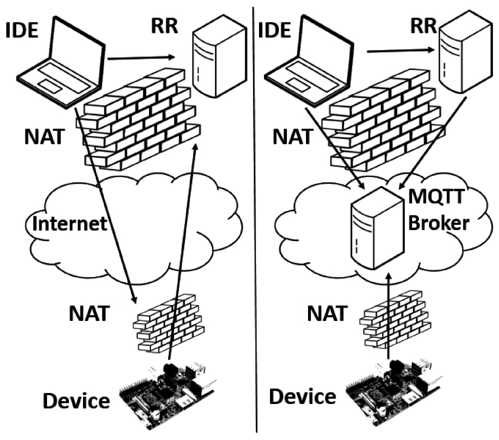

Fig. 3 *Left: NATs restricts communication with HTTP. Communicating with parties behind NATs is difficult. Right: MQTT broker in the public Internet allows communication despite the NATs. Similarly to left figure, the cloud presents the Internet*

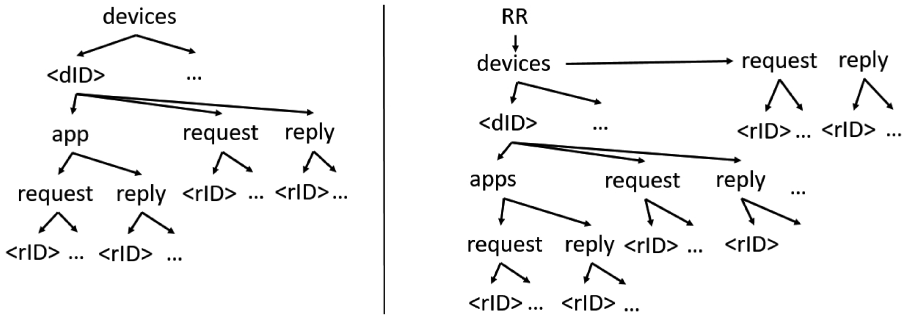

Fig. 4 *Topic hierarchies for MQTT implementation. Left: device. Right: RR*

Table 1 Mapping of the URLs to MQTT topics when a device sends requests to RR

URL	MQTT topic
RR-host-address/devices	RR/devices/request/<rID>
RR-host-address/devices/<DID>/ apps/<APPID>	RR/devices/<DID> /apps/ request/<rID>

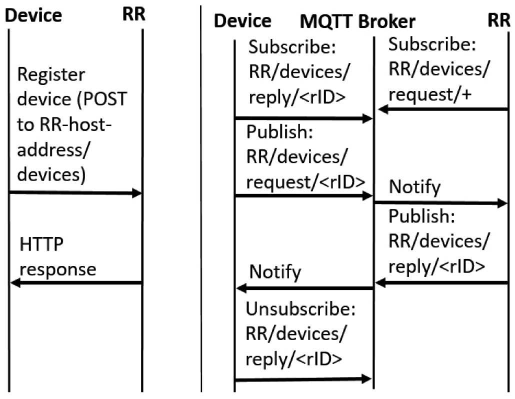
Device
RR
Register device (POST to RR-host-address/ devices)
HTTP response
Device
MQTT Broker
RR
Subscribe: RR/devices/ reply/<rID>
Subscribe: RR/devices/ request/+
Publish: RR/devices/ request/<rID>
Notify
Publish: RR/devices/ reply/<rID>
Notify
Unsubscribe: RR/devices/ reply/<rID>

Fig. 5 *Left: register a device with HTTP. Right: register a device with MQTT*

behind a firewall. Another reason is that MQTT is more suitable for IoT devices with limited resources, and MQTT is used in practical IoT implementations. The main technical challenges in

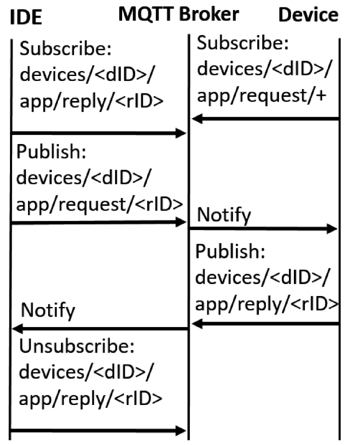
IDE
MQTT Broker
Device
Subscribe: devices/<dID>/ app/reply/<rID>
Subscribe: devices/<dID>/ app/request/+
Publish: devices/<dID>/ app/request/<rID>
Notify
Publish: devices/<dID>/ app/reply/<rID>
Notify
Unsubscribe: devices/<dID>/ app/reply/<rID>

Fig. 6 *IDE deploys an application to a device*

our research were related to the implementation of a request-response paradigm. Our solution is based on separate response messages and design of topic hierarchy with specific request and response topics. In addition, a status code needs to be added to the content of the response. Nevertheless, the similarities between the concepts of REST and MQTT helped us in the design work. Our comparison of resource usage between the protocols suggests that MQTT uses less CPU time and memory even with the request-response pattern, which is significant when using IoT devices with low resources.

```
var options =
  {uri: RRInfo.url, method: 'POST',
  json: deviceInfo};
request(options,
  function (err, res, body) {

  if(!err && res.statusCode  == 200) {

    //Read dID from HTTP response
    RRInfo.idFromRR
      body.toString();

    //Save dID to config file...
    // (removed from this snippet)
    } else {
    //Error handling..
  }
});
//-----------------------------------

//Subscribe for response and register
// the device. Use unique request ID.
client.on('connect', function ()  {
  client.subscribe('RR/devices/reply/' + rId);
  client.publish('RR/devices/request/' + rId,
    JSON.stringify(deviceInfo));
});

// Listen for responses and parse the
// status code from the body.
client.on('message',
  function (topic, message) {

  if(topic == 'RR/devices/reply/' + rId) {
    if(message.toString().substr(0, 3)
      == '200') {

        // Read dID from MQIT response.
        // It is after the status code.
        deviceInfo.idFromRR =
        message.toString().substr(5,
          message.length - 1);

        //Save dlD to config file...
        //(removed from this snippet)

        client.unsubscribe(
          'RR/devices/reply/' + rId);
    } else {
      //Error handling...
    }
  }
});
```

Fig. 7 *Top: Original HTTP version. Bottom: Original code refactored to use MQTT*

One way to extend the work would be to implement the whole system to support MQTT. Currently, only the communication that prevented us from demonstrating the system remotely has been implemented with MQTT. For example, the IDE still uses REST for communicating with the RR. The security aspects require further analysis and research since the current implementation assumes that all participating entities trust each other, which is not case in the real world. By adding authentication and encryption technologies, the security could be improved. In addition, the system should be evaluated with a larger scale set-up and amount of data. The number of messages, required processing, and energy consumption should be measured in more detail in a real-world scenario to answer the questions rising from the growing IoT phenomenon. Comparison with other approaches, like CoAP, HTTP long polling, and hole punching would also be an interesting research topic.

Table 2 Means and confidence intervals of the measurements. Confidence level is 90%

	Mean	Confidence interval
HTTP CPU, s	17.72	[17.43, 18.01]
MQTT CPU, s	7.46	[7.37, 7.55]
HTTP memory, %	6.05	[6.04, 6.06]
MQTT memory, %	4.46	[4.45, 4.47]

8 Acknowledgments

We would like to thank NOKIA for funding the work.

9 References

[1] Ahmadighohandizi, F., Systä, K.: 'Application development and deployment for IoT devices'. The Fourth Workshop on Cloud for IoT (CLIoT), Vienna, Austria, September 2016

[2] Datta, S.K., Bonnet, C., Nikaein, N.: 'An IoT gateway centric architecture to provide novel M2M services'. IEEE World Forum on Internet of Things (WF-IoT), Seoul, South Korea, March 2014, pp. 514–519

[3] Maier, D., Haase, O., Wäsch, J.*, et al.*: 'NAT hole punching revisited'. 36th Conf. on Local Computer Networks (LCN), Bonn, Germany, October 2011, pp. 147–150

[4] Lin, Y.D., Tseng, C.C., Ho, C.Y.*, et al.*: 'How NAT-compatible are VoIP applications?', *IEEE Commun. Mag.*, 2010, **48**, (12), pp. 58–65

[5] 'MQTT'. Available at http://mqtt.org/, accessed 5 October 2016

[6] Lampkin, V., Leong, W.T., Olivera, L.*, et al.*: '*Building smarter planet solutions with MQTT and IBM websphere MQ telemetry*' (IBM Redbooks, 2012)

[7] 'Power profiling: HTTPS long polling vs. MQTT with SSL, on android'. Available at http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https, accessed 5 October 2016

[8] Naito, K.: 'A survey on the internet-of-things: standards, challenges and future prospects', *J. Inf. Process.*, 2017, **25**, pp. 23–31

[9] Kim, G., Kim, J., Lee, S.: 'An SDN based fully distributed NAT traversal scheme for IoT global connectivity'. IEEE Int. Conf. on Information and Communication Technology Convergence (ICTC), Jeju, South Korea, 2015, pp. 807–809

[10] Cugola, G., Migliavacca, M., Monguzzi, A.: 'On adding replies to publish-subscribe'. Proc. 2007 Inaugural Int. Conf. on Distributed Event-based Systems, Toronto, Ontario, Canada, June 2007, pp. 128–138

[11] Hill, J.C., Knight, J.C., Crickenberger, A.M.*, et al.*: 'Publish and subscribe with reply'. (No. UVA-TR-CS-2002–32), Department of Computer Science, Virginia University, Charlottesville, 2004

[12] Rodríguez-Domínguez, C., Benghazi, K., Noguera, M.*, et al.*: 'A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems', *Sensors*, 2012, **12**, (6), pp. 7648–7668

[13] Technical Committee, O.A.S.I.S. M.Q.T.T. 'Request/reply message exchange patterns and MQTT version 1.0 working draft 02'. Available at https://www.oasis-open.org/committees/download.php/56280/reqreply-v1.0-wd02.docx, accessed 5 October 2016

[14] Collina, M., Corazza, G.E., Vanelli-Coralli, A.: 'Introducing the QEST broker: scaling the IoT by bridging MQTT and REST'. IEEE 23rd Int. Symp. on Personal indoor and mobile radio communications (PIMRC), Sydney, NSW, Australia, September 2012, pp. 36–41

[15] Chen, H.W., Lin, F.J.: 'Converging MQTT resources in ETSI standards based M2M platform'. 2014 IEEE Int. Conf. on the Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom), and IEEE Cyber, Physical and Social Computing (CPSCom), Taipei Taiwan, September 2014, pp. 292–295

[16] Bellavista, P., Zanni, A.: 'Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP'. IEEE 2nd Int. Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), Bologna Italy, September 2016, pp. 1–6

[17] Uehara, M.: 'A case study on developing cloud of things devices'. IEEE Ninth Int. Conf. on Complex, Intelligent, and Software Intensive Systems (CISIS), Blumenau Brazil, July 2015, pp. 44–49

[18] 'Kura remote device management'. Available at https://eclipse.github.io/kura/ref/mqtt-namespace.html#mqtt-request/response-conversations, accessed 5 October 2016

[19] 'Request/Reply'. Available at http://dev.solacesystems.com/get-started/mqtt-tutorials/request-reply_mqtt/, accessed 5 October 2016

[20] 'Request/Response Pattern Over MQTT'. Available at http://www.bitreactive.com/mqtt-request-response/, accessed 10 May 2016

[21] 'Stock Explorer: Using Pub/Sub for Request/Response'. Available at https://www.codeproject.com/Articles/1159256/Stock-Explorer-Using-Pub-Sub-for-Request-Response, accessed 3 February 2017

