Enhancing video applications through timed metadata

Emmanouil Potetsianakis

université PARIS-SACLAY

NNT : 2019SACLT029

Thèse de doctorat

TELECOM Paris

IP PARIS

Enhancing Video Applications Through Timed Metadata

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom Paris

Ecole doctorale n°580 Sciences et technologies de l'information (STIC)
Spécialité de doctorat : Big Data, Knowledge, Machine Learning, and Interactions

Thèse présentée et soutenue à Paris, le 12/09/2019, par

EMMANOUIL POTETSIANAKIS

Composition du Jury :

Michel Beaudouin-Lafon Professor, Université Paris-Sud	Examinateur
Gwendal Simon Professor, IMT Atlantique	Rapporteur
Christian Timmerer Assoc. Professor, Alpen-Adria-Universität Klagenfurt	Rapporteur
Anissa Mokraoui Professor, Université Paris 13 (L2TI)	Président
Jean Le Feuvre Assoc. Professor, Telecom Paris	Co-encadrant de thèse
Jean-Claude Dufourd Directeur d'Etudes, Telecom Paris	Co-directeur de thèse

Emmanouil Potetsianakis: *Enhancing Video Applications Through Timed Metadata,* Paris, September 2019

ABSTRACT

Video recording devices are often equipped with sensors (smartphones for example, with GPS receiver, gyroscope etc.), or used in settings where sensors are present (e.g. monitor cameras, in areas with temperature and/or humidity sensors). As a result, many systems process and distribute video together with timed metadata streams, often sourced as User-Generated Content. Video delivery has been thoroughly studied, however timed metadata streams have varying characteristics and forms, thus a consistent and effective way to handle them in conjunction with the video streams does not exist.

In this Thesis we study ways to *enhance video applications through timed metadata*. We define as timed metadata all the non-audiovisual data recorded or produced, that are relevant to a specific time on the media timeline.

"Enhancing" video applications has a double meaning, and this work consists of two respective parts. First, using the timed metadata to *extend* the capabilities of multimedia applications, by introducing novel functionalities. Second, using the timed metadata to *improve* the content delivery for such applications.

To extend multimedia applications, we have taken an exploratory approach, and we demonstrate two use cases with application examples. In the first case, timed metadata is used as input for generating content, and in the second, it is used to extend the navigational capabilities for the underlying multimedia content. By designing and implementing two different application scenarios we were able to identify the potential and limitations of video systems with timed metadata.

We use the findings from the first part, to work from the perspective of enhancing video applications, by using the timed metadata to improve delivery of the content. More specifically, we study the use of timed metadata for multi-variable adaptation in multi-view video delivery - and we test our proposals on one of the platforms developed previously. Our final contribution is a buffering scheme for synchronous and low-latency playback in live streaming systems.

RESUMÉ

Les dispositifs d'enregistrement vidéo sont souvent équipés de capteurs (smartphones par exemple, avec récepteur GPS, gyroscope, etc.) ou utilisés dans des systèmes où des capteurs sont présents (par exemple, caméras de surveillance, zones avec capteurs de température et/ou d'humidité). Par conséquent, de nombreux systèmes traitent et distribuent la vidéo avec des flux de métadonnées temporels, souvent sous forme de contenu généré par l'utilisateur (UGC). La diffusion vidéo a fait l'objet d'études approfondies, mais les flux de métadonnées ont des caractéristiques et des formes différentes, et il n'existe en pratique pas de méthode cohérente et efficace pour les traiter conjointement avec les flux vidéo.

Dans cette thèse, nous étudions les moyens d'améliorer les applications vidéo grâce aux métadonnées temporelles. Nous définissons comme métadonnées temporelles toutes les données non audiovisuelles enregistrées ou produites, qui sont pertinentes à un moment précis sur la ligne de temps du média.

"L'amélioration" des applications vidéo a une double signification, et ce travail se compose de deux parties respectives. Premièrement, utiliser les métadonnées temporelles pour étendre les capacités des applications multimédias, en introduisant de nouvelles fonctionnalités. Deuxièmement, utiliser les métadonnées chronométrées pour améliorer la distribution de contenu pour de telles applications.

Pour l'extension d'applications multimédias, nous avons adopté une approche exploratoire et nous présentons deux cas d'utilisation avec des exemples d'application. Dans le premier cas, les métadonnées temporelles sont utilisées comme données d'entrée pour générer du contenu, et dans le second, elles sont utilisées pour étendre les capacités de navigation pour le contenu multimédia sous-jacent. En concevant et en mettant en œuvre deux scénarios d'application différents, nous avons pu identifier le potentiel et les limites des systèmes vidéo avec métadonnées temporelles.

Nous utilisons les résultats de la première partie afin d'améliorer les applications vidéo, en utilisant les métadonnées temporelles pour optimiser la diffusion du contenu. Plus précisément, nous étudions l'utilisation de métadonnées temporelles pour l'adaptation multi-variables dans la diffusion vidéo multi-vues et nous testons nos propositions sur une des plateformes développées précédemment. Notre dernière contribution est un système de buffering pour la lecture synchrone et à faible latence dans les systèmes de streaming en direct.

ACKNOWLEDGEMENTS

This Thesis came to fruition due to the avail, support and collaboration offered by a series of people within my work, education and personal environment. I am thankful to everyone that helped me either directly or indirectly on my research, and/or on easing the strain that undertaking the task of completing a PhD could cause.

I am grateful to the director of my Thesis, Jean-Claude Dufourd, for providing me with feedback on my work when requested and for fighting the omnipresent bureaucratic hurdles.

I would also like to express my gratitude to the advisor of my Thesis, Jean Le Feuvre, foremost for inspiring me by being an exemplary researcher, also for offering his remarkable insights on a variety of fields, and for always asking the right questions, using his notable perceptiveness.

Having Jean and Jean-Claude as supervisors provided me with the unique opportunity of being able to work on projects only bounded by my imagination and research interests. I am beholden to both of them for creating this setting and for their guidance to explore it as a means to my PhD.

I much appreciate the rest of the current and alumni members of the Multimedia team of Télécom Paris for rendering my days at the office interesting. Additionally, I would like to thank my colleagues Manos, Bruno, Marc and James from the DIVA team of Télécom Paris, for generously providing me with both their HCI expertise and their companionship.

Finally, there are no words to describe the extend of adoration and appreciation I feel towards my parents, Vaggelis and Matina, and my brother Kostas, who offered me unlimited and unconditional love, support and encouragement in every single step of my life leading to this moment.

CONTENTS

I TIMED METADATA & VIDEO APPLICATIONS 1

1 INTRODUCTION 3
- 1.1 Video Streams with Timed Metadata 4
- 1.2 Characteristics and Challenges 6
- 1.3 Classification of Extended AV Streams Systems 10
- 1.4 Architectures for Delivery of Extended AV Streams . . 12

2 ESSENTIAL ELEMENTS OF THE STATE OF THE ART 15
- 2.1 Standardization Efforts 19
- 2.2 Delivery of Video with Timed Metadata 20

II EXTENDING VIDEO APPLICATIONS 23

3 TIMED METADATA FOR CONTROL OF INTERACTIVE MULTIMEDIA APPLICATIONS 25
- 3.1 Scenario Description 28
- 3.2 State of The Art 28
- 3.3 Audio Synthesis Application Example 30
- 3.4 System Performance 36
- 3.5 Discussion 37

4 SPATIOTEMPORAL NAVIGATION FOR EXTENDED VIDEO STREAMS 39
- 4.1 State of The Art 40
- 4.2 Spatiotemporal Video Navigation 46
- 4.3 Software For Adaptive Playback of User-Generated Content (SWAPUGC) 49
- 4.4 Discussion 53

III IMPROVING VIDEO DELIVERY 55

5 TIMED METADATA-BASED ADAPTATION OF SENSOR-ENRICHED VIDEO STREAMS 57
- 5.1 State of the Art 58
- 5.2 Proposal Overview 61
- 5.3 Stream Selection Policies 63
- 5.4 Implementation Considerations 67
- 5.5 Experimental Setup 70
- 5.6 User Study 78
- 5.7 Discussion and Future Work 84

6 BUFFER MANAGEMENT FOR SYNCHRONOUS AND LOW-LATENCY PLAYBACK OF MULTI-STREAM USER GENERATED CONTENT 87
- 6.1 State of The Art 89
- 6.2 Client-side Buffering Scheme 90
- 6.3 Discussion 96

IV CONCLUSION AND DELIVERABLES 99
7 CONCLUSION 101
7.1 Outlook and Potential Application Fields 102
7.2 Research Perspectives and Future Work 103
8 DELIVERABLES 105
8.1 Publications 105
8.2 Software 106

BIBLIOGRAPHY 108
A APPENDIX 121
A.1 Buffer-based Adaptation algorithm 121
A.2 Field-of-View and Region-of-Interest 123
A.3 User Study Feedback 125
A.4 Buffer Notation Used 130