[22] Fremantle, P.: 'A reference architecture for the internet of things'. WSO2 White Paper, 2014

[23] Fette, I.: 'The websocket protocol', RFC 6455, 2011

[24] Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F.*, et al.*: 'A survey on application layer protocols for the internet of things', *Trans. IoT Cloud Comput.*, 2015, **3**, (1), pp. 11–17

[25] Wu, Z., Luo, H., Zhang, S.*, et al.*: 'Design of a distributed network address translation system architecture'. Proc. Int. Conf. on Advanced Intelligence and Awareness Internet (AIAI 2011), Shenzhen, China, October 2011, pp. 207–210

[26] Kubota, A., Miyake, Y.: 'Public Key-based rendezvous infrastructure for secure and flexible private networking'. IEEE Int. Conf. on Communications (ICC'09), Dresden, Germany, 2009, pp. 1–6

[27] Eppinger, J.L.: 'TCP connections for P2P apps: a software approach to solving the NAT problem', Institute for Software Research, 2005, pp. 1–8

[28] Duquennoy, S., Grimaud, G., Vandewalle, J.J.: 'The Web of things: interconnecting devices with high usability and performance'. IEEE Int. Conf. on Embedded Software and Systems (ICESS'09), Zhejiang, China, 2009, pp. 323–330

[29] Müller, A., Carle, G., Klenk, A.: 'Behavior and classification of NAT devices and implications for NAT traversal', *IEEE Netw.*, 2008, **22**, (5), pp. 14–19

[30] Srirama, S.N., Liyanage, M.: 'TCP hole punching approach to address devices in mobile networks'. Int. Conf. on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 2014, pp. 90–97

[31] Ford, B., Srisuresh, P., Kegel, D.: 'Peer-to-peer communication across network address translators'. USENIX Annual Technical Conf., General Track, Anaheim, CA, USA, 2005, pp. 179–192

[32] Pyylampi, T.: 'Design and implementation of a real-time multiplayer system for android operating systems'. Master Thesis (in Finnish), Tampere University of Technology, 2015

[33] 'OMA LightweightM2M V1.0 Overview'. Available at http://www.openmobilealliance.org/wp/overviews/lightweightm2m_overview.html, accessed 29 September 2017

[34] Rao, S., Chendanda, D., Deshpande, C.*, et al.*: 'Implementing LWM2M in constrained IoT devices'. IEEE Conf. on Wireless Sensors (ICWiSe), Melaka Malaysia August 2015, pp. 52–57

[35] 'CoAP – RFC 7252 constrained application protocol'. Available at http://coap.technology/, accessed 29 September 2017

[36] 'MQTT and CoAP, IoT protocols'. Available at https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php, accessed 3 February 2017

[37] OASIS MQTT Technical Committee.: 'MQTT request/reply MQTT-197'. Available at http://www.oasis-open.org/committees/download.php/58854/MQTT.Request-Reply.Overview-Sept.2016.pptx, accessed 16 August 2017

[38] 'MQTT essentials part 5: MQTT topics & best practices'. Available at http://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices, accessed 5 October 2016

[39] Ahmadighohandizi, F.: 'Liquid-IoT runtime environment (LIoTRE) API specification'. Available at http://farshadahmadi.github.io/, accessed 8 February 2018

[40] Hylli, O., Ruokonen, A., Mäkitalo, N.*, et al.*: 'Orchestrating the internet of things dynamically'. Proc. 1st Int. Workshop on Mashups of Things and APIs (Middleware'16, 17th Int. Middleware Conf.), Italy Trento, 2016

[41] 'Node.js'. Available at https://nodejs.org/en, accessed 8 November 2016

[42] 'MQTT.js'. Available at https://www.npmjs.com/package/mqtt, accessed 5 October 2016

[43] 'Mosquitto – an open source MQTT v3.1/v3.1.1 broker'. Available at https://mosquitto.org/, accessed 5 October 2016

[44] 'Configuring and testing Mosquitto MQTT topic restrictions'. Available at http://www.steves-internet-guide.com/topic-restriction-mosquitto-configuration/, accessed 10 February 2018

[45] 'MQTT security fundamentals: MQTT payload encryption'. Available at https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption, accessed 10 February 2018

[46] Singh, M., Rajan, M.A., Shivraj, V.L.*, et al.*: 'Secure MQTT for internet of things (IoT)'. Fifth Int. Conf. on Communication Systems and Network Technologies (CSNT), Gwalior, India, April 2015, pp. 746–751

PUBLICATION
V

Log Analysis of 360-degree Video Users via MQTT

A. Luoto

DOI: 10.1145/3318236.3318248

Log Analysis of 360-degree Video Users via MQTT

Antti Luoto
Laboratory of Pervasive Computing
Tampere University of Technology
Tampere, Finland
antti.l.luoto@tut.fi

***Abstract*—Analyzing 360-degree video users is beneficial for 360-degree video application development. The analysis can be done with logged user data. In this paper, we argue that MQTT is a conventional technology for distributed logging of mobile 360-degree video users. MQTT not only saves resources also allows communication from the logging server to mobile clients in various networking conditions relatively easy. We constructed a proof of concept to show the feasibility of the approach. As log analysis examples, the proof of concept visualizes results of the most popular region of interest analysis and k-means clustering. The used research method is design science.**

Keywords-component; 360-degree video, MQTT, Log Analysis

I. Introduction

360-degree videos are getting more and more popular. They can be watched with head-mounted displays (HMD) but they appear in other contexts, such as websites, nowadays. We primarily discuss 360-degree videos in hand-held mobile device applications though the techniques are applicable in HMD or web usage as well.

In our earlier work, we made a user logging and visualization framework for 360-degree videos [21]. The framework works on Android and uses the video player offered by Google VR SDK. We presented a technique to place simple graphics over the video and discussed HTTP based user logging. We made the work with lightweightness and resource usage in mind. In this paper, we propose improvements to the framework.

Low resource usage can be seen as one of the essential aspects of 360-degree video development [20] and HTTP is not an optimal solution for logging device orientation. 360-degree video user logging applications need to send small log entries often, for example 23 times a second, which causes overhead with HTTP.

We believe that the resource usage of the framework could be improved by using a lightweight communication protocol MQTT. Besides being lightweight, publish-subscribe pattern of MQTT brings network architectural benefits in multi-user environment and it supports applications located in various networking conditions suffering from bad connections or NATs.

One application for user log data is data analysis. In our earlier work, we did not discuss data analysis of the logged data. Since that, we have reported visual heat map analysis of the logged data in a web user interface [10]. This time, we present data analysis for the logged data that also benefits from MQTT. We implemented SQL-based solutions for calculating the most popular region of interest in the video and k-means clustering algorithm. MQTT helps sending the results for visualization on the mobile devices without a need for requesting them.

Thus, the challenges we aim to solve with MQTT are:

- Supporting near real-time log analysis.
- Saving resources on a battery powered mobile device.

The used research method is design science. It is a research method used in software engineering [31] that includes six steps: *problem identification and motivation*, *definition of the objectives for a solution*, *design and development*, *demonstration, evaluation and communication* [25]. We include all the steps in this paper.

II. Background

This section covers the step *problem identification and motivation* of design science method.

A. 360-degree Videos and User Logging

360-degree videos are omnidirectional spherical videos where the user can choose the direction of the view port. 360-degree videos have applications in security, robotics, education etc.

User logging in 360-degree video domain is an important feature for multiple parties and applications [5, 18, 19, 33]. For example, logs can be used to analyze not only the videos but also the video user interface such as graphical elements or interaction points on top of the video. While we are not, strictly speaking, in virtual reality domain, the discussion of Ritchie et al. [28] about the benefits of user logging in virtual reality applies to our work: it is an almost non-intrusive method of capturing a rich data source for analysis, it minimizes user interactions during the data capture, it has potential to reduce time overhead of the capturing process, and the captured data can also be reused.

Popular data for logging 360-degree video users includes the direction of the view (for example, in Euler angles) and timestamp. This is valuable information since many applications are immersive and only show the chosen view port that adapts to the device orientation. One challenge of 360-degree video user logging comes from the frequently updating view. A video with 23 frames per second requires a log packet to be sent every 43th millisecond if every frame triggers a logging event.

B. Log Analysis

360-degree video user data can be analyzed visually, for example, with heat maps. Applications for heat maps include analysis to find the areas of interest in the video [5]. Heat map shows an aggregation of users viewing a part of the video by coloring the area. The brighter the color, the more often the part was included in the users' field of view. Continuously updating frame video adds challenge to the heat map generation when compared to static images. Heat map generation is also a resource-consuming task [7].

Another visual analysis method is scanpath. Scanpath is a linearly connected dot sequence following the eye movement coordinates in time but it is not good for aggregate data since markers and labels are easily confusing when analyzing multiple users [7]. Löwe et al. [19], for example, generate heat maps and scanpaths.

Visual analysis, however, requires manual work. Data mining algorithms could be an alternative. With data mining methods, we can programmatically find the regions of interest from the video. Visualizing such data mining results can be also less resource consuming than generating a heat map. Of course, producing a very simplified heat map could be an option but we were interested to explore other methods as well. K-means clustering, for example, is a popular clustering algorithm used in data mining that is generally easy to implement, simple, efficient and empirically successful [13].

C. Why MQTT

Capturing the device orientation for a single moment does not require a lot of data but the difficulty comes from the frequently updating video frame. Thus, we need to transfer a big number of tiny packets. However, protocol overhead of HTTP causes serious problems when transferring a big number of tiny packets [34].

MQTT is a lightweight messaging protocol that follows the publish–subscribe paradigm, which makes it suitable for resource constrained devices and non-ideal network connectivity conditions. Because of its simplicity, and a very small message header compared with other messaging protocols, it is often recommended for IoT. [6]

MQTT is often proposed for communication from sensors to (edge) gateways in IoT domain but we believe it has other uses as well. We consider mobile phone sensors similar to IoT sensors. In addition, MQTT allows the mobile phone to receive data by subscribing to MQTT topics. Thus, it is easy to send data to multiple subscribers, for example, when multiple users watch the same video. An example of a mobile use case of MQTT is Facebook chat [36] that is often used with mobile devices in non-ideal network conditions.

IoT development survey 2016 [29] shows that MQTT and HTTP are the most popular IoT protocols which supports choosing MQTT as an implementation technique. Dizdarevic et al. [6] comment on the survey that "The reason for this is that MQTT and HTTP REST are currently comparably more mature and more stable IoT standards than other protocols."

Still, there are other IoT protocols, such as CoAP, available. It can be argued that MQTT is more suitable for IoT than CoAP because of the publish–subscribe architecture without responses [14]. Moreover, CoAP has more likely packet loss [14]. Furthermore, communication with the users can be restricted by NATs. Several authors mention the benefits of MQTT when communicating beyond NAT [3, 8, 22, 30]. Other suggested techniques for NAT traversal include Websockets, destination network translation (DNAT), virtual private networks (VPN), Universal Plug and Play (UPnP), and TCP hole punching. We consider them to be either more complex, less reliable, difficult to configure or not ideal for saving resources.

III. RELATED WORK

Related work shows that MQTT logging has been used in various domains such as home and industrial automation. Those domains are quite different from mobile 360-degree videos but they show that MQTT has been used for logging in the scientific literature.

The presented logging research from the 360-video perspective includes a few publications about publishing HMD tracking data sets and analyzing them. In contrast to their work, we concentrate more on hand-held usage with aim in lightweightness, supporting multi-user logging and near real-time data analysis.