LIST OF FIGURES

Figure 1 Examples of extended AV streams systems . . 5
Figure 2 Illustration of different content consumption approaches 8
Figure 3 Outline of AV system architecture 12
Figure 4 Outline of extended AV system architecture . . 13
Figure 5 Schema of extended AV server 13
Figure 6 Services specification for the Padova Smart City project [123] 16
Figure 7 Comparison between visual motion tracking and WSN based solution for rehabilitation supervision 16
Figure 8 Concept of MPEG-V sensory effects (from [111]) 20
Figure 9 Architecture of the SEVino tool (from [111]) . . 21
Figure 10 Overview of data processing paradigms in literature 22
Figure 11 Producer outline for audio (a) synthesis (b) editing 25
Figure 12 Audio output based producer-consumer chain 26
Figure 13 Controller output based producer-consumer chain 26
Figure 14 Controller output based producer-consumer Chain 27
Figure 15 Skeleton with the Joints used in the application highlighted 32
Figure 16 Audio Engine architecture 32
Figure 17 Configuration panel of the Unveil reverb-removal tool 33
Figure 18 Application data handling during runtime . . 34
Figure 19 Visualization examples 35
Figure 20 Instagram screenshot of "Bridge Dom Luis" results 41
Figure 21 Google Maps screenshot of "Bridge Dom Luis" results 42
Figure 22 GeoUGV captures 43
Figure 23 Interfaces of multi-view video navigation (from [6]) 44
Figure 24 NEWSMAN Middlebox scheduler architecture overview (from [98]) 45
Figure 25 Screenshot of Spatiotemporal Video Navigation 46
Figure 26 Outlines of implemented architectures 49
Figure 27 Screenshot of the SWAPUGC implemented application 52
Figure 28 System architecture overview 59

Figure 29 AVRS end-to-end system design (from [69]) . . 59
Figure 30 (a) Sensor, Content analysis methods and (b) comparison (from [69]) 60
Figure 31 Visualization of available views 72
Figure 32 Scores over time for all the videos used 73
Figure 33 Image Quality metric (Iq) over time for all the videos used 74
Figure 34 Shakiness metric (Ss) per video, over time for all the videos used 74
Figure 35 Roll & Tilt metric (St) per video, over time for all the videos used 75
Figure 36 Scores over time for the streams selected by cinematic-only and proposed algorithms . . . 75
Figure 37 Scores over time for streams selected by respective algorithms, paired by network trace used 76
Figure 38 Visualization of the representation supported by the simulated throughput, per trace over time 78
Figure 39 Scores given for S3, for all videos, by user ID . 80
Figure 40 Average scores per statement, by video ID . . . 81
Figure 41 Average scores for Cinematic and Metric-based adaptation (perfect network) 81
Figure 42 Average scores for Cinematic and Metric-based adaptation (server-client degradations) 82
Figure 43 Average scores for Metric-based and Full adaptation (recorder-server degradations) 83
Figure 44 Extended AV stream Architectures Overview . 88
Figure 45 Delay distribution of generated frames 92
Figure 46 Buffering duration to `Binit` value 93
Figure 47 Percentage of dropped (to consumed) frames per `Binit` value 94
Figure 48 Percentage of dropped (to consumed) frames per `Binit`, for various delays 96
Figure 49 Overview of buffer-based quality adaptation algorithm 122
Figure 50 RoI identification (a) visualization of recorders (b) histogram of cameras orientation (from [30]) 124
Figure 51 Buffer visualization with received (gray) and missing (white) frames 130

LIST OF TABLES

Table 1	Sensors commonly employed in rehabilitation	15
Table 2	Parameter List	31
Table 3	Parameter Map	31
Table 4	Simulation characteristics	77
Table 5	Evaluation statements and target QoE aspect	79
Table 6	Mean statement scores per generated videos (simulated runs)	80

ACRONYMS

AV Audiovisual

BCI Brain-Computer Interface

BSN Body Sensor Network

DASH Dynamic Adaptive Streaming over HTTP

DAC Digital-to-Analog Converter

DAW Digital Audio Workstation

DSP Digital Signal Processing

EEG Electroencephalography

EXIF Exchangeable Image File Format

FoV Field-of-View

fps Frames Per Second

GPS Geographic Positioning System

HAS HTTP Adaptive Streaming

IR Infra-red

MDU Media Data Unit

NR Non-Reference

NTP Network Time Protocol

QoE Quality of Experience

QoS Quality of Service

PoI Point of Interest

PTP Precision Time Protocol

RoI Region of Interest

RTP Real-time Transport Protocol

SMIL Synchronized Multimedia Integration Language

UGC User-Generated Content

UI User Interface

UX User Experience

VQA Video Quality Assessment

WSN Wireless Sensor Network

XML Extensible Markup Language

Part I

TIMED METADATA & VIDEO APPLICATIONS

This part is an introduction to video applications and timed metadata. We introduce the concept of *Extended Audiovisual Streams*. We also define the challenges of such systems and overview which of these challenges we address. Then, we present our first contributions, that are a classification method and an architecture for these systems. Part I ends with a review of the most important elements in the current State of The Art.

1 INTRODUCTION

This Thesis studies how video applications can be enhanced by using accompanying timed metadata. In this context, "Enhancing" can signify either *extending video applications* or *improving video delivery*. In the first case the timed metadata is used to add novel functionalities to media-centric applications, and in the latter to increase the performance of existing functionalities.

We define timed metadata as any non-audiovisual data recorded and/or produced that contains information concerning a specific point (or duration) in the media timeline. Timed metadata are also referred to as extra-data, or timed data (non-audiovisual) – or sensor data, if they are recorded (or generated from) information coming from sensors. The streams that contain video (with audio) and timed metadata are called *extended audiovisual (AV) streams*.

A main motivation for this Thesis is to examine the feasibility of distributing all types of timed metadata in everyday applications and their potential usefulness. Therefore we are looking to answer questions like the following:

- What kind of added value does the timed metadata offer to AV applications?
- Where and how should the timed metadata be processed?
- Are there common building blocks / techniques that can be studied and then applied to different application scenaria?

In order to enhance video applications, we begin by examining the characteristics and studying the challenges for diffusion and efficient synchronized treatment of multimedia systems that use timed metadata – i.e. Extended Audiovisual Systems. We consider different delivery technologies (broadcast, broadband, IP video, adaptive streaming), as well as different recording/processing platforms (embedded, mobile, web). Then, we propose a multimedia architecture, that can be adapted to facilitate delivery of different extended AV streams. We conclude the first part of the manuscript with a review of the state of the art for the elements that are essential for the body of our work – with the rest of the state of the art being reviewed at the respective chapter where relevant.

Then, for Part II of the Thesis, using as basis the above architecture, we examine ways to *extend the functionalities* of video applications through the use of timed metadata. We demonstrate two multimedia applications built with their main features enabled by using the timed metadata.

For the third part of the Thesis, we introduce methods that *improve the performance* of extended AV applications. We examine how extra-data collected and generated can help optimize existing applications like multi-view video and then, how to improve delivery of extended AV streams. The manuscript closes with the final conclusions and list of deliverables from this Thesis.

The overview of extended AV systems and the relevant delivery challenges are examined further in this chapter. Then, in order to facilitate the relevant system requirements and specifications, we propose a classification method for applications utilizing timed metadata, which is detailed prior to presenting our architecture design for such applications further in this chapter. Following in Part I - Chapter 2 is a platform and literature overview of the state of the art, to compare with the approaches and tools that we use; the specific state of the art for each aspect is examined in the respective chapters.

In Part II - Chapter 3, we examine the scenario of extending the capabilities of interactive multimedia applications by using the timed metadata for input and control, with an accompanying example application. Afterwards, in Part II - Chapter 4, we examine using the available metadata as navigational modalities - as opposed to control parameters of the previous chapter.

From the experiences obtained, we developed methods to use sensor data (recorded timed metadata) and stream information (generated timed metadata) for facilitating delivery of the extended AV streams by adapting to the underlying data and recording information, in Chapter 5 of Part III. We also propose buffering techniques and synchronization policies, that can be applied to real-life applications (Part III - Chapter 6).

Finally, we conclude this work by reflecting on the conclusions drawn from our work, discussing on the potential impact of our proposals and presenting our plans for future work (Chapter 7). The manuscript closes with a final overview of the deliverables from this Thesis (Chapter 8), the bibliography used for this work and an appendix containing accompanying material.

1.1 VIDEO STREAMS WITH TIMED METADATA

Over the past years, there has been a decrease in cost and size of digital cameras, while at the same time the image and audio quality of the captured videos has increased. These evolutions had a direct effect in two directions. First, their embedding in various "objects" - from mobile phones and tablets, to drones and cars - that carry several other sensors (GPS Receiver, Motion Sensors, Light Sensors etc). Due to the ubiquity of such devices, there is also an increasing interest in platforms collecting, processing, distributing and rendering User-Generated Content (UGC). Secondly, devices using novel types

of sensors (Kinect, MindWave, Leap Motion, etc.) were introduced to the commercial market, that either feature a camera, or are frequently used jointly with one.

Additionally to the introduction of new data types and the embedding of sensors on more types of devices, the approach of handling these metadata has evolved. For example, the Exchangeable Image File Format (EXIF) introduced capabilities to accompany images (JPEG or TIFF) or audio (WAV) with location information since the 1990s. However, this geospatial information was often manually added in a later stage or omitted altogether. Allowing automated and continuous logging of geospatial data in photos and videos is a relatively new practice. We observe a paradigm shift from the sporadic use of minimal, loosely timed and of dubious accuracy metadata *samples*, to ubiquitous, automated, accurate and synchronous production/consumption schemes for metadata *streams*.

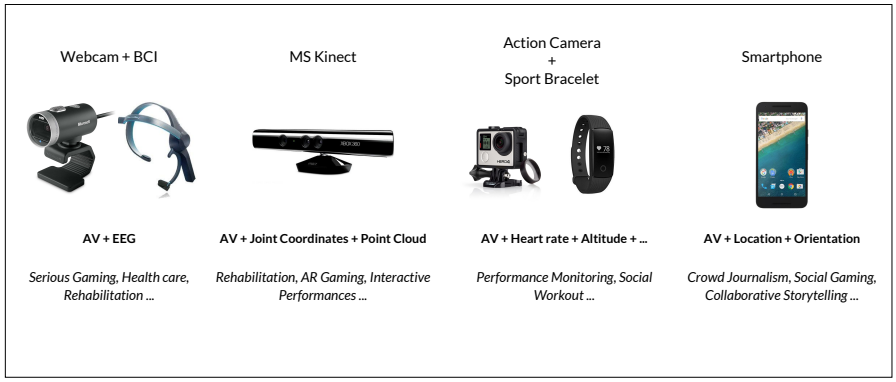


Figure 1: Examples of extended AV streams systems

We focus on these systems, based on the aforementioned devices, that produce *extended AV streams*, where the AV content is accompanied by timed metadata. *In this work, as extra-data, we define every non-Audiovisual timed data recorded/processed.* Since all the systems we study are video-based, we refer to an elementary timed element of a stream (either AV or extra-data) as "Frame" or "Sample" [1]. Some examples of systems that produce extended AV streams (as shown in Figure 1) are:

- **Kinect** (AV + Depth): Kinect is a device that is able to record audio, video and depth information (Point Cloud) of the scene and track the spatial coordinates of the user.
- **Smartphone** (AV + Location + Orientation): Most smartphones are equipped with a camera/microphone combo for recording AV, as well as GPS (for location) and magnetic field / acceleration sensors (for orientation).

1 Also mentioned in literature as Media Data Units (MDUs), Access Units, Information Units, Stream Granules et.al.

- **Brain-Computer Interfaces (BCI)** (AV + Brainwaves), **Sport Sensors** (AV + Altitude + Speed + ...), etc.

1.2 CHARACTERISTICS AND CHALLENGES

Each system (as in Figure 1) poses different aspects of similar problems. For example, the timing information between the AV and extra-data frames in the Kinect scenario is handled by the device, while on the BCI it might have to be a product of post-processing. In this case, even though we do not study how the timing information between the streams is achieved (e.g. for a setup of BCI for emotion recognition [102] it is achieved by using camera and sensor triggers [64]), we do study how it is preserved and utilized during distribution and at the receiving end.

To generalize, the extended AV streams, due to their diverse modalities, may differ from the standard AV streams in several characteristics, notably: **Number of Streams**, **Frame Rate**, **Frame Rate Behaviour** and **Frame Size**. For instance, in most of the cases of typical AV stream delivery, it concerns one audio stream, one video stream and maybe one timed-text stream (subtitles). Even if there are several audio streams available (e.g. multiple audio languages) and several video streams, only one of each is consumed at a time. A second video stream might be present in a Picture-in-Picture setup, but it is usually[2] irrelevant (in terms of content and requirements) from the others. So, for the characteristics mentioned, we have a fixed number of streams consumed (audio, video, subtitles), at a fixed frame rate (specified by the audio and video encoder - for video typically from 25 to 60 fps), that has a predictable behaviour (the frame rate in most cases is constant) and a bounded frame size (average bitrate is set up during the encoder initialization process).

On the other hand, when studying extended AV streams, all of the aforementioned characteristics might vary significantly between different scenarios. Just for the example of Body Sensor Networks (BSNs) [22], dozens of streams might be present, with sampling rates varying from a few samples per hour to several samples per second, of frame sizes spanning orders of magnitude, and stream throughput varying from 38 kbps to 720 kbps. Because of this uncertainty, our first contributions is to list the main challenges posed from extended AV streams systems, and later (in Section 1.3) to propose a classification method for such systems. The identification of the challenges and the classification of the streams is essential, in order to map our further contributions to the respective systems they address. To facilitate the organization of the identified challenges, we separate them in two categories: *Streaming* and *Presentation* challenges.

2 Unless the rendered accompanying video stream is a preview of an alternative view

1.2.1 *Streaming Challenges*

The *Streaming* part of a system, refers to the process right after the recording of each frame and includes the steps until that frame (or its product) is ready for consumption on the client (i.e. after it is decoded). More specifically, we consider streaming challenges on the domains of: *Synchronization*, *Delays*, *Communication Channels and Protocols*, *Packaging* and *Number of Sources/Producers*.

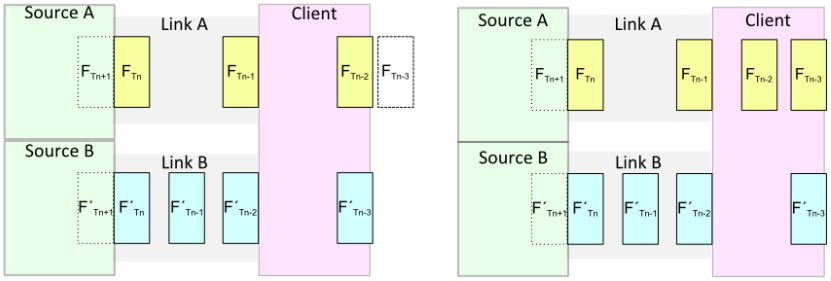
Synchronization (producer-side) deals with the common notion of time attached to each stream. In order to achieve synchronization, each frame is assigned a timestamp - indicating the time it was recorded, and/or a frame-number - indicating the order in which it was recorded, in respect to the other frames of the same stream. Even though the specifics of generating the synchronization information on the production-end are not part of this Thesis, how synchronization is achieved on playback and how the relevant information is used on the client side is an essential aspect of our work.

Delay refers to the time each frame takes from recording, until it reaches the client. The most obvious source of delay is the transmission time from the source to destination, however it is not the only one, since significant delay can be caused either from application-level design (e.g. normalizing values over time, prior to emission), or medium access protocol selection (e.g. `IEEE 802.15.4` slotted low-energy wireless [93]), or bad network conditions (forcing retransmissions). Regarding the delay, in this work our main contributions are proposals on how to deal with large or fluctuating delays, especially in live, or real-time scenarios.

Communication Channel(s), Protocols and Standards selection is another paramount design decision in such systems. The designs we propose are protocol-agnostic, but there are cases for which our proposals apply to specific protocols/standards, or that selecting the appropriate protocol/standard improves performance; in such cases we make clear recommendations. For example, in on-demand and live AV content transportation, a predominant delivery method, used by popular content providers, such as YouTube and Netflix is adaptive streaming over HTTP on TCP (e.g. MPEG-DASH [101]). However, for low-latency live systems Real-time Transport Protocol (RTP) on UDP [97] is commonly used. While both methods offer Quality of Service (QoS) *features* (i.e. timestamps and sequence numbers, for application-level QoS provisions), only the former offers QoS *guarantees* (e.g. no packet losses, in-order delivery etc.), at least until the application level (there the client can make decisions that affect these guarantees, by emptying its buffers for example).

Number of Producers is a scalability concern that should be taken in consideration when addressing all of the aforementioned challenges. For example, on the *synchronization* aspects of the system, if

a single producer is the source of the extended AV stream, studying synchronization between frames (intra-stream) and between streams (inter-stream) will suffice. If however, multiple users provide content, synchronization between the different sources (inter-bundle synchronization), might be required. The sources might have synchronized clocks (e.g. by using NTP [74]), otherwise, if the inter-bundle synchronization occurs on the server without any accurate timing information, content-based techniques must be used and the introduced delay can be in the order of minutes [20]. This is a grave scalability concern, especially in User-Generated Content (UGC) systems, for which multiple users provide content at overlapping timelines.



(a) Low-latency playback (b) Synchronous playback

Figure 2: Illustration of different content consumption approaches

1.2.2 *Presentation Challenges*

The *presentation* part of the system refers to the treatment of the frames after they are available at the client, until they are used to render the output. This area is close to the User Experience (UX) aspects of each respective application, since it facilitates novel rendering possibilities (i.e. more options on consuming the content). On this end, we are focusing on two areas: *Presentation Timing* and *Presentation Medium*.

Presentation timing (a.k.a. client-side synchronization) concerns the way the frames of a stream are consumed in a timely manner. Maintaining the temporal relationship between the frames of one stream is called *intra-stream* synchronization, while maintaining the temporal relationship between the streams is called *inter-stream* synchronization [16]. Assuming a wildlife monitoring example, where the cameras are on a central server (or directly connected to it) and temperature sensors are deployed throughout a forest, connected in a low-power wireless network with energy harvesting nodes, the temperature data might arrive several minutes late [110]. In case an event occurs (e.g. a fire), the consumer should be able to see it on the video output as soon as possible, without waiting for the temperature data

to arrive. However, when having on-demand or time-shifted playback (e.g. to investigate the spread of the fire), the client must render the video streams synchronized with the temperature measurement streams. Figure 2 illustrates such an example, with two sources (A and B) that produce frames (F and F′ respectively). For the sake of simplicity, we assume both sources have the same fixed frame duration (T) and frame generation rate ($\frac{1}{T}$). Source A sends the frames to the client via Link A, and the delay from production of the frame until arrival is the same as the duration of a frame. Source B sends the frames to the client via Link B, that has double that delay – i.e. twice the duration of a frame. Thus, at time T_n, Source A just produced frame F_{Tn} and Source B just produced frame F'_{Tn}. Meanwhile, the client is receiving frame F_{Tn-1} from Source A and F'_{Tn-2} from Source B. In the low-latency playback mode (illustrated in Figure 2a), that frames are rendered as they arrive, the client will be rendering F_{Tn-2} and F'_{Tn-3}, while in the synchronous playback mode (illustrated in Figure 2b) F_{Tn-3} and F'_{Tn-3}. The problem in such scenarios is to be able to easily switch from the low-latency, to the synchronized mode, and limit the asynchronies in the latter.