Resource consumption comparisons between MQTT and HTTP are related work in a sense that we aim to save resources using MQTT. We did not made a resource consumption comparison since the existing studies support the assumption that MQTT saves resources also in our case.

A. Logging with MQTT

Avizheh et al. [2] propose a secure event logging system for smart homes using MQTT. They emphasize security by using Bitcoin blockchain for ensuring the integrity of log files. While we did not concentrate on security yet, authentication, authorization and symmetric payload encryption could be the first steps [12, 22].

In another home automation study, Agerwal et al. [1], use MQTT to send home automation data from sensors to Raspberry pi 3 which works as a central unit with data-logging. They use Node-RED and Mosquitto MQTT broker to create a cost-effective open source solution. We also use open source components such as Google VR SDK and Mosquitto.

Wenger et al. [32] discuss lightweight logging in industrial automation environment. In their solution, control devices connect to a cloud MQTT broker publishing logging data so that monitoring service subscribes to logging topics. We are not explicitly discussing cloud environment but, for example, a web based monitoring service can be implemented with our database [10].

Zabasta et al. [35] have an MQTT enabled event handling and historian systems (used for data logging) for utility networks. They note that the responsibilities for those components should be investigated more carefully. As in their work, we use relational database to store the sensor data and use JSON when sending data with MQTT.

Herle et al. [11] propose a REST bridge for using MQTT via standard HTTP methods. They use the bridge as a message logger that logs all the received messages to a MongoDB database. With their solution, users are able to get a whole history of events instead of just the latest event. In contrast, our solution does not provide a way get the whole history of events via MQTT. This is a clear disadvantage when compared to our earlier REST solution.

Kodali and Gorantla [15] made a Python application to receive weather-tracking data (JSON) via MQTT broker and store it to SQLite database. The broker, the application and the database are located on the same Raspberry Pi. Kodali and Sarjerao [16] also hint about a similar system where air quality data is continuously stored in a MQTT server. Likewise, we use JSON and relational database, but we have the logger application and the database in one server and MQTT broker on other. The broker is the only component available in the open Internet in our solution.

Harris and Curry [9] note that MQTT does not provide inherent logging capabilities. Therefore, they made a tracker for logging MQTT traffic by LoRaWAN devices. Because of the inherent logging capabilities, we also had to implement logging components.

B. Analyzing 360-degree Video Users

Lo et al. [18] offer public HMD user logging data sets. Their architecture is primarily for only local logging and they create saliency maps and motion maps with the logs. In contrast, our logging server architecture enables a way to collect data sets simultaneously with multiple users and analyze the logs in real-time.

Corbillon et al. [5] also released a public 360-degree video head movement data set. They used the logs to create example statistics for analyzing users' navigation patterns.

Wu et al. [33] also published a data set for exploring user behavior in spherical videos. They present preliminary analysis of their data set by presenting visualizations and statistics. However, they do not discuss producing statistics in real-time. Examples of their statistics include head movement angular speeds, where as we calculate the most popular region of interest and k-means clustering.

Qian et al. [27] computed weighted linear regression on head orientation data with results that the head orientation can be predicted with over 90% accuracy. They predict single user traces, where as we concentrate on multi-user analysis. However, they propose (not implemented) using multi-user statistics to help with the inaccuracy of single user prediction.

C. Resource Consumption Between MQTT and HTTP

Luoto and Systä [22] made a memory and CPU usage comparison between MQTT and HTTP. MQTT used less resources even with the suggested request-response pattern and symmetrically encrypted message payload. Comparison with 1000 request-responses is applicable to 360-degree video domain because 23 seconds of 23 FPS video produces about 1000 log entries.

Chen and Lin [4] compared MQTT and HTTP proxies with the result that the MQTT proxy had a shorter latency and saved more power than the HTTP proxy. While proxy comparison is not directly related to our work, their results suggests that MQTT could use less power in our solution as well.

A comparison of power consumption between MQTT and HTTP [23] on Android shows that MQTT has lower power consumption for example on maintaining open connection and sending and receiving messages (when compared to HTTPS long polling). We also use Android and aim to reduce power consumption by choosing MQTT.

Yokotani et al. [34] made comparisons in different scenarios with 10, 100 and 1000 devices with MQTT and HTTP. They concluded their study so that MQTT performs better than HTTP. While we could not evaluate our work even with ten devices, such results encouraged us even more to choose MQTT, and evaluation with more devices could be future work.

IV. Solution

To define the *objectives for the solution* for design science method, our objectives are: saving resources by choosing other network protocol than HTTP, and supporting near real-time data analysis by enabling an easy access to analysis results in a realistic Internet architecture with NATs. The section also covers the steps *design and development,* and *demonstration* (as far as it is possible with text and images).

A. Original HTTP Logging

The original logging and visualization framework runs on Android mobile phone [21]. It uses the class VrVideoView of Google VR SDK to create a video player in Android application. OnNewFrame function of the class is executed when the video frame updates. This also triggers our logging events. Therefore, we get log only from drawn frames and sometimes the player skips frames. The most important logged information contains yaw, pitch, accelerometer values (x, y, z), and video time. Yaw and pitch present rotation in a spherical space. Yaw is the vertical angle between -180 and 180 degrees, and pitch is the horizontal angle between -90 and 90 degrees.

The log entry was sent to RESTful logging server that used PostgreSQL as database. The used data format was JSON. The logged data was used, for example, to create heat maps in a web application [10]. In addition, we could visualize traces of the previous view sessions in the Android application. Creating a HTTP request for every frame in 23 FPS video already makes a lot of traffic, and the FPS can be even higher. The log entries could be sent as batches but it does not support the near real-time aim of the study.

B. MQTT Logging

To test the feasibility of MQTT on our objectives, we implemented a proof of concept. The architecture of the proof of concept can be divided to three main components: mobile phones with video players, MQTT broker delivering messages, and back end with Logger and database. The used MQTT broker is Mosquitto. Figure 1 shows the components on a high abstraction level.

1) Logger

We implemented the logger application with Node.js and MQTT.js library. The logger subscribes to topics for registering a viewing session and sending logging data. After receiving the data, the logger stores it to the database. It also publishes data analysis results via MQTT.

2) Mobile Client

We used the class MqttAndroidClient of the Eclipse Paho project to implement MQTT on Android. The mobile application presented in our earlier work was not changed other than by refactoring the HTTP related code to use MQTT, adding subscribing to the data analysis topic, and drawing the analysis results on video (using the visualization technique presented in our earlier work). Log message payloads are still in JSON format.

3) MQTT Topic Design

MQTT topics are strings that the broker uses for delivering messages for interested clients. Clients can subscribe to a topic to receive messages published to that topic. MQTT topics have levels that are separated by slash characters.

The topics needed to be designed with one-way communication in mind. For example, in request-response pattern, it is possible to ask for a unique client id from the back end, and then use that for identifying view sessions. Since we cannot do request-response with publish-subscribe protocol, we had to generate the needed ids in the video player client. For example, when a user starts to watch a video, the video application generates an id for the view session using a hash function. Then, when the application sends the first log entry, it also registers a view session using an MQTT topic that contains the generated hash. For example, publishing for the first time to topic *video/(video URL hash)/viewsession/(view session hash)* registers a new view session and adds the log data (from payload) for the video identified by *view session hash*. Later publishing to the same topic, only adds the log data to database since the view session has been already generated.

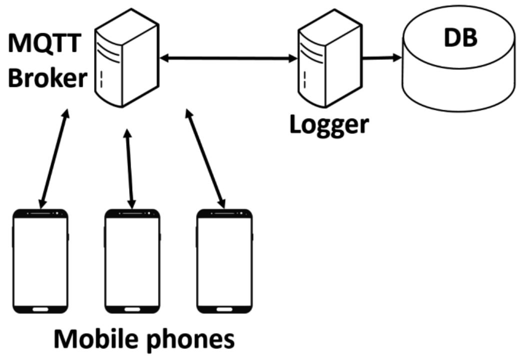

Figure 1. Components of the proof of concept. Bidirectional arrows illustrate that both the logger and the phones publish and subscribe.

Figure 2 presents how MQTT communication works between the mobile clients and the logger. For comparison, Figure 3 shows an original HTTP version of the same sequence.

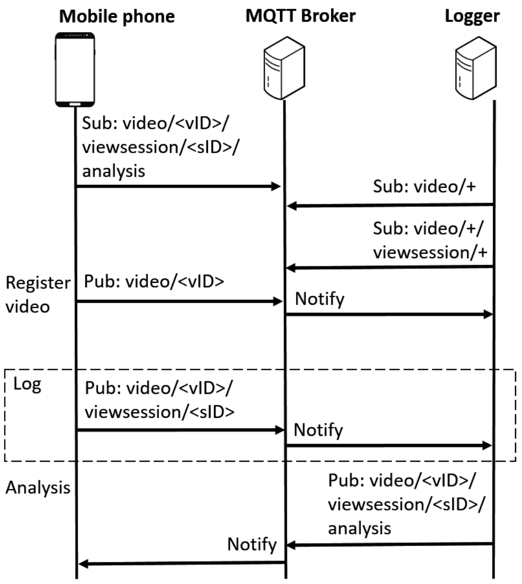
Mobile phone
MQTT Broker
Logger
Sub: video/<vID>/viewsession/<sID>/analysis
Sub: video/+
Sub: video/+/viewsession/+
Register video
Pub: video/<vID>
Notify
Log
Pub: video/<vID>/viewsession/<sID>
Notify
Analysis
Pub: video/<vID>/viewsession/<sID>/analysis
Notify

Figure 2. MQTT communication. The dashed box illustrates an operation that happens the most often. Subscriptions and video registration happen once in a view session. Frequency of analysis can be adjusted. Plus character is MQTT wildcard that matches a single topic level. vID refers to video URL hash and sID to view session hash.

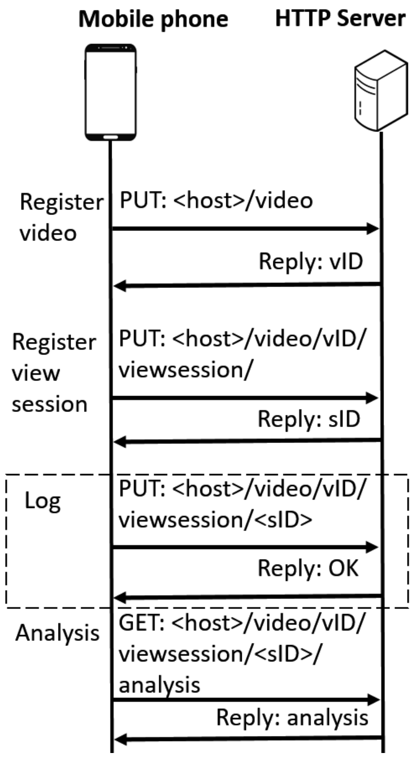

Figure 3. HTTP communication for comparison with MQTT version. The same steps can be found from both the sequences (except that registering a view session is included in 'log' step in MQTT version). The last request is hypothetical since it was not implemented with HTTP.

The MQTT topic structure is based on three base topics:

- *video/(video_URL_hash)*—For registering a new video to the system. The recipient can parse the hash from the topic string of the incoming message while the message payload contains only the original URL.
- *video/(video_URL_hash)/viewsession/(view_session_hash)*—For registering a new view session, and publishing log entries related to certain video. Again, the hashes can be parsed from the incoming topic string. The payload contains the logged data (orientation, video time etc.).
- *video/(video_URL_hash)/viewsession/(view_session_hash)/analysis*—For the back end to send analysis results about a certain view session to a device. *Analysis* could be replaced with a descriptive name of the analysis such as *popularRegionOfInterest*.