Presentation medium is relevant to the possible data processing required to fit the rendering environment. In our work, we are focusing on technologies applicable to browsers, that should be OS and apparatus independent. We try to achieve that by adhering to the standards produced by the W3C consortium[3], however each vendor has different levels of conformity to these standards, despite the fact that all major vendors are involved in the standardization process.

1.2.3 *Approach on The Challenges*

To sum up, our study is influenced by current and future use-cases for systems based on extended AV streams, like UGC-based Crowd Journalism for example. Many of these systems have evolved from existing purely video-centric designs, that had limited variety and number of timed meta-data streams (e.g. subtitles); in contrast, their new incarnations, introduce several streams and modalities. Also, the accompanying data are often user (e.g. via crowd-sourcing platforms), or automatically (e.g. via on-field sensor deployment) generated, thus provided in unpredictable and/or bursty manner.

We are considering the challenges rising from the aforementioned evolutions, and we use the timed metadata to enhance relevant applications. Our work should be able to integrate to the current ecosystem, so we will not develop new approaches on problems that already have established solutions. Instead, after evaluating industry and academia methods, we will select the most suitable and use it on our designs. For example, we have studied the impact of end-to-end

3 https://www.w3.org/Consortium/

delays, on the client (Chapter 6); in order to run relevant experiments, a system requirement is the synchronization between the server and the client clock. For the purpose of clock synchronization, we can use Network Time Protocol (NTP) [74] that is already widely used for such purposes.

1.3 CLASSIFICATION OF EXTENDED AV STREAMS SYSTEMS

To categorize the different possible scenarios we created a classification method for applications handling extended AV streams. This classification is based on the requirements and the underlying apparatus of the targeting systems. It can be used to describe systems on which our contributions (or other contributions on extended AV streams) can be applied. The classification uses the criteria of **Device Type**, **Content Availability** and **Synchronization Level**, as following:

- **Device Type**: Common, Specialized, Network
- **Content Availability**: On-demand, Live, Real-Time
- **Synchronization Level**: Loose, Strict, Critical

First, according to the *Device Type*, we can have *common* video-recording devices (such as cameras and mobile phones) that also support other data; *specialized* (such as depth cameras and BCIs) that support video-recording or are used in conjunction with cameras; or several connected devices forming a *network* - usually supporting multiple modalities (Body Sensor Networks, or e-Health Points for example). The technical incentive for these classes occurs from the number of potential different streams per class, as well as differences in modalities and their data rates.

Common devices typically have a limited number of accompanying sensors (e.g. geospatial, luminosity etc.), that provide small size extra-data (usually $<$ 1 kB per frame) at relatively low (1 to 20 Hz) sampling rates [94].

Specialized devices have a varying number of sensors (e.g. 1 IR receiver for depth-sensing in Kinect, 16-32 electrodes for a typical EEG cap used for BCI), with a device-specific frame size ($\sim$ 300 kB for a depth frame, 2 B per electrode per measurement), at diverse sampling rates ($\sim$ 30 Hz for Kinect, $256-512$ Hz for a basic EEG cap). However, even the aforementioned examples are indicative, because specialized device specifications vary significantly according to the intended application, e.g. a professional depth sensor (like Eva by Artec [9], used for 3D scanning) can have frame size of $3-4$ MB, that is $\times 10$ the size of Kinect-produced frames [72]. Similarly, in the BCI case, the number of sensors can reach up to 128 electrodes, at a sampling rate of 20 kHz for clinical research equipment [109]. The variance of characteristics according to the device type and the

targeting application is the reason that we consider these devices as a separate class.

The Network category differs from the Common and Specialized, because as the number of devices grows in the network case, so does the provision required for inter and intra stream synchronization, which are usually handled in firmware or middleware on the single-device cases. This scalability synchronization issue occurs because various synchronization algorithms rely on bounded delays - that change as streams and network overhead is added, or global clocks that have to be in sync [56]. For example, most low-energy wireless sensor networks (WSNs), utilize allocation of discrete time windows when data exchange occurs; as such, the frames arrive in bursts at the servers, and even within the same network, frames from different sensors can have varying delays, that can vary from tens of milliseconds, to a few minutes, according to the maximum number of hops (a.k.a. depth, or node level) of the network [110]. For the same reasons, when a device joins or leaves the network, the event might affect significantly the network characteristics either in the long term (e.g. if it changes the depth of the network), or the short-term (until the communication protocol reconfigures) [93].

Regarding the *Content Availability* delay requirements, we consider *on-demand* delivery, where the content is available after the recording is concluded, *live* scenarios when the content is being transmitted during the recording process, and a *real-time* case in which the recording-to-display delay is as short as possible.

Finally, on *Synchronization*, we do not differentiate inter-media synchronization, which is the synchronization of streams of different modalities, from inter-stream synchronization, which is between AV and the different streams. We identify *loose* (asynchronies over 500 ms), *strict* (asynchronies between 500 ms and 100 ms) and *critical* (asynchronies below 100 ms) classes. The classification system for the synchronization requirements is based on a combination of system requirements and human perception. The loose class is created with the main purpose of accommodating low-latency scenarios, that the imminent rendering of the frames precedes in priority the synchronization accuracy (e.g. environmental monitoring). Strict is a synchronization class that most state of the art synchronization techniques can support [50] and aligns with the "medium" synchronization category in literature [76] [28]. This synchronization level is acceptable when different streams are rendered in different devices (accompanying screens, smartwatches etc.) [34], and/or have secondary roles (e.g notifications, actuators) [42]. However, for the cases that all of the rendered streams are essential (e.g. game control, avatar overlays, virtual instruments), critical synchronization is required, in order for the QoE not to be affected by asynchronies [14].

In this section we present the architecture outline proposed for streaming and presentation of video systems with timed metadata. This architecture serves the purpose of a general guideline to identify the distinct parts of a system that form the delivery chain. It can (and should) be modified according to the specifications of the respective scenario, however the modifications and the tasks assigned to each building block must be thoroughly specified.

To begin discussing architectures for video streams with timed metadata, first we have to overview architectures for typical audio-visual systems. In Figure 3 we can see the basic components of an example system. First, the video is captured and encoded on the device, then it is transmitted to the server (over a local or remote network connection, depending on the scenario). Then, on the server side, it might be transcoded and re-packaged, before being transmitted to the client, where it is decoded and rendered.

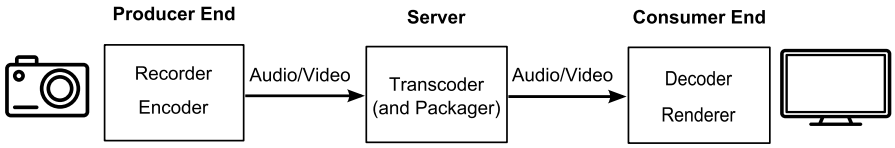


Figure 3: Outline of AV system architecture

In the simplest case of that example (e.g. security camera), all of the connections are over a local network and there is no transcoding on the server side. In a more complex scenario (e.g. using an online streaming platform, like Vimeo[4]), all of the connections are either broadband or mobile and transcoding is performed on the server (potentially several times, with different configurations).

In contrast, when timed metadata are introduced, the complexity and the possible configurations of the system increase. To demonstrate, we illustrate an architecture of a system for delivering extended AV streams in Figure 4.

On the producer end, where the data capturing occurs, there is an example for each of the three aforementioned device types. The synchronization information (timestamps and/or frame sequence numbers), which is used for intra-stream synchronization, should be attached there. This decision is based on the fact, that even though every device type differs in terms of data rates, number of streams, etc. the timing information should be present, to be used in local applications. For example, in a sensor network, timing information is essential for the packets exchange mechanism, because the nodes in the network should have a common notion of time in order to synchronize their packet exchange timeslots [93] [38]. The Recorder/Pre-

4 https://vimeo.com

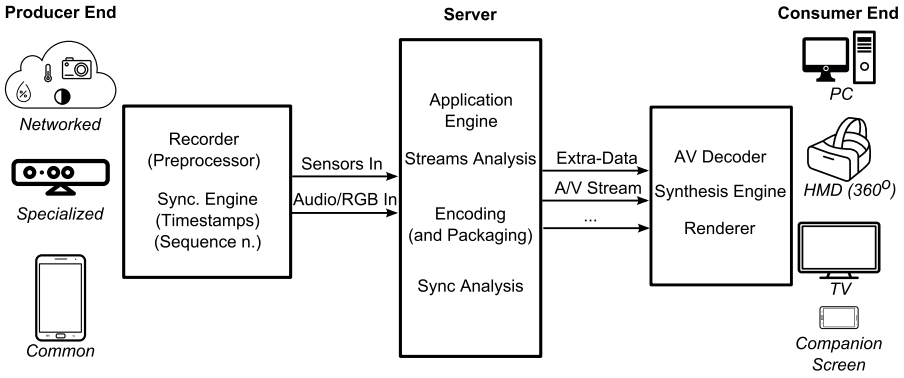


Figure 4: Outline of extended AV system architecture

processor step, if needed, can either be on the device or directly on the server, and handles any data formatting required.