4) Database

The log database consists of three tables. Table Video contains URL of the video and a shortened hash of the URL used to form MQTT topics. We used shortened hashes (10 characters) because long topic names cause overhead in MQTT [34]. Table Viewsession contains an id for the current view session. These ids are used to identify certain view sessions in MQTT topics and to send back analysis information to devices. The last table is for the actual log entries. The database is implemented with PostgreSQL.

C. Log Analysis

1) Algorithms

We believe that the most convenient place to perform data mining would be the database in our architecture. Thus, we would not need to first retrieve the data, and then calculate it with the Logger or even with mobile phone. Retrieving lots of data could be heavy especially with multiple simultaneous users.

Our first algorithm calculated the most popular region of interest. The algorithm divides the 360-degree sphere to a graticule of 30 times 30-degree spherical rectangles. This results in 72 spherical rectangles if we count the topmost and bottommost spherical triangles as rectangles for the sake of simplicity. From now on, we call these rectangles as tiles. A sphere can be tiled in different 30 degrees was chosen because the minimum and maximum for both the yaw (from -180 to 180 degrees) and pitch (from -90 to 90 degrees) are divisible by 30 and it felt conventional enough, but it could be any other value as well. Figure 4 visualizes the division. After that, every coordinate within a second is positioned in one of the tiles. For example, a coordinate in 10 degrees of yaw 10 degrees of pitch is positioned in a tile defined by area of 0--30 degrees of yaw and 0--30 degrees of pitch. Then the algorithm counts the tiles with most positioned coordinates and returns an ordered list. With an ordered list, it is easy to get, for example, the three most popular tiles and visualize them in the application.

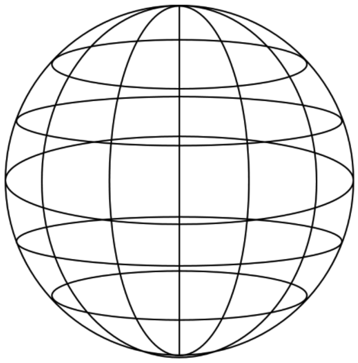

Figure 4. A sphere divided to graticule of 72 tiles each with 30 times 30 degrees area.

PostgreSQL evaluates comparisons in SELECT statement as true or false and returns accordingly either 't' or 'f' character. We use a combination of those characters to identify a certain tile on the sphere surface. Each tile is identified by a combination of 16 truth-values. The first 11 values identify the yaw position and five latter values identify the pitch position of the tile. The resulting combination has from zero to two 't' characters and the rest are 'f'. Therefore, for example, 'fftfffffffffftff' refers to a tile with left bottom corner located in -120 degrees in yaw and -30 degrees in pitch and tile identified by 16 'f' characters has left bottom corner located in 150 degrees in yaw and 60 degrees in pitch. The combination resembles a binary number and could be converted to an integer easily. However, on the application side it is straightforward to iterate through the result table and multiply 30 degrees with index of 't' to find the correct position in degrees. Figure 5 shows measured execution times on a modern laptop (ThinkPad W541, Windows 10, PostgreSQL 10.5) where execution times seem to be near linear in relation to amount of rows with our logged data. The following listing shows the SQL source code of the algorithm:

```
SELECT *, COUNT(tile) FROM
 (SELECT yaw < -150,
   yaw BETWEEN -150 AND -120,
   yaw BETWEEN -120 AND -90,
   -- ... (going through all the 30-degree yaw sections)
   yaw BETWEEN 120 AND 150,
   pitch < -60,
   pitch BETWEEN -60 AND -30,
   -- ... (going through all the 30-degree pitch sections)
   pitch BETWEEN 30 AND 60
   FROM record
   WHERE time BETWEEN 0 AND 1000) AS tile
GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
 15, 16
ORDER BY count DESC
```

To elaborate the query, the derived table named 'tile' contains truth-values. The query that produces that derived table uses table 'record' that contains rows with yaw, pitch and time. In the example, we choose time between 0 and 1000 ms but it could other timeframe as well. BETWEEN operator is inclusive meaning that begin and end values are included but we do not consider that a disruptive problem with the proof of concept. The main query counts the amount of different rows in the derived table 'tile'. The numbers (1--16) in GROUP BY refer to the unnamed columns of the derived table. Then the result is ordered in descending order according to the amount of different truth-value combinations. One row of the main query result contains 16 truth-values and how many times such combination was included in the derived table 'tile'. Figure 6 shows an example of the visualized result in the application.

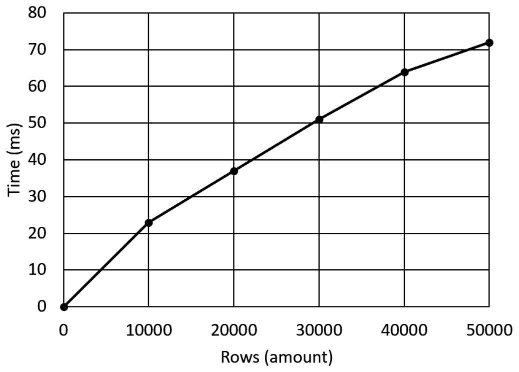

Figure 5. Execution times of the SQL query calculating the most popular tile.

The second algorithm we implemented was k-means clustering algorithm. With the algorithm, we strive to find clusters from the orientation data inside the timeframe of one second. While implementing data mining algorithms with SQL is not always considered efficient or feasible, k-means clustering can be implemented in SQL efficiently and flexibly [24]. We used a recursive PostgreSQL solution presented in Periscope Data blog [26] with minor modifications (which is not standard SQL). An example of the flexibility of the SQL implementation was that limiting the chosen timeframe (one second) is easy using SQL BETWEEN operator. Experimenting the query with our test data, using two clusters and ten iterations, takes 850 ms with 10000 log records, 2200 ms with 30000 log records and 3700 ms with 50000 log records. By halving the iterations to five, the query execution times halve but the accuracy declines. Finally, we send the cluster information to the devices for visualization.

2) Receiving Analysis Results

With HTTP, the common way to get the analysis results would be to request them from the back end. However, the back end can be protected by NAT not allowing incoming requests. The parties involved in MQTT messaging can use MQTT for NAT traversal that does not require special work other than setting up the MQTT broker and making it accessible. This works also the other way around if a mobile client is behind NAT and the back end needs to push data to it. Such functionality makes the architecture flexible.

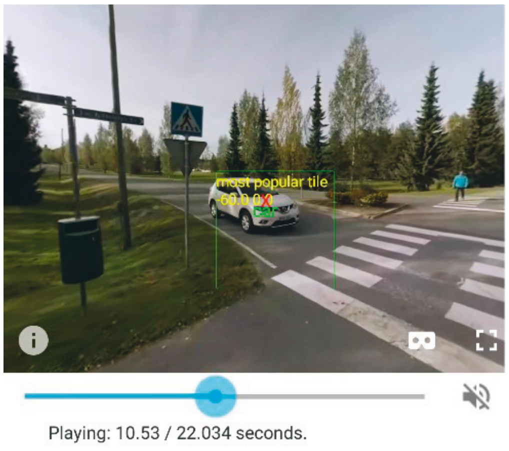

Figure 6. Example of most popular tile visualization. -60.0 tells the yaw angle and 0.0 pitch angle of the tile. Label 'car' comes from the metadata visualization presented in our earlier work. Red 'X' marks the center of the view port.

3) *Evaluation*

In this section, we evaluate the solution and the design decisions as required by the step *evaluation and communication* of design science. The criticism is directed at significance of the resource savings, claimed real-time functionality, heaviness of k-means clustering in SQL, id clashing, what we lost with HTTP, our MQTT topic design, and generally the usefulness of k-means clustering in our use case.

Is it significant to save resources by optimizing logging if a 360-degree video application already consumes much computational resources? If the logging is frequent (for example, multiple times a second), then logging users in long videos can take a considerable amount of resources. Let us assume a scenario where application makes a log entry ten times a second. In two minutes, we will end up with 1200 log entries. 1000 messages with either HTTP or MQTT already makes a clear difference in resource usage [22]. In addition, the presented ideas are applicable to software without 360-degree video players as well, such as, mobile applications logging device orientation or other often-updating sensors. Resource savings with other applications could be even more significant, if the basic functionality is not as heavy as, for example, playing 360-degree video.

We aim to support near real-time analysis. In our case real-time means that user gets the analysis results visualized on the video so that the lag is not distracting. Measuring a tolerable lag is out of the scope of this paper. However, in our database the maximum time used for the most popular region of interest analysis was about 70 ms with 50000 log records added to the time of transferring the data. We did not consider this lag distracting. One reason is also that the most popular location changes so slowly that updating it around once a second provided good enough user experience. Even less sparse updating pace could be enough that would reduce the workload of the back end. Bigger tiles also shorten SQL query time, for example diving the sphere to 60 times 30 degrees tiles reduces the query time around 15 ms with 50000 log records. K-means clustering starts to be too slow with our test data but it could be optimized further.

While MQTT is considered relatively mature technology [6], combining Android (Java) and MQTT was not especially straightforward and could be a dead-end for an inexpericed developer. In contrast, our experience suggests that starting MQTT development with Node.js and MQTT.js is easier.

Since we generate ids used for logging on Android devices, and try to keep them short to prevent MQTT topic overhead, id clashing is possible. The possibility increases when the amount of users grows. We did not consider this an important issue with the proof of concept but it needs to be solved for real usage scenarios. One solution could be to use centralized id generation technique and use MQTT request-response to receive unique ids for clients [22].

Long topic names cause overhead with MQTT [34]. Our topics could be shorter but we wanted to keep the structure readable and hierarchical. It also allows the flexible usage of MQTT wildcards, for example, if a party is interested in all the messages related to a certain video. The topic for analysis results is less frequently used than the others, so lengthy topic overhead is not as harmful with that particular topic.

With the original HTTP solution, it was easy to create a Node.js server for browsing the logging results via WWW and REST. Web browsers do not directly support MQTT so requesting and browsing the logging history for a certain view session is not as straightforward. Generally using MQTT makes it more difficult to implement a browser based user interface.

Our division of spherical space in spherical rectangles results so that the topmost and bottommost rectangles are actually spherical triangles. This can add a bit of complexity for visualizing the tiles. On the other hand, we assume that most of the videos have most of the interesting content relatively near the equator of the sphere and thus concentrating on those areas has higher priority.

Using PostgreSQL for k-means clustering was relatively simple. However, if k-means clustering is too heavy it can be lightened, for example, by reducing the amount of iterations, shortening the timeframe, or limiting the amount of data (SQL LIMIT operator). Ordonez [24] discusses multiple optimization ideas. It is also good to note that PostgreSQL allows running Python (PL/Python) and R (PL/R) for more advanced data mining but using those might bring overhead when compared to raw SQL.

One challenge of k-means clustering is that the correct amount of clusters is not always clear beforehand. For experimentation, we used two clusters with the proof of concept but one way to determine a proper suggestion for a good amount of clusters could be elbow method [17].

It is good to note that our analysis does not tell the actual gaze point because our devices cannot track users' gaze. However, it is a problem with all the techniques and devices that do not track the gaze, and on the other hand, the ideas presented in this paper could be applicable for gaze tracking as well.