The server part is responsible for gathering the recorded video and extra-data frames, applying any processing, if needed, and sending them to the client, which in turn is responsible for the playback of the incoming streams (or their processing result). An example outline of a server, with its distinct parts visible, as described in our architecture, is shown in Figure 5.

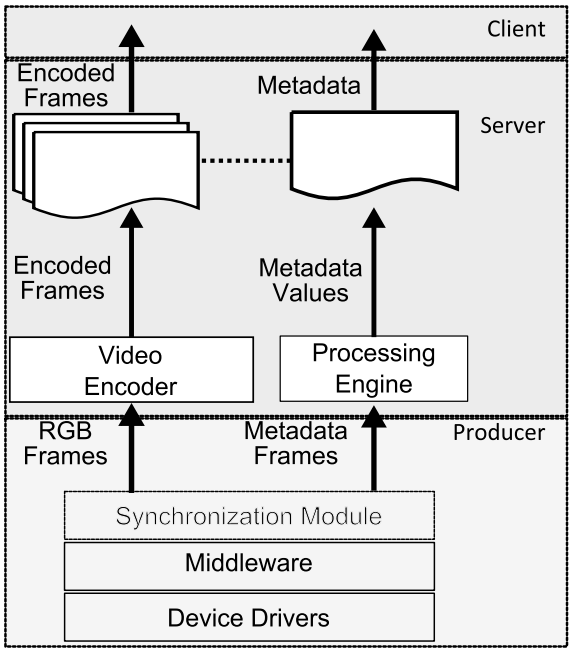


Figure 5: Schema of extended AV server

In the illustrated example we consider a single capture device (e.g. Kinect) and a single producer, thus we utilize the synchronization information from the middleware to handle inter-stream synchronization. In the case of multiple sources, an extra server-side synchronization layer is required, responsible for inter-bundle synchroniza-

tion between the bundles received from the producers. Then, the captured video frames are encoded, while the timed metadata might be subject to a similar processing. Afterwards, they are prepared for distribution; in the illustrated case, for HTTP streaming, by referencing the encoded video frames, the metadata frames and the processed metadata values inside a playlist, (and) transmitted to the client. On the client side, the received streams are decoded and because of the synchronization information provided, we can achieve coordination between the resulting AV and metadata frames during playback.

Depending on the underlying hardware and targeted application, different server modules must be added or altered. As example, when the distributed content is UGC, buffers must be added in order to maintain synchronization, and as a direct result, the buffer size affects the overall end-to-end delay of the system. Such an example is detailed in Chapter 5. Similarly throughout the manuscript we demonstrate the changes of the basic architecture according to the respective scenario.

2 ESSENTIAL ELEMENTS OF THE STATE OF THE ART

In this chapter we do a brief review of the current State of The Art relevant to recording, processing and distribution of timed metadata for video-centric applications. Most of the literature review is referenced in the respective chapters of the manuscript where it is most relevant. In this chapter, essential elements are gathered that are common throughout the Thesis, or works that due to the scope or time limitations of the Thesis we did not actively contribute to.

Regarding the devices that provide the timed metadata, we have use cases studied and example systems implemented for the *Specialized* (Kinect) and *Common* (Smartphone) categories. However, we have not implemented an application that uses a *Network* of sensors. Even though we did not create a system to test, when designing the architecture, and exposing techniques (e.g. the buffering scheme in Chapter 6), we took in consideration characteristics (e.g. data format, delays etc.) of different such networks, like Wireless Sensor Networks (WSNs) for Smart Cities [123], or Body Sensor Networks (BSNs) for rehabilitation [45], etc.

Particularly, with regards to the diversity of sensor networks, we can have varying characteristics, depending on the targeting applications and underlying infrastructure. Table 1 shows the characteristics of some indicative sensors, used jointly with cameras in rehabilitation applications [45]. We can observe that such sensors have predominately high data rates, that can be supported only because in most cases these sensors are directly connected to the gateway/basestation (in which case we classify them as Specialized devices), or because they are interconnected via a local area network - using high-throughput / low-latency connection protocols like the `IEEE 802.11` suite [12].

Sensor Type	Data rate	Bandwidth	Sampling	Data
Temperature	Very Low	120 bps	0 - 1 Hz	8 bits
Acc	High	35 kbps	0 - 500 Hz	12 bits
Gyro	High	35 kbps	0 - 500 Hz	12 bits
ECG (12 leads)	High	288 kbps	100 - 1000 Hz	12 bits
ECG (6 leads)	High	71 kbps	100 - 500 Hz	12 bits
EEG (12 leads)	High	43 kbps	0 - 150 Hz	12 bits
EMG	Very High	320 kbps	0 - 10000 Hz	16 bits

Table 1: Sensors commonly employed in rehabilitation

Service	Network type(s)	Traffic rate	Tolerable delay	Energy source
Structural health	802.15.4; WiFi and Ethernet	1 pkt every 10 min per device	30 min for data; 10 s for alarms	Mostly battery powered
Waste management	WiFi; 3G and 4G	1 pkt every hour per device	30 min for data	Battery powered or energy harvesters
Air quality monitoring	802.15.4; Bluetooth and WiFi	1 pkt every 30 min per device	5 min for data	Photovoltaic panels for each device
Noise monitoring	802.15.4 and Ethernet	1 pkt every 10 min per device	5 min for data; 10 s for alarms	Battery powered or energy harvesters
Traffic congestion	802.15.4; Bluetooth and WiFi; Ethernet	1 pkt every 10 min per device	5 min for data	Battery powered or energy harvesters
City energy consumption	PLC and Ethernet	1 pkt every 10 min per device	5 min for data; tighter requirements for control	Mains powered
Smart parking	802.15.4 and Ethernet	On demand	1 min	Energy harvester
Smart lighting	802.15.4; WiFi and Ethernet	On demand	1 min	Mains powered
Automation and salubrity of public buildings	802.15.4; WiFi and Ethernet	1 pkt every 10 min for remote monitoring; 1 pck every 30" for in-loco control	5 min for remote monitoring, few seconds for in-loco control	Mains powered and battery powered

Figure 6: Services specification for the Padova Smart City project [123]

These capabilities are permitted due to the close proximity of the sensors to the gateway. Figure 6 shows the characteristics of sensors used for smart cities monitoring [123]. We can observe orders of magnitude lower data and sample rates, comparing to the BSNs. The reason for this is that the city-wide distribution of the sensors renders the maintenance (e.g. changing batteries, or updating firmware) difficult, therefore most implementations use low-energy protocols (like ZigBee / `IEEE 802.15.4` [93]), that switch off the radio when inactive and are suitable for autonomous energy harvesting nodes [110]. Because the transmitter/receiver is most of the time "sleeping", these protocols have very high latency (from hundreds of milliseconds up to several minutes) and low throughput.

	Visual motion tracking	WSN based solutions
Cost	High	Low
Accuracy	High	Good
Complexity	High	Low
Automation	Moderate	High
Feedback	High	Moderate
Mobility	Low	High
Comfort	High	Good
Multi modality	NA	High

Figure 7: Comparison between visual motion tracking and WSN based solution for rehabilitation supervision

Nonetheless, timed metadata sensor networks are often used with cameras, e.g. for traffic monitoring in smart cities, or motion tracking in BSNs for rehabilitation. Regarding the latter example, sensors offer different advantages over cameras, as shown in Figure 7. Our relevant contributions on the domain can be applied to build hybrid systems that include both sensors and cameras.

Similarly, for the *Common* devices category, we studied UGC scenarios with mobile phones, that have a camera and geospatial sensors (GPS, accelerometer etc.). In the studied cases, it is a requirement to be aware of several parameters, that we assume are known or calculated. When several users are filming, the sensors of their smartphone can be used to identify Region of Interest (RoI) [31]. This work proposes a consensus system, based on the orientation of the devices and an assumed viewing angle (of 90 degrees). In the same paper, the authors propose monitoring changes in the behaviour of the users (i.e. panning or zooming to a subregion the RoI) to identify specific events - e.g. a guitar solo at some point during a concert.

To increase precision, if the characteristics of the camera are available, the camera Field-of-View can be estimated (i.e. Viewable Scene Modeling) [10], instead of approximated (as in the aforementioned method). If that is the case, indexing and querying geo-tagged videos are enabled [67].

We also had to review the literature on topics that are not directly connected to this Thesis, but were required in order for our research to be thorough. Such an example is Video Quality Assessment (VQA) algorithms, that are used in the system described in Chapter 7. VQA algorithms, fall under two categories; the reference-based, that compare the image sample with a "ideal" or highest-quality version of it and the Non-Reference (NR) algorithms that rely only on the information contained within the described sample.

MSE [113] is a widespread, though disputably outdated, technique and VMAF [63] is a relatively new, yet promising solution. However, both of the algorithms are reference-based and in our UGC scenarios we do not have any higher quality recordings of videos to use as reference (the highest quality is the one transmitted), therefore we focused on using NR algorithms.

In our implementation we opted for applying edge detection to sampled frames in order to measure blurriness [5]. We chose this method because it is one of the most lightweight NR techniques, thus is suitable to run on mobile devices, yet it performs adequately to indicate the relative image quality of the image. Among the other alternatives, BRISQUE [75] is an algorithm based on identifying "unnatural" distortions by using luminance coefficients. We did not select BRISQUE, because even though it has lower computational complexity comparing to other algorithms with similar performance, like BLINDS-II [95], it is still significantly more demanding than our choice. Also, the performance of BRISQUE is related to the training set provided, and we could not find a suitable set to our use case in order to fine-tune the algorithm.