V. Conslusions

In this paper, we showed how MQTT can be used for logging 360-degree video users' device orientation. The work is based on earlier work about logging users with HTTP. By choosing MQTT we aim to save computational resources, and furthermore battery usage of mobile devices. Using MQTT also enables relatively easy NAT traversal that makes publishing data in realistic network environment easier. This way we can send log analysis results to the clients without a need for specific request-response for continuous updating. As a log analysis example, we discuss SQL algorithms for calculating the most popular region of interest and k-means clustering. Such data analysis could be used, for example, to show announcements or advertisements on popular regions of interest on the video.

Acknowledgment

The authors would like to thank Business Finland for funding the project 360 Video Intelligence.

References

[1] A. Agerwal, R. Singh, A. Gehlot, G. Gupta, and M. Choudhar, "IOT Enabled Home Automation through Nodered and MQTT," International Journal of Control Theory and Applications, vol. 10(18), 2017, pp. 255–260.

[2] S. Avizheh, T.T. Doan, X. Liu, and R. Safavi-Naini, "A Secure Event Logging System for Smart Homes," Proc. Workshop on Internet of Things Security and Privacy, ACM, Nov., 2017, pp. 37–42.

[3] P. Bellavista and Z. Alessandro, "Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP", 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), IEEE, 2016, pp. 1–6.

[4] H. W. Chen and F. J. Lin, "Converging MQTT resources in ETSI standards based M2M platform", IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), 2014, IEEE, pp. 292–295.

[5] X. Corbillon, F. De Simone, and G. Simon, "360-Degree Video Head Movement Dataset", In Proceedings of the 8th ACM on Multimedia Systems Conference. ACM, 2017, pp. 199–204.

[6] J. Dizdarevic, F. Carpio, A. Jukan, and X. Masip-Bruin, "Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration", 2018, arXiv preprint arXiv:1804.01747.

[7] A. T. Duchowski, M. M. Price, M. Meyer, and P. Orero, "Aggregate gaze visualization with real-time heatmaps", In Proceedings of the Symposium on Eye Tracking Research and Applications, ACM, 2012, pp. 13–20.

[8] P. Fremantle, "A reference architecture for the internet of things", WSO2, White paper (2014).

[9] N. Harris and J. Curry, "Development and range testing of a Lo-RaWAN system in an urban environment", World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, vol. 12(1), 2018, pp. 43–51.

[10] P. Heino, "360-videoiden katselustatistiikan visualisointi", Master's thesis, 2018, Tampere University of Technology (TUT), Tampere, Finland.

[11] S. Herle, R. Becker, and J. Blankenbach, "Bridging GeoMQTT and REST", In Geospatial Sensor Webs Conference, 2016.

[12] HiveMQ, "MQTT Security Fundamentals: MQTT Payload Encryption", https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption, accessed 5th Dec. 2018.

[13] A. K Jain, "Data clustering: 50 years beyond K-means", Pattern recognition letters, vol. 31(8), Elsevier, 2010, pp. 651–666.

[14] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things", Transaction on IoT and Cloud Computing, vol. 3(1), 2015, pp. 11–17.

[15] R. K. Kodali and V. S. K. Gorantla, "Weather tracking system using MQTT and SQLite". 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), IEEE, 2017, pp. 205–208.

[16] R. K. Kodali and B. S. Sarjerao, "MQTT based air quality monitoring", In Humanitarian Technology Conference (R10-HTC), IEEE Region 10, IEEE, 2017, pp. 742–745.

[17] T. M. Kodinariya and P. R. Makwana, "Review on determining number of Cluster in K-Means Clustering", International Journal of Advance Research in Computer Science and Management Studies, vol. 1(6), 2013, pp. 90–95.

[18] W. C. Lo, C. L. Fan, J. Lee, C. Y. Huang, K. T. Chen, and C. H. Hsu, "360 Video Viewing Dataset in Head-Mounted Virtual Reality", In Proceedings of the 8th ACM on Multimedia Systems Conference, ACM, 2017, pp. 211–216.

[19] T. Löwe, M. Stengel, E. C. Förster, S. Grogorick, and M. Magnor, "Visualization and analysis of head movement and gaze data for immersive video in head-mounted displays". In Proceedings of the Workshop on Eye Tracking and Visualization (ETVIS), vol. 1, 2015.

[20] A. Luoto, "Towards Framework for Choosing 360-degree Video SDK", Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 1: SIGMAP, (ICETE), 2017, pp. 81–86.

[21] A. Luoto, "Lightweight Visualization and User Logging for Mobile 360-degree Videos. In 11th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), IEEE, in press.

[22] A. Luoto and K. Systä, "Fighting network restrictions of request-response pattern with MQTT", IET Software, 2018.

[23] S. Nicholas, "Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android", 2012, http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https, accessed 5th Dec. 2018.

[24] C. Ordonez, "Integrating K-means clustering with a relational DBMS using SQL", IEEE transactions on Knowledge and Data engineering vol. 18(2), 2006, pp. 188–201.

[25] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee "A design science research methodology for information systems research", Journal of management information systems, vol. 24(3), 2007, pp. 45–77.

[26] Periscopedata, "Multi-dimensional Clustering Using K-Means in Postgres SQL", 2015.

[27] 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, ACM, 2016, pp. 1–6.

[28] J. M. Ritchie, R. C. W. Sung, H. Rea, T. Lim, J. R. Corney, and I. Howley, "The use of non-intrusive user logging to capture engineering rationale, knowledge and intent during the product life cycle", In Portland International Conference on Management of Engineering & Technology, PICMET, IEEE, 2008, pp. 981–989.

[29] I. Skerrett, "IoT Developer Survey 2016", Eclipse IoT Working Group, IEEE IoT and Agile IoT, 2016, pp. 1–39.

[30] M. Uehara, "A case study on developing cloud of things devices", In Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), IEEE, 2015, pp. 44–49.

[31] V. Vaishnavi and W. Kuechler, “Design research in information systems”, 2004.

[32] M. Wenger, A. Zoitl, J. O. Blech, I. Peake, and L. Fernando, “Cloud based monitoring of timed events for industrial automation”, In 21st International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2015, pp. 827–830.

[33] C. Wu, Z. Tan, Z. Wang, and S. Yang, “A Dataset for Exploring User Behaviors in VR Spherical Video Streaming”, In Proceedings of the 8th ACM on Multimedia Systems Conference, ACM, 2017, pp. 193–198.

[34] T. Yokotani and Y. Sasaki, ”Comparison with HTTP and MQTT on required network resources for IoT”, In International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), IEEE, 2016, pp. 1–6.

[35] A. Zabasta, K. Kondratjevs, J. Peksa, and N. Kunicina, ”MQTT enabled service broker for implementation arrowhead core systems for automation of control of utility systems”. In 5th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), IEEE, 2017, pp. 1–6.

[36] L. Zhang, “Building Facebook Messenger”, 2011, https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920/, accessed 5th Dec. 2018.

PUBLICATION VI

Using IoT Platform for 360-degree Video User Logging and Analysis

A. Luoto, K. Systä, O. Hylli, and V. Heikkilä

DOI: 10.5220/0010015600410050

Using IoT Platform for 360-degree Video User Logging and Analysis

Antti Luoto[1][a], Kari Systä[1][b], Otto Hylli[1] and Ville Heikkilä[1]
[1]*Computing Sciences, Tampere University, Korkeakoulunkatu 1, Tampere, Finland*
antti.luoto@tuni.fi, kari.systa@tuni.fi, otto.hylli@tuni.fi, ville.heikkila@tuni.fi

Keywords: FIWARE, 360-degree video, IoT, MQTT, user data visualization

Abstract: Smart cities are getting more and more attention due to urbanization and IoT trends. At the same time, 360-degree videos are also getting more popular. The watchers of 360-degree videos provide a data source that fit to the data collection aim of smart cities. This paper explores how well 360-degree video user data can be collected, using MQTT as a data transfer protocol, and analyzed with an open source IoT platform. The results suggest that using MQTT with the chosen IoT platform is convenient and general chart visualizations can provide useful insight about 360-degree video watchers. The used research method is design science.

1 INTRODUCTION

The amount of people living in urban areas is expected to grow in the near future. This growth can lead to a variety of problems (Nam and Pardo, 2011). Smart city is a concept that offers a partial solution. Smart city has multiple definitions (Nam and Pardo, 2011), out of which, we can use one that sees the smart city as a combination of technologies that make the critical infrastructure components and services of a city more intelligent, interconnected and efficient (Washburn et al., 2009).

Smart cities have an interconnection with Internet of Things (IoT) (Su et al., 2011). The idea of IoT is that objects can be embedded with sensors that can be connected to cloud technologies via Internet. In smart city context, it means that city infrastructure can be equipped with sensors collecting data from various domains and sending the data to data collection platforms via Internet. Thus, the technologies that enable IoT also support smart cities.

Unfortunately, IoT suffers from a diverse set of initiatives, standards and implementations (Fersi, 2015) (Araujo et al., 2019). However, the recent interest in smart cities has motivated the development of IoT platforms that provide scalability, reliability, sustainability and security. One of such platforms is FIWARE. It is a framework of open source components to help the development of smart solutions (fiware.org, 2020b). We used FIWARE because it provides a number of reusable building blocks and it is present in many different sectors in Europe, for example, healthcare, telecommunications, environmental services and agriculture (Rodriguez et al., 2018). In addition, this study was done in a research project called CityIoT (CityIoT, 2020). One of the objectives of the project was to build a smart city IoT piloting environment for and this study works as one of the pilots. The project had not used a popular IoT data transfer protocol called Message Queuing Telemetry Transport (MQTT) and we thought it would be important to experiment with it.

360-degree video watchers provide an interesting data source for experimenting with an IoT platform in a smart city context. 360-degree videos are relatively popular nowadays (Qian et al., 2016) and they have applications in multiple smart city related domains such as surveillance, remote working, robotics, traffic, etc. 360-degree videos can be watched with smart phones that contain various sensors for collecting data.

An example benefit of 360-degree video watcher analysis in a smart city is that it is possible to analyze where video watchers focus their attention in traffic or near tourist attraction. By analyzing video watchers, it is possible, for example, to place advertisements so that they are often seen.

For analyzing 360-degree video watchers, it is important to log their view orientation since only a cropped view port of the video can be seen at a time. It is clear that lots of view orientation data can be collected from the users from a constantly updating video. However, what is not clear, is that how the data collection can be implemented with FIWARE

[a] https://orcid.org/0000-0002-9318-7665
[b] https://orcid.org/0000-0001-7371-0773

and MQTT, and how well general visualizations, such as two-dimensional charts provided by open source dashboard tool Grafana, support visualizing and analyzing view orientation data.

Thus, our research questions are the following. (1) How to use an IoT platform for collecting 360-degree video watching logs via MQTT? (2) What are the pros and cons of using a combination of an IoT platform and a general dashboard tool for visualizing 360-degree video watching logs?

In short, the experiment was successful. We were able to collect 360-degree video watching data from a smart phone to an IoT platform using MQTT and then use Grafana for creating useful visualization.

2 BACKGROUND

The most important data gathered from users is the rotation of their device while playing 360-degree video. Yaw, pitch and roll format is one way of presenting rotations in a spherical space. Yaw is the vertical rotation, pitch is the horizontal rotation, and roll is the rotation around front-to-back axis. We use that format as well but without roll. The main reason for that is that the API of Google VR SDK used by our smart phone application does not provide the roll angle.