Similar methodology was applied when addressing Synchronization challenges as identified in Section 1.2. We assume that basic source control techniques are present in all systems [16] - i.e. all

of the produced streams have attached synchronization information (source identifiers, timestamps and/or frame numbers etc.). We are using this information in the systems described in Chapters 4, 5 and to achieve inter-stream synchronization [50]. In Chapter 6 the same information is utilized for preventive and reactive receiver control.

In all of the studied aspects concerning synchronization of multiple streams, as well as those presented in the following section, a common notion of time between the actors is always useful, to sometimes essential (e.g. in companion devices). This *clock synchronization* is commonly achieved by the parties exchanging messages with a timekeeping server (for global synchronization), or between them (for local synchronization). Historically, the Network Time Protocol (NTP) [74] (maintained by IETF), was used for such applications and is still widespread due to its simplicity and the availability of NTP servers. It offers accuracy of a few milliseconds, for locally connected devices, to $\approx$ 100 ms for devices connected over the internet. The Precision Time Protocol (PTP) [37] (by IEEE), offers capabilities for hardware implementation, and can achieve sub-millisecond delta for devices connected over a local network.

When clock synchronization is not present during the recording, either because the recorders are connected to different networks, or because the inter-stream synchronization is required after the recording ends, content-based synchronization methods must be applied. Because we are focusing in sensor-enabled systems, one of the modalities can be used to transfer synchronization trigger signals (in this example, a video source [64]). However, even though this solution is lightweight and can achieve synchronization granularity starting from approximately 30 ms (depending on the number of sources), it does not scale efficiently and the authors tested it in a local setup. For a distributed setup, it would require a global synchronization system in place, at least for the initialization step, therefore it can be replaced by NTP/PTP altogether. Also, it might interfere with the actual content of the modality – e.g. by inserting artificial cues during recording.

For these reasons, often video and/or audio [20] [104] events extraction is used. With this approach, instead of inserting artificial cues, algorithms scan the recordings for parts with distinctive features (fingerprints) and use those features as cues. This approach can offer similar synchronization level, but it is computationally expensive to an extent that is excluded for any real-time or low-latency live streaming applications. If such content-based synchronization method must be used, it has to be carefully selected because video-only solutions might suffer if camera(s) are filming from "outlier" viewing angles (e.g. drone or backstage cameras); and audio-based methods can skew the synchronization level if the microphones are sparsely positioned and the sound from the source arrives with different delays

to each of them. This phenomenon is exaggerated if the sound source is also mobile, thus the delta between the sources is changing - for example, recording a tennis game with cameras in opposite sides of the court.

2.1 STANDARDIZATION EFFORTS

As mentioned previously, we take in account standards that apply to our domain. The standardization consortia that are most relevant to our work include MPEG, W3C, IETF, DVB, 3GPP, IEEE and HbbTV (among others).

Because all of our implementations target browser-based clients, elements from the suite of standards by the W3C were paramount to our work. All of our applications use the `video` element (and its respective APIs), which is defined by the HTML W3C standard [39]. We use the W3C Extensible Markup Language (XML) [18] for several types of metadata and it is also used by MPEG-DASH for its descriptor files. Another format that is utilized for timed metadata in our work is JSON [17], which is a standard maintained by IETF.

Other standards can facilitate functions like synchronization between different streams and/or multiple screens [34]. Synchronized Multimedia Integration Language (SMIL) is a standard maintained by W3C and defines a XML-based format for expressing multimedia presentations [106]. It includes specifications for temporal and spatial relationships between the media elements and actions (e.g. transitions, control etc.). With these capabilities enabled, SMIL can facilitate synchronized rendering of multi-modal media, companion screen applications etc. Specifically for companion screens, DVB has developed the Companion Screens and Streams (DVB-CSS) standard, that targets secondary devices in a interconnected TV-centric setup [33].

Because our work is focused on video platforms, we have strong interest in such standards. The preferred method used for delivering the video over the internet is HTTP Adaptive Streaming (HAS). The principle of HAS is that each video is segmented to shorter clips, that are transcoded to several bitrates which are made available at the server and the client requests the bitrate (of each segment) that better suits its available bandwidth. We are using MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [101] as reference, mainly due to its popularity, active community, extensibility and because it is an international standard (as opposed to the other alternatives, such as: HLS by Apple [7], HDS by Adobe [54] and SmoothStreaming by Microsoft [122]).

In low-latency streaming scenarios, we use the Real-time Transport Protocol (RTP) over UDP [97]. It offers capabilities that minimize the end-to-end delay, with the trade-off that it does not offer QoS

guarantees. As a result, packets might be dropped or arrive in the wrong order. In Chapter 6, we demonstrate a technique to deal with such issues.

When packaging of the video is required, we prefer to use the ISO BMFF format [52] in `mp4` containers, also standardized by MPEG [53] and, most importantly, able to carry timed metadata with their associated timing information [26].

For the rest of the manuscript whenever the choice of a specific standard/protocol is essential, we mention it and justify its use, otherwise it is safe to assume that one of the aforementioned standards was selected, or that any substitute can be used instead.

2.2 DELIVERY OF VIDEO WITH TIMED METADATA

As mentioned in the previous section, we commonly use MPEG-DASH for video streaming. The reasoning behind this choice is that the standard has capabilities of extending the delivery of other media on top of audio and video [25]. Also, it is interoperable with other standards of MPEG, like MPEG-V that is designed to handle timed metadata (sensor and actuator information/commands) and its applications vary from Internet-of-Things enabled systems [61] to building interfaces for virtual worlds [108].

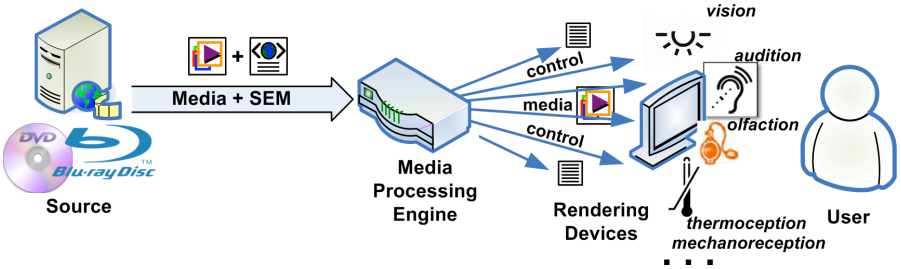


Figure 8: Concept of MPEG-V sensory effects (from [111])

We can get insights from works on sensory information (at the time focused on MPEG-V), from the perspective of both the capabilities as output accompanying video (as shown in Figure 8) and the enabled architectures. As an example of architectures for video systems with MPEG-V capabilities, Figure 9 shows the architecture of SEVino tool [111], that allows authoring of sensor effects as timed metadata.

MPEG is also working towards a way to achieve "Media Orchestration" that they define as the process of coordinating devices, media streams and resources to achieve a unified immersive experience [1]. MPEG MORE [35] is a standard developed with this goal of media orchestration. It is reusing some elements from DVB-CSS, like the

1 Definition taken from Chapter 1. of W16133 "Call for Proposals on Media Orchestration Technologies": https://mpeg.chiariglione.org/standards/exploration/media-orchestration/call-proposals-media-orchestration-technologies

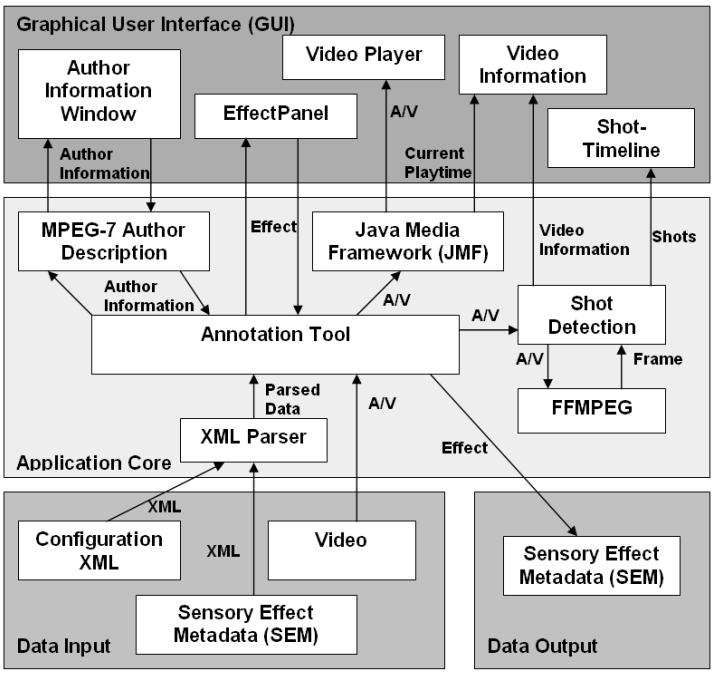
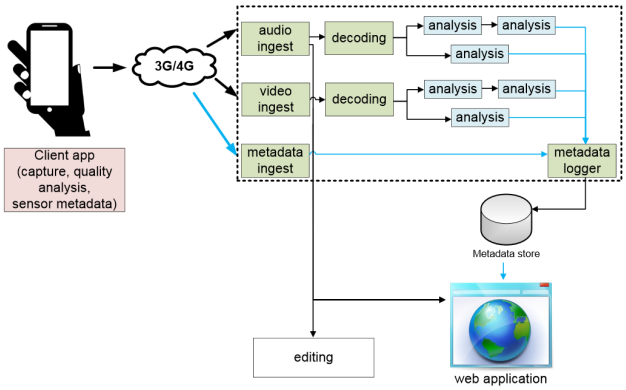


Figure 9: Architecture of the SEVino tool (from [111])

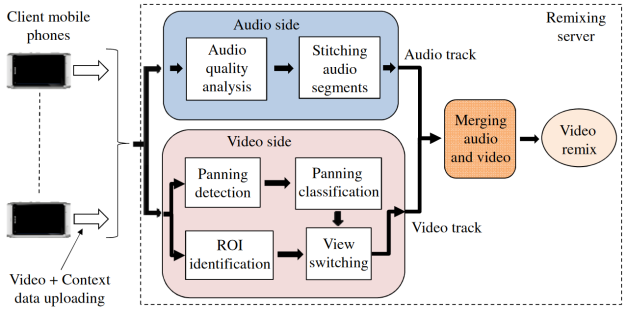
Wall Clock and Time Synchronization parts. However, it differs from DVB-CSS in two main aspects: it facilitates orchestration for non TV-centric systems, and has provision for synchronizing multiple sources (on top of the DVB CSS capabilities for multiple receiver devices).