One challenge with 360-degree video watcher logging is that the view orientation changes constantly when the user moves, turns, etc. If the view orientation is recorded for every frame in a 23 FPS video, a set of values is collected every 43th millisecond. Some back ends might suffer from such a fast pace if there are multiple users being logged simultaneously. Therefore, a trade-off decision must be made between the precision and the amount of data. For example, some other 360-degree video user logging studies have had a sampling rate of 7-10 Hz (Bao et al., 2016) (Nasrabadi et al., 2017). So, meaningful 360-degree video user analysis should be possible even if the view orientation is not recorded for every frame. We decided to experiment with a sampling rate of about two times a second. We were able to see nice results already with our 2-Hz pace and there should not be challenges in increasing the sampling rate for a small number of simultaneous users. Naturally, some accuracy is lost but the analysis is already inaccurate since the gaze orientation is not considered. If a faster sampling rate is needed and the performance suffers, the performance of FIWARE can be improved by scaling (Araujo et al., 2019).

It is possible to make expressive visualizations of 360-degree video watching logs on top of 360-degree video applications but implementing such visualizations can require writing custom source code (Luoto, 2019). Thus, it is beneficial to find out what can be visualized with general graphs, charts, etc. provided by web-based dashboard tools. What is lost in usefulness can be made up in the easiness of implementation.

MQTT is a lightweight publish-subscribe data transfer protocol aimed for constrained devices. While MQTT is mainly used to send data from low-resource IoT sensors to (edge) gateways, it can be used with smart phones as well, especially when a lot of small data packets are sent, request-response is not needed, and there can be multiple simultaneous data sources. MQTT is among the most popular IoT protocols (Skerrett, 2016). MQTT has potential for sending 360-degree video user data from a smart phone to an IoT platform. MQTT works via cellular or Wi-Fi connections.

3 RELATED WORK

We are not aware of studies where 360-degree video watching data that would have been collected or analyzed with IoT platforms. Neither, we are not aware of IoT platforms being used for traditional video user analysis. In general, there is a relatively small amount of research literature about user logging and analysis architectures in 360-degree video domain available.

There are some publications that connect IoT and 360-degree videos, such as, a study about using 360-degree videos for teaching IoT security (Okada et al., 2019). In another study, the combination of IoT infrastructure including 360-degree videos is used to generate VR spaces (You et al., 2018). However, these studies do not use IoT platform to collect or analyze video user orientation data.

3.1 Traditional Video and Smart City

There are clearly research efforts in using traditional videos (i.e non-360-degree videos) with smart cities. However, it seems that there is not a strong explicit connection between 360-degree videos and smart cities in the research literature. For example, in a systematic mapping review about big data in smart cities (Brohi et al., 2018), there is only one paper out of 65 that has word 'video' in its title. However, it is possible that some of the papers included in the review do discuss using videos with smart cities despite not having the word 'video' in title. Especially, 360-degree videos are not mentioned at all in the review.

That particular paper with word 'video' presents a study about real-time video processing for traffic control in a smart city context using Hadoop with

GPUs (Rathore et al., 2018). In contrast to our work, they study video processing whereas we concentrate on user analysis. There are also other video processing studies for traffic management in a smart city context such as a study about vehicle counting for smart cities (Trivedi et al., 2018) and a study about automated pedestrian data collection (Sayed et al., 2016).

3.2 360-degree Content and Smart City

Neither does a survey on 360-degree video streaming emphasize smart cities (Fan et al., 2019). However, they have included a study that discusses object tracking application that would, according to Fan et al., have use in various smart city applications (Delforouzi and Grzegorzek, 2017). Again, that study is on the field of video processing rather than on the field of video watcher analysis.

A study that explicitly discusses smart cities provides analysis of panoramic images (instead of panoramic videos) (Feriozzi et al., 2019). An example of relatively rare study, that explicitly connects 360-degree videos with smart cities, presents real-time annotation of 360-degree videos (Tang et al., 2018). The authors plan to extend their work by using it in a smart city context. Once again, their work is about video processing and not about video user analysis.

3.3 User Logging and Analysis

There are some 360-degree video watcher logging and analysis studies available. For the most part, their aim is not in multi-user logging, smart city context, nor making interactive web-based dashboards.

There are a few studies that offer a public 360-degree video user logging data. In the first one, the authors use an architecture primarily for local logging and they use the logs for creating saliency and motion maps (Lo et al., 2017). In the second one, the authors used the logs to create example statistics for analyzing users' navigation patterns (Corbillon et al., 2017). In the third one, the authors present preliminary analysis of their data set by presenting visualizations such as plotting gaze data over video, density maps, and gazing directions in a 3D graph (Wu et al., 2017). 3D graphs should be possible with a Grafana plugin, but we concentrated on 2D graphs.

Another study predicted head orientation with weighted linear regression (Qian et al., 2016). Such advanced analysis could be very difficult (or impossible) with our dashboards made with simple SQL queries. A study about 360-degree video streaming in 5G networks presents two 360-degree video user traces on a timeline (Sun et al., 2018), but the study does not concentrate on producing visualizations.

View Similarity visualization, which shows the angular proximity of all 360-degree video viewers' viewing directions over time, helps to quickly analyze attentional synchrony (Löwe et al., 2015). Such visualization would definitely be useful but making one would be difficult with our toolbox.

An exceptional study presents a platform for logging interaction in 360-degree multimedia (Bibiloni et al., 2018). In addition to logging view orientation, the authors log interactions such as pressing play or pause. They exceptionally discuss the logging and visualization architecture in detail and have a dashboard with web support. They present, for example, user activities on a timeline and view orientation histograms.

3.4 FIWARE, MQTT, and Grafana

There is some research on the performance of FIWARE available. In an extensive performance evaluation of FIWARE, the authors aimed to a real smart city scale with their testbed that can send data via MQTT (Araujo et al., 2019). Their conclusions include that FIWARE's IoT Agents do not scale well due to Node.js implementation. This could be a problem in a real-life use case, but in our experiments with only a few users, we did not encounter performance issues. Another FIWARE study includes performance evaluations of IoT components in an agricultural domain (Martínez et al., 2016).

In 360-degree video context, Grafana has been used at least for monitoring tile-based streaming (Tagami et al., 2019), and potentially for monitoring 5G network while streaming 360-degree video (Kanstrén et al., 2018).

4 METHODOLOGY

The research was conducted by following design science methodology. It is a research method used in software engineering (Vaishnavi and Kuechler, 2004) that includes six steps: *problem identification and motivation*, *definition of the objectives for a solution*, *design and development*, *demonstration*, *evaluation and communication* (Peffers et al., 2007).

We include the steps of design science in this publication as follows. *Problem identification and motivation* is presented in Sections 1-3 but it is summarized here: the problem is that 360-degree video watchers could provide useful data in smart city context but logging and analyzing users requires infrastructure. We think that usage of IoT infrastructure and

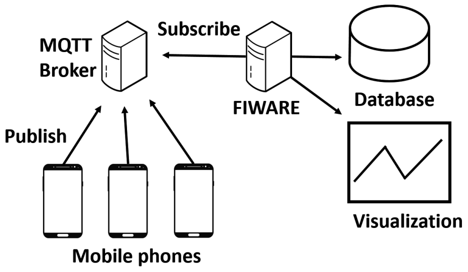

Figure 1: How MQTT integrates with FIWARE platform.

techniques, such as, IoT platform FIWARE, MQTT, and visualization tool Grafana can provide a promising solution for the data collection and analysis needs. The essential parts of *design and development* are presented in Section 4. *Demonstration* is included in Section 5 by presenting the produced visualizations. *Evaluation* is included in Sections 5-6 by stating the essential pros and cons of the approach, and *Communication* is covered by writing the publication and the presenting the work to a scientific audience. Lastly, to *define the objectives for a solution*, our objective is to use FIWARE so that 360-degree video user log transferred via MQTT can be visualized with Grafana conveniently.

4.1 Architecture Overview

We used FIWARE platform as the tool for providing communication interfaces, storing data, and creating visualizations. An overview of the architecture can be seen in Figure 1. The idea is that smart phones send data to MQTT broker which delivers the data to IoT Agent in FIWARE platform. That data is then stored to a database and used for creating visualizations.

4.2 FIWARE

FIWARE is a framework of open source components to accelerate the development of smart solutions (fiware.org, 2020b). Orion Context Broker is a core component of the system. It enables the system to perform updates and access the data via FIWARE Next Generation Service Interface version 2 (NGSIv2) API. The Context Broker is surrounded by a set of additional components which may be gathering data from diverse sources, such as mobile applications or IoT sensors, help with data processing, analysis, and visualization of data.

4.3 IoT Agent

Among the mentioned 'additional components', FIWARE provides components called IoT Agents. "An IoT Agent is a component that lets groups of devices send their data to and be managed from a FIWARE NGSI Context Broker using their own native protocols" (fiware.org, 2020a). There are a few IoT Agents that support MQTT. We decided to use IoT Agent for Ultralight since it seemed easy to try Ultralight format with our simple log data. IoT Agent for Ultralight is a bridge that can be used for communication between the devices using Ultralight 2.0 protocol and NGSI Context Brokers (fiware.org, 2019).

Ultralight 2.0 is a lightweight text-based protocol aimed to constrained devices. It is used by sensor devices to send data to IoT platform. Ultralight 2.0 does not order the use of communication protocol, only the format the of the payload. The payload follows the format of the example: "a|1|b|2", which includes values for two attributes: value 1 for attribute a and value 2 for attribute b. When compared to JSON, the format makes a shorter payload which helps saving resources. The usage of Ultralight was not very essential in this experiment. The format is just explained here to clarify a few things later.

After installing IoT Agent, the following steps are required for setting up devices communicating via MQTT (fiware.org, 2020c). Figure 2 summarizes the steps.

1. Provisioning a service group. The idea of 'service group' is to create a top level MQTT topic for a group of devices related to the same service. A service group is provisioned by making an HTTP POST to Service API of IoT Agent (Telefonica IoT, nd). The request payload defines the base MQTT topic for a group of devices that form a service.

2. Provisioning a device: HTTP POST which contains information about a new device. In our case, 'device' is a vague concept meaning that any entity can be a 'device'. For example, the 'devices' we provision are 360-degree videos, and view orientations in a view session of a 360-degree video. While 'view orientation' is not a real device, it constains sensor measurements. Here is an example of HTTP POST made with curl that adds an entity for 360-degree video view orientation:

```
curl -iX POST 'http://<host>/iot/devices' \
    -H 'Content-Type: application/json' \
    -H 'fiware-service: 360video' \
    -H 'fiware-servicepath: /' \
    -d '{ "devices": [
{
    "device_id":
        "viewOrientationInViewSession001",
```

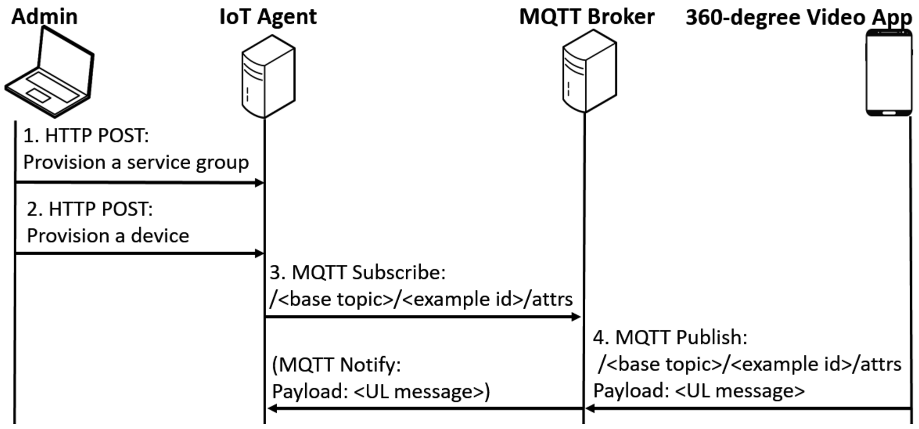

Figure 2: The steps for setting up devices communicating via MQTT.

```
    "entity_type": "Device",
    "transport": "MQTT",
    "attributes": [
        {
            "object_id": "y",
            "name": "yaw",
            "type": "Integer" },
        {
            "object_id": "p",
            "name": "pitch",
            "type": "Integer" },
        {
            "object_id": "t",
            "name": "videoTime",
            "type": "Integer" } ],
    "static_attributes": [
        {   "name":"ref360Video",
            "type": "Relationship",
            "value": "360VideoExample" } ]
} ] } '
```

IoT Agent HTTP API is described in (Telefonica IoT, nd). The 'device' resource (<host>/iot/devices) is used to publish data to context broker via IoT Agent. There are two mandatory HTTP headers: fiware-service and fiware-servicepath. FIWARE service is a multi-tenancy feature which ensures that entities, attributes and subscriptions inside one service are invisible to other services (fiware.org, 2018a). Service path is a hierarchical scope where entities can be divided to hierarchies (fiware.org, 2018b). We did not use service paths as they only work with HTTP and we used on MQTT. The actual data is sent as a JSON object that has an attribute named 'devices' which contains a list of devices to be provisioned.

3. MQTT subscription: setting the value 'MQTT' for the attribute 'transport' is enough for the IoT Agent to subscribe to the service group base topic (made in step 1) extended with the device id, for example, /<base topic>/<device-id> (fiware.org, 2020c). The 'attributes' list contains attributes that are active readings from the device. It has also a mapping from abbreviated Ultralight 2.0 attributes to actual entity attributes. For example, an entity attribute named 'yaw' can be mapped to 'y'. Static attributes can be also defined. The idea of those is that their values cannot be changed via the chosen IoT protocol and they are initialized in the provisioning phase. In our example, we add only one static attribute that is a reference to the video being watched.

4. MQTT message: attributes of the added entity can be updated with an MQTT message. For example, the view orientation entity can be updated by publishing a message to the base topic that is catenated with a device id and '/attrs' string, for example, /<base topic>/ExampleId/attrs. The payload of the message follows the Ultralight 2.0 format, for example, "y|15.05 |p|0.50|t|1234" where yaw is 15.05 degrees, pitch is 0.50 degrees and videoTime is 1234 milliseconds of video time.

4.4 Smart Phone Application

The smart phone application uses Google VR SDK for Android. It has a 360-degree video player with basic controls. Figure 3 presents an example view on the application. The application sends event-based measurements to FIWARE, in other words, it does not send only updated values.

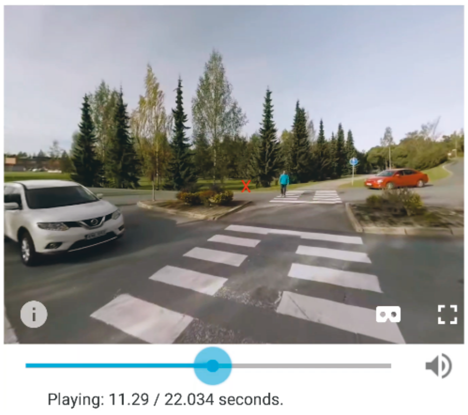

Figure 3: Screenshot of the smart phone application.

4.5 Visualization

Grafana is an open source visualization dashboard platform for multiple databases. We used version 6.5.3. The default graph visualization provided by Grafana is not conventional for visualizing user traces of non-streaming videos since the charts require having real-world timestamps on X axis. Luckily, a Grafana plugin named Plotly allows using any data on X axis which allows setting video time on X axis.

4.6 Database

By default, FIWARE's Orion Context Broker uses MongoDB and stores only the latest value of the attribute. That is not conventional for time-based analysis. Luckily, FIWARE component QuantumLeap offers a database that can be used to store data as time series data that can be visualized with Grafana. The data is not automatically copied from Orion to QuantumLeap unless there is a subscription for that. Thus, we made a subscription to Orion using the `/notify` endpoint of the QuantumLeap API.

5 RESULTS

This section discusses the three Grafana visualizations, 'Graph', 'Plotly', and 'Table', and other related features that were used. We present the visualizations we made and remark other observations and experiences gathered while using the visualization tool, and the IoT platform in general. The data in the visualizations is a result of an SQL query that is written individually for each visualization via Grafana UI.

5.1 Graph

The graph visualization is a general-purpose tool for visualizing data on timeline. FIWARE automatically generates a timestamp for logged data records which helps making timelines. However, timeline charts do not allow using other than real-world timestamp data on X axis. That makes analyzing user data of non-streaming videos difficult since the video time is often more important than the moment of time when the video was being watched. On the other hand, for streaming videos real-world timestamps can be useful because streaming fits better in real-world time.

Graph tool also provides histograms which can show counted data. In 360-degree video user analysis, this feature can be used, for example, to see which are the most watched seconds of the video as presented in Figure 4. It is a clear and quick visualization that can be used, for example, for analyzing which parts of the video people consider interesting. For example, in Figure 4 more people have watched the start of the video than the end of the video.

5.2 Plotly

Plotly plugin is not installed by default in Grafana. Plotly allows using any data on X axis which makes it useful for making visualizations that are not dependent on real-world time.

Figure 5 presents a Plotly visualization that shows all the recorded view orientations within a single video. On X axis there is video time in milliseconds, and both the yaw and pitch are on Y axis in degrees. Those measurements were selected because they provide the device orientation at the certain moment in video time. With the visualization, it is possible to get an overall impression where multiple users have watched during video playback. Thus, it is possible to see, for example, that most users have watched to yaw direction 0 between -50 degrees at the video time of 10 seconds.

Figure 6 presents a visualization of a single-user view session trace. Similarly to visualization presented in Figure 5, it contains video time on X axis, and both the yaw and pitch on Y axis. The wanted view session can be selected using a drop-down menu on top of the dashboard. The visualization also shows that our two Hz sampling rate can be enough for analyzing a single user at least in some situations. When compared to multi-user visualization in Figure 5, a single-user visualization is clearer and helps concentrating on an interesting user.

Plotly supports showing additional information about the data point with a hover tooltip. Only one

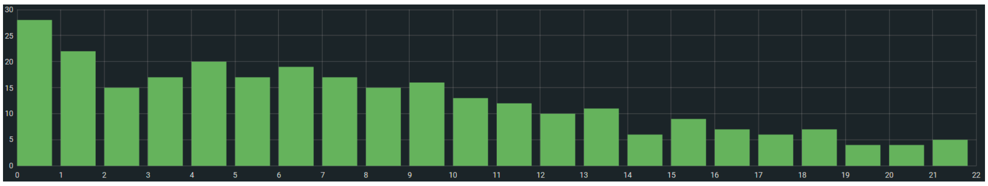
Figure 4: Most watched seconds of the video.

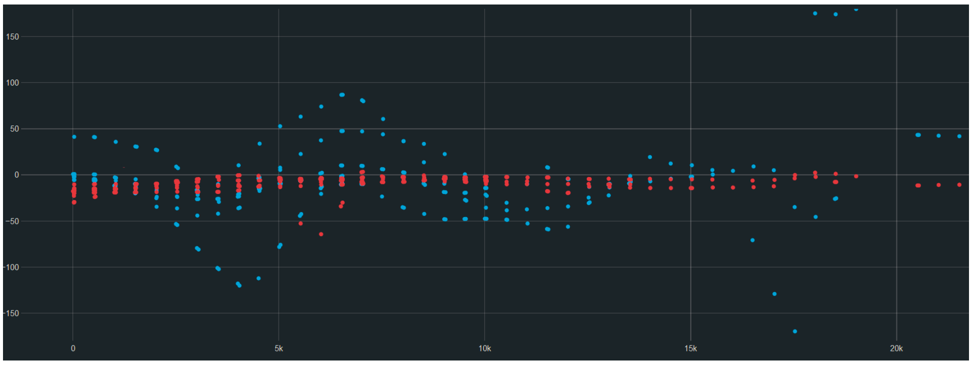
Figure 5: All the data from all the view sessions for a single video. Blue is for yaw and red is for pitch.

field for additional information is available, so adding more information to the tooltip requires using string functions, such as concat, in SQL query.

5.3 Table

Table visualization provides a way a see data in a basic tabular format. The pros include a detailed view on the raw data and a way to choose and organize columns freely. However, it is not a good tool for getting an overall picture.

5.4 Drop-down Menu

Grafana allows creating a drop-down menus for making more dynamic graphs. A chosen value is then used in SQL queries that generate the graphs. Drop-down menus can be seen in top of Figure 6. The Figure has three drop-down menus: Start time, End time, and View session.

As an example, to make a drop-down menu, we defined a variable called VideoStartTime that can be used to adjust the video start time:

```
SELECT videoTime / 1000
FROM mt360video.etdevice
ORDER BY 1
```

The query is explained in detail in the following. SELECT: videoTime is divided by 1000 to convert it to seconds because using milliseconds in a drop-down menu would be inconvenient, FROM: QuantumLeap makes a table for every FIWARE service and entity type pair where the used table name is mt<fiware_service>.et<entity_type>, and ORDER BY: the result needs to be ordered to be user friendly in a drop-down menu.

The variable can then be used in SQL queries for the graphs in the following way:

```
SELECT yaw, pitch, videoTime, entity_Id
FROM mt360video.etdevice
WHERE entity_Id = '$entity_Id'
AND videoTime BETWEEN '$VideoStartTime' * 1000
AND '$VideoEndTime' * 1000
```

The variables are used in the queries by adding a dollar sign to the beginning of the variable name and wrapping the string with single quotation marks. The variables for video start and end times need to be multiplied by 1000 to convert seconds back to milliseconds. The resulting visualization, with video time set between 0 to 22 seconds, can be seen in Figure 6.

5.5 Pros and Cons

To answer the second research question and to generalize the results, we ponder what are the pros and cons of using a general purpose IoT platform and a dashboard tool for storing and analyzing 360-degree

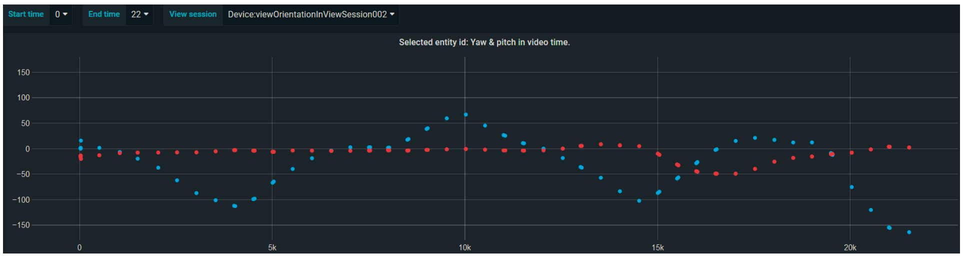

Figure 6: Selected view session.

video user data. By no means using an IoT platform for the described case is necessary. However, using one is meaningful considering the following **pros**:

General building blocks and general workflow: it is possible to add different kind of devices and send diverse data for storage using the same workflow. Once installed, a general purpose IoT platform can be used for various use cases.