On systems that capture and distribute videos with timed metadata, there is a debate concerning the pre-processing of the captured content. This pre-processing step can include cleaning, formatting, extracting other metadata etc. One approach is the metadata (and occasionally the AV content) to be processed locally, on the capturing device, as in Figure 10a [115], and the other is to process it remotely, at the server, as in Figure 10b [30]. The main advantage for the on-device pre-processing is that the server receives the data ready for application-specific processing and distribution, thus it has lower resource requirements for scaling the system. Otherwise, when all of the processing happens on the server, typically more processing power is available, thus more elaborate techniques can be applied. Additionally, on the server side there is a global view of the connected content producers and consumers thus allowing functions (e.g. comparative metrics between the sources) that can not be achieved without peer-to-peer communication which is throughput demanding.

A "golden rule" does not exist to guide development of applications on whether server-side or source-side processing should be used. The decision should be taken considering several factors, like the underlying hardware, targeting application and expected number of devices. For our implementations we prefer to perform computationally inexpensive processing on the source (e.g. data formatting) and mitigate



(a) Producer-end Processing (from [115])



(b) Server-side Processing (from [30])

Figure 10: Overview of data processing paradigms in literature

to the server functions that require extensive processing power, or global view of the systems (e.g. feature extraction). Throughout the manuscript, on the application examples, we mention recommendations concerning the processing approach, when necessary.

Part II

EXTENDING VIDEO APPLICATIONS

The second part of this manuscript focuses on applying the timed metadata in order to extend multimedia applications by introducing novel functionalities. We demonstrate two use cases with respective application examples; first, timed metadata as input for generating content, and second, to extend the navigational capabilities for the underlying multimedia content.

3

TIMED METADATA FOR CONTROL OF INTERACTIVE MULTIMEDIA APPLICATIONS

In this chapter we examine the first scenario of extending video applications, by using timed metadata for control of interactive applications. "Control" in this context is creating or modifying content (not to be confused with playback control). More specifically, we chose distributed multimedia synthesis control, by using Kinect as an input device. *The content producer uses the Kinect device to generate or live edit an audio signal and (using the same input) to produce relevant visualizations.* We chose the scenario of multimedia synthesis, so we can study the *Specialized* device case and, most importantly, because multimedia content authoring can be a process that yields a rigid output (as explained in the following paragraph). *We use timed metadata to extend the production of audio and video by adding flexibility over the final output.* As all of our implemented systems, the platform is designed to run in the browser.

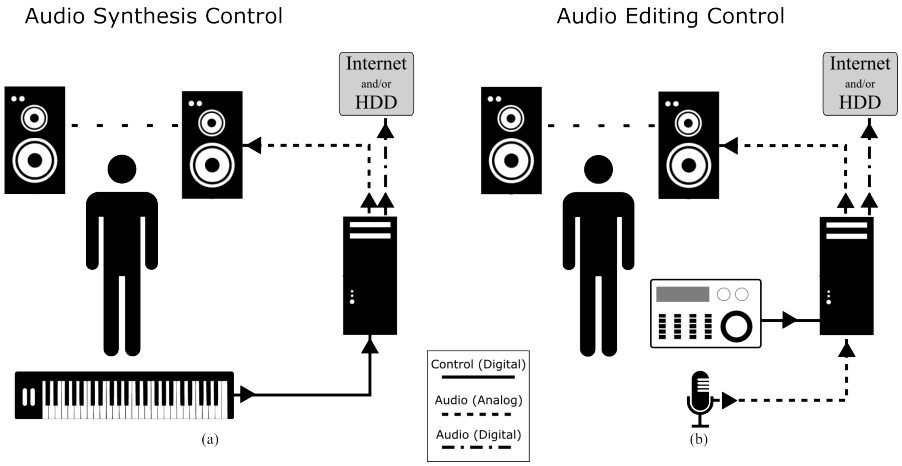


Figure 11: Producer outline for audio (a) synthesis (b) editing

For audio synthesis (Figure 11 (a)), the producer uses a controller that sends the input (as MIDI, OSC etc.) to the Digital Audio Workstation (DAW), which typically is a PC with an adequate audio interface. The DAW in order to be initialized has registered a set of "instructions" for the input. As example instructions, turning knob A at the controller alters the soundwave frequency, while turning knob B alters the amplitute etc. Then, the synthesis engine of the DAW, uses these "instructions" to transform the signals from the controller, to signals for the synthesis engine. The synthesis engine in turn creates an audio file, that is transmitted to the network and/or saved for later use. At the same time, it sends the output to a local output de-

vice (i.e. speakers or headphones), that the producer uses to monitor her performance.

For audio editing (Figure 11 (b)), again the producer uses a control device that sends the input to the DAW, which also takes as input an external audio source that can be coming from a physical device (microphone, or musical instrument), or an external source (live or pre-recorded audio track). The triggered commands from the controller are used to modify the audio from the audio source. Again, the output is rendered locally for monitoring purposes and exported as a digital audio file for distribution or later use.

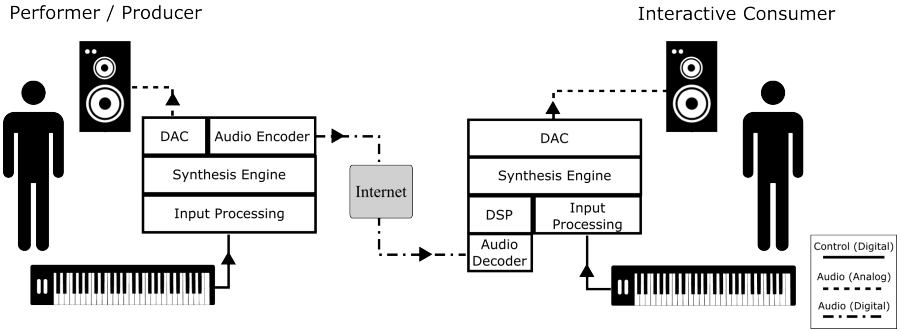


Figure 12: Audio output based producer-consumer chain

On the client, the content consumer receives the audio stream of the performance and directly renders the output. This is a rigid setup because once the original content is authored and transmitted, on the client side, there is minimal control over the final audible output. If any modification is required, computationally expensive Digital Signal Processing (DSP) techniques have to be applied to the audio signal (e.g. Fourier Transformations) as in Figure 12. Also, if audio filters have been applied on the producer side (e.g. band-pass frequency filter, or reverb effect) their result can not be reversed or modified on the consumer side, even with DSP.

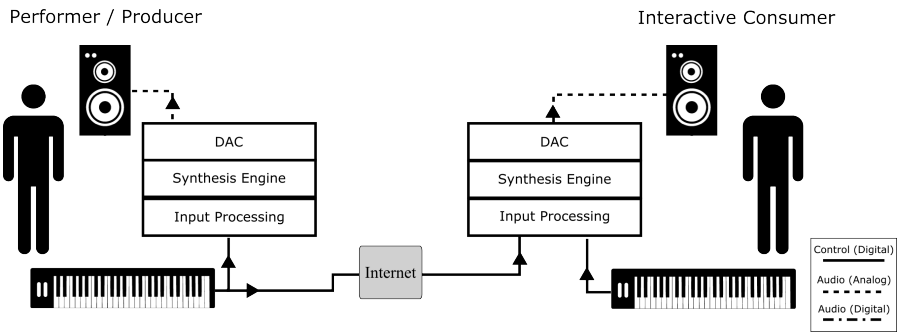


Figure 13: Controller output based producer-consumer chain

Alternatively, if modifying the output on the client side is essential, instead of transmitting the audio file, the producer can transmit the inputs (as timed metadata) to the client and generate the final out-

put there, as in Figure 13. This way, all the control parameters are exposed and the user can modify them at will. This approach however gives absolute control to the client, and can completely alter the result, to an extend that does not resemble the original works.

The same principles apply to accompanying visualization of the performance. The visuals can be a product of audio processing, video recording, input-triggered, or a combination of them. Typically they are produced on the content producer end and they are transmitted as regular video to the client - that again does not allow significant modifications without heavy processing.