High abstraction level: APIs help with abstraction and many operations are automated – developers do not need to worry about details, such as the database solution. For example, IoT Agent of FIWARE provides an useful API for provisioning devices.

MQTT support: since MQTT is an important IoT protocol, it is expected to be supported by IoT platforms. For example, FIWARE supports MQTT via IoT Agent.

General visualizations: since it is a common practice to visualize IoT data with dashboards, IoT platforms often offer a general visualization tool. According to our experiments, a general web-based visualization tool can provide an easy way for creating useful 360-degree video user log visualizations.

Naturally there are also **cons** in using an IoT platform for 360-degree video user analysis. We were able to cope with all of them, but the cons include:

Installation, maintenance, etc.: lots of work can be required for setting up and maintaining an IoT platform. However, this work can be outsourced and reused at least to some extent.

Complexity: using a whole IoT platform for logging and visualization naturally increases complexity of the system. Taking FIWARE as an example, while much of the complexity is located under the hood and many details of the FIWARE core components are not important in the context of this publication, knowledge of the whole system is important, for example, for debugging.

Lack of specialized visualizations: 360-degree video user logs are a special case of visualization where, for example, having the video and visualization overlapping or placed next to each other could be useful, but having such a special visualization out-of-the-box cannot be expected from a general IoT platform.

Defects: IoT platforms can have defects. Using FIWARE as an example, we experienced that the used IoT Agent assumes that new entities are devices. It is conceptually confusing when adding entities that are not clear devices. Further, FIWARE provides ready-made data models which can help in many use cases (Smart Cities, Smart Agrifood, Smart Environment, Smart Energy, etc.), but we did not find a useful data model for our data. We also observed some errors in the documentation of FIWARE.

Poor UI: when using general components, UI is not necessarily optimized for special use cases. For example, visualizations can get messy if there are overlapping traces, similarly to Figure 5. Drop-down menus can be used for limiting the shown data in Grafana, but a more user-friendly way, for example, by selecting a trace with a mouse, would be nice.

6 DISCUSSION

We mostly concentrated on using the IoT Agent with MQTT and making visualizations with Grafana. However, there are other aspects to be taken into account when creating an implementation for public use. The functionality described in this publication can be naturally implemented without an IoT platform, but our research interests were aimed at what are the benefits of using general IoT infrastructure.

We used only a single 22-seconds video. For longer videos UX will need to be considered better. E.g., a drop-down menu for every second in video that is five minutes long would be an annoying to use. Further, we did not see the difficulties of visualizing multiple videos making comparisons between two videos.

One might argue that using MQTT and Ultralight, that are meant for constrained devices, is not convenient with modern smart phones that do not lack memory. However, we wanted to try the MQTT support of FIWARE, and 360-degree video user logging provides a domain that produces data that goes well with IoT. Since the amount of data is expected to grow in the future, it is good to prepare by using lightweight technologies. In addition, the low memory and CPU footprint of MQTT helps saving batteries.

Further, one might argue that IoT communication systems are designed to transport data from a vast amount of distributed sensors. That is true, and in a future smart city there is a vast amount of sensors, and the 360-degree user data provided by smart phone sensors is just a small subset. Probably it is not meaningful to set up an IoT platform just for one use case, but the platform should be used for many use cases collecting data from various domains.

7 CONCLUSIONS & FUTURE WORK

The experiment was successful, using MQTT with FIWARE platform was relatively easy, and making visualizations with Grafana was practical. We managed to implement many useful 360-degree video watcher log visualizations with a relatively low effort. The biggest challenges were related to some inconsistencies with FIWARE documentation when performing the steps of setting up MQTT communication.

The 360-degree videos are often divided to tiles, for example, to deliver only the needed parts of the video in high-quality. Tiles can be used in user analysis as well. It can be enough to know which tiles are seen by the user instead of the exact yaw and pitch orientation. Integrating object detection algorithms for 360-degree video content analysis would be interesting as well.

While we mostly discuss non-streaming videos, the used approach could be more useful with streaming videos since many smart city applications require streaming video. Using alerts in Grafana could be useful for automating parts of the analysis.

Smart cities are more efficient when smart objects and applications operate without human intervention. However, there are situations where a human is required to make decisions with the help of collected data. Visualizations help with such decisions and are a steppingstone on the way towards autonomous smart city.

REFERENCES

Araujo, V., Mitra, K., Saguna, S., and Åhlund, C. (2019). Performance evaluation of fiware: A cloud-based iot platform for smart cities. *Journal of Parallel and Distributed Computing*, 132:250–261.

Bao, Y., Wu, H., Zhang, T., Ramli, A. A., and Liu, X. (2016). Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1161–1170. IEEE.

Bibiloni, T., Oliver, A., and del Molino, J. (2018). Automatic collection of user behavior in 360 multimedia. *Multimedia Tools and Applications*, 77(16):20597–20614.

Brohi, S. N., Bamiah, M., and Brohi, M. N. (2018). Big data in smart cities: a systematic mapping review. *Journal of Engineering Science and Technology*, 13(7):2246–2270.

CityIoT (2020). Cityiot – future operator independent data integration platform. https://www.cityiot.fi/english Last accessed: August 12, 2020.

Corbillon, X., De Simone, F., and Simon, G. (2017). 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, page 199–204, New York, NY, USA. Association for Computing Machinery.

Delforouzi, A. and Grzegorzek, M. (2017). Robust and fast object tracking for challenging 360-degree videos. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 274–277. IEEE.

Fan, C.-L., Lo, W.-C., Pai, Y.-T., and Hsu, C.-H. (2019). A survey on 360 video streaming: Acquisition, transmission, and display. *ACM Computing Surveys (CSUR)*, 52(4):1–36.

Feriozzi, R., Meschini, A., Rossi, D., and Sicuranza, F. (2019). Virtual tours for smart cities: A comparative photogrammetric approach for locating hot-spots in spherical panoramas. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W9:347–353.

Fersi, G. (2015). Middleware for internet of things: A study. In *2015 International Conference on Distributed Computing in Sensor Systems*, pages 230–235. IEEE.

fiware.org (2018a). Multi tenancy. https://github.com/telefonicaid/fiware-orion/blob/c59401dcead76a77e28daddbd0e127628b19422d/doc/manuals/user/multitenancy.md Last accessed: April 3, 2020.

fiware.org (2018b). Service paths. https://github.com/telefonicaid/fiware-orion/blob/d07ced613432237728c0983e11223a5d68d1e163/doc/manuals/user/service_path.md Last accessed: April 3, 2020.

fiware.org (2019). Iot agent for the ultralight 2.0 protocol. https://github.com/telefonicaid/iotagent-ul/tree/8557733aaac1a7428f295eec7b74dac8b805e91e Last accessed: April 3, 2020.

fiware.org (2020a). Fiware iot agent node.js library. https://github.com/telefonicaid/iotagent-node-lib/tree/d76d0216f6d2247bcc2131ebbf81c74867afa447 Last accessed: April 3, 2020.

fiware.org (2020b). What is fiware. https://www.fiware.org/about-us/ Last accessed: April 3, 2020.

fiware.org (2020c). What is mqtt? https://github.com/FIWARE/tutorials.IoT-over-MQTT/tree/c1c27aa7d29a388001d62d0c51f2d8df66208123 Last accessed: April 3, 2020.

Kanstrén, T., Mäkelä, J., Uitto, M., Apilo, O., Pouttu, A., Liinamaa, O., Destino, G., Kivinen, P., and Matilainen, A. (2018). Vertical use cases in the finnish 5g test network. In *2018 European Conference on Networks and Communications (EuCNC)*, pages 329–334. IEEE.

Lo, W.-C., Fan, C.-L., Lee, J., Huang, C.-Y., Chen, K.-T., and Hsu, C.-H. (2017). 360-degree video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 211–216. ACM.

Löwe, T., Stengel, M., Förster, E.-C., Grogorick, S., and Magnor, M. (2015). Visualization and analysis of head movement and gaze data for immersive video in head-mounted displays. In *Proceedings of the Workshop on Eye Tracking and Visualization (ETVIS)*, volume 1. Citeseer.

Luoto, A. (2019). Log analysis of 360-degree video users via mqtt. In *Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis*, pages 130–137.

Martínez, R., Pastor, J. Á., Álvarez, B., and Iborra, A. (2016). A testbed to evaluate the fiware-based iot platform in the domain of precision agriculture. *Sensors*, 16(11):1979.

Nam, T. and Pardo, T. A. (2011). Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times*, pages 282–291.

Nasrabadi, A. T., Mahzari, A., Beshay, J. D., and Prakash, R. (2017). Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1689–1697.

Okada, Y., Haga, A., Wei, S., Ma, C., Kulshrestha, S., and Bose, R. (2019). E-learning material development framework supporting 360vr images/videos based on linked data for iot security education. In *International Conference on Emerging Internetworking, Data & Web Technologies*, pages 148–160. Springer.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77.

Qian, F., Ji, L., Han, B., and Gopalakrishnan, V. (2016). Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pages 1–6. ACM.

Rathore, M. M., Son, H., Ahmad, A., and Paul, A. (2018). Real-time video processing for traffic control in smart city using hadoop ecosystem with gpus. *Soft Computing*, 22(5):1533–1544.

Rodriguez, M. A., Cuenca, L., and Ortiz, A. (2018). Fiware open source standard platform in smart farming-a review. In *Working Conference on Virtual Enterprises*, pages 581–589. Springer.

Sayed, T., Zaki, M., and Tageldin, A. (2016). Automated pedestrians data collection using computer vision. In *Smart City 360*, volume 166, pages 31–43.

Skerrett, I. (2016). Iot developer survey 2016. *Eclipse IoT Working Group, IEEE IoT and Agile IoT*, pages 1–39.

Su, K., Li, J., and Fu, H. (2011). Smart city and the applications. In *2011 international conference on electronics, communications and control (ICECC)*, pages 1028–1031. IEEE.

Sun, L., Duanmu, F., Liu, Y., Wang, Y., Ye, Y., Shi, H., and Dai, D. (2018). Multi-path multi-tier 360-degree video streaming in 5g networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 162–173.

Tagami, A., Ueda, K., Lukita, R., De Benedetto, J., Arumaithurai, M., Rossi, G., Detti, A., and Hasegawa, T. (2019). Tile-based panoramic live video streaming on icn. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE.

Tang, L., Subramony, H., Chen, W., Ha, J., Moustafa, H., Sirlapu, T., Deshpande, G., and Kwasniewska, A. (2018). Edge assisted efficient data annotation for realtime video big data. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 6197–6201. IEEE.

Telefonica IoT (n.d.). Iot agent provision api documentacion. https://telefonicaiotiotagents.docs.apiary.io Last accessed: April 3, 2020.

Trivedi, J., Devi, M. S., and Dhara, D. (2018). Vehicle counting module design in small scale for traffic management in smart city. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–6. IEEE.

Vaishnavi, V. and Kuechler, B. (2004). Design science research in information systems. *Association for Information Systems*.

Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N., and Nelson, L. E. (2009). Helping cios understand "smart city" initiatives. *Growth*, 17(2):1–17.

Wu, C., Tan, Z., Wang, Z., and Yang, S. (2017). A dataset for exploring user behaviors in vr spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 193–198. ACM.

You, D., Seo, B.-S., Jeong, E., and Kim, D. H. (2018). Internet of things (iot) for seamless virtual reality space: Challenges and perspectives. *IEEE Access*, 6:40439–40449.