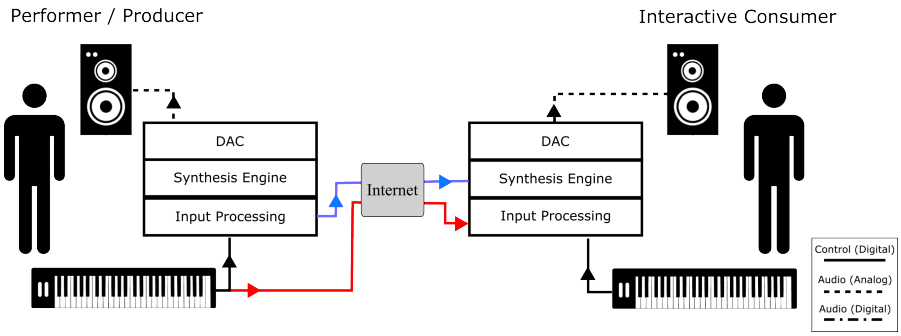


Figure 14: Controller output based producer-consumer Chain

To achieve flexibility over the final output on the receiving end, while the producer maintains the artistic control, we propose a modular architecture, with symmetric clients on both the producer and the consumer side, as in Figure 14. The direct outputs from the controller are both sent to the client and processed locally. Additionally, the output of the controller processing unit is used locally as input for the synthesis engine to produce the audio used for monitoring, *and* is also sent to the client as timed metadata (alongside the raw controller outputs), to be used for rendering the original unmodified output. At the initialization stage the content consumer has received a redacted "instruction" set from the producer. This way the content consumer is able to modify the parameters listed on the instruction set by reassigning their functionalities, while the missing parameters have to be rendered by the received timed metadata. Because the producer decides on which parameters are included in the instruction set, the creative part remains on the content creator, while at the same time, other parameters can be altered according to the preference of the content consumer, thus enhancing the overall experience.

We analyse our proposed solution by going through the design and development of an example audio application controlled with Microsoft Kinect. We chose Kinect over audio application-specific controllers, because it is a multi-modal input device, therefore permitting to demonstrate a broader spectrum of possible implementations.

In the following section we elaborate on the use cases we consider for our solution. Then, in Section 3.2 we present the state of the art

on relevant technologies for distributed audio applications, followed by the design of the application built with our solution in Section 3.3. Finally, in Section 3.4 we conclude the chapter by discussing current and future perspectives of similar multimedia applications using timed metadata.

3.1 SCENARIO DESCRIPTION

The setup we study consists of a content producer on the server side and an interactive content consumer on the client. The main content is an audio performance, while we also consider dynamic content-based accompanying visuals, as it is common in such cases. The creator uses some controller (in our example the Kinect) to produce the content, while the consumer receives it and has the ability to modify the parameters of the final output as allowed by the content producer.

For example, the pitch and the rhythm of a music piece should not be altered, but the creator might permit the user to modify the level of an effect (e.g. reverb).

On the technical challenges, the content produced by the content-author should be identical as received by each client, even if the final output is subject to the parameters set by the respective user. That means, that if the client does not alter any parameters, the final result should be the same as the one that the producer renders on her monitors. Also, since there are several resources transmitted (instead of a single audio file), there are extremely tight timing restrictions, thus synchronization must be retained.

3.2 STATE OF THE ART

The paradigm for controlling audio using Kinect is already examined in the literature, and the approaches can be categorized in those that perform direct mapping of inputs [86], and those that are based on mapping of actions (i.e. using gesture recognition) [41] [24] [79]. We excluded those based on gesture recognition because the synthesis engine must be separately trained for each user and each application. Also, some of these approaches are application-specific [24], or have latency issues [79]. Therefore, we resolved to use the direct mapping of inputs approach [86], that can be extended to also support gestures if needed.

With direct mapping, the user input values are assigned to the parameter they control in the audio or visuals synthesis pipeline. We extended this practice by adding a distributed processing feature that is based on cloning the synthesis engine to the client, thus allowing the receiving user to obtain more flexibility on the final output by

modifying the unlocked parameters. The whole process is detailed in Section 3.3.1.

Even though our work is different, the distributed processing aspect is influenced by a symmetrical paradigm used for graphics in mobile gaming applications [32] and generic remote-rendering platforms[1], to reduce the GPU workload for resource or power constrained devices. In these platforms, inputs from the user are transmitted alongside the current state of the application to a server, where the engine renders the output and sends it to the client. Our goal is flexibility, instead of power/processing constraints; thus our approach differs because the rendering happens on the client side too (on the server it is used for monitoring), but it is similar because we are also sending the inputs (and the mapping parameters) to the clients. Such frameworks are application-specific, while we propose a generic browser application architecture, in order to provide flexibility on the user end.

A relevant field to this work is Collaborative Music Creation, where various approaches [3] and platforms [44] exist. However, they are not suitable for unidirectional music performance, because they either allow full control over the final output, as every user is a potential artist, or they produce a single audio file.

On modular in-browser audio synthesis, Gibber [92] is an online platform that offers composition via coding. Even though Gibber, as a programming environment, offers great flexibility (that also extends to visual arts), it exposes the full code complexity to the user, hence requiring music and programming experience, for any modifications. Regarding live performances, its usability is limited, because the sound is produced by typing code; furthermore, its only streaming support is a collaborative mode that suffers from the same drawbacks mentioned in the previous paragraph.

Also, all collaborative platforms expose the full production system to the clients, therefore it is very difficult to reserve the rights for parts of the composition. However such platforms can be useful in cases that the artist encourages adaptation of their works (e.g. Nine Inch Nails remix project [78] and Radiohead's Reckoner/Nude mixes [2]). With our proposal we are providing the producer with the liberty of exposing only the desired parts of a project to the clients.

Regarding the AV delivery, as we mentioned in Section 2.1, HTTP Adaptive Streaming (HAS) is common for such applications. In the typical approach where the output consists of a visualizations video file with audio, this delivery method can lead to a loss of quality in constrained networks. However, in our architecture the audio and

1 VirtualGL: 3D Without Boundaries - http://www.virtualgl.org

2 https://web.archive.org/web/20120218161743/http://www.radioheadremix.com/information/

parts of the visuals are generated on the client, thus avoiding any degradation due to networking limitations.

Because in our scenario we are transmitting the raw video recording (alongside the generated visualizations), we use HAS only for that purpose. We designed our solution to be DASH-compatible, since DASH streams can also carry other data alongside the audio and video, in a synchronous manner [25].

3.3 AUDIO SYNTHESIS APPLICATION EXAMPLE

To elaborate on our paradigm, we designed a server-client application, that implements basic audio synthesis functionalities with the following specifications:

- Monophonic synthesis engine.
- Support for Stereo Panning, Some Effect (Reverb), and Waveform type.
- Record video of the performance and provide input-based visualizations.

With the aforementioned specifications, the application must comply with the following requirements:

- Provide auditory feedback of the output to the producer.
- Elementary sonic characteristics (rhythm and tonality[3]) should be locked (i.e. controlled solely by the producer).
- Control over the effects by the consumer.
- Control over the visualizations by the consumer.

The producer uses Kinect as input to set the sound frequency and effect parameters. Kinect (via its middleware and accompanying SDK[4]), tracks the position of the user and provides a set of spatial coordinates for predefined body points of interest - that, in this scenario, are part of the timed metadata utilized. The set of the spatial coordinates is called a "Skeleton" and the individual points are called *Joints* and their naming is according to their respective body part (LHand, RShoulder, Head etc.). For monitoring the output, real-time auditory feedback is provided to the content author. On the client side, the user receives the performance from within his browser, and has control over the effects and rendered visualizations. The fundamental elements of the performance, like the sound frequency, are not subject to change by the consumer.

3 the term tonality refers to the soundwave frequency, since the presence of effects affects the auditory perception of the pitch [83]

4 https://dev.windows.com/kinect

Parameter	Min. Value	Max. Value	Locked
Frequency	0 (Hz)	22000 (Hz)	YES
Volume	0 %	100 %	YES
Panning	-100 %	100 %	NO
Reverb	0 %	100 %	NO
Active	OFF	ON	YES
WaveForm	Sine	Shaw	NO

Table 2: Parameter List

Parameter	Joint	Min. (mm)	Max. (mm)
Frequency	RHand Y	Torso Y	Head Y
Volume	RHand X	RS X[4] - 300	RS X + 1000
Panning	Spine X	-1500	+1500
Reverb	LHand Y	Torso Y	Head Y
Active	RHand Z	Head Z - 20	

Table 3: Parameter Map

[4]Right Shoulder X

3.3.1 *Audio Synthesis*

For the Synthesis Engine input, we are performing direct mapping of the Kinect data, instead of gesture recognition, in order to obtain low-latency accurate control [86]. To achieve this, a *Parameter Set* is required, that consists of a *Parameter List* and a *Parameter Map*. The Parameter List, contains all the parameters of the Mapping Engine, with their respective value ranges. The Parameter List we are using for the demo application is shown in Table 2. We modified it and added a "Locked" field, which indicates whether the specific parameter, is subject to be changed by the user, or if it is considered to be essential for the performance hence used as-is.

The Parameter Map we are using for the aforementioned Parameter List is shown in Table 3, and a visualization of a skeleton with the Joints included in the Map highlighted, in Figure 15. The Parameter Map contains the Joint Coordinate we are using for each parameter of the Parameter List and its respective range. Since the Join Coordinates are in space, we can use as reference other Joints, or distance (in mm) from the Kinect.

A Parameter Set P is used to initialize the Mapping Engine on the server side, and the client uses a P' subset of P (i.e. $P' \subseteq P$), containing a Parameter Map with the parameters defined as "locked"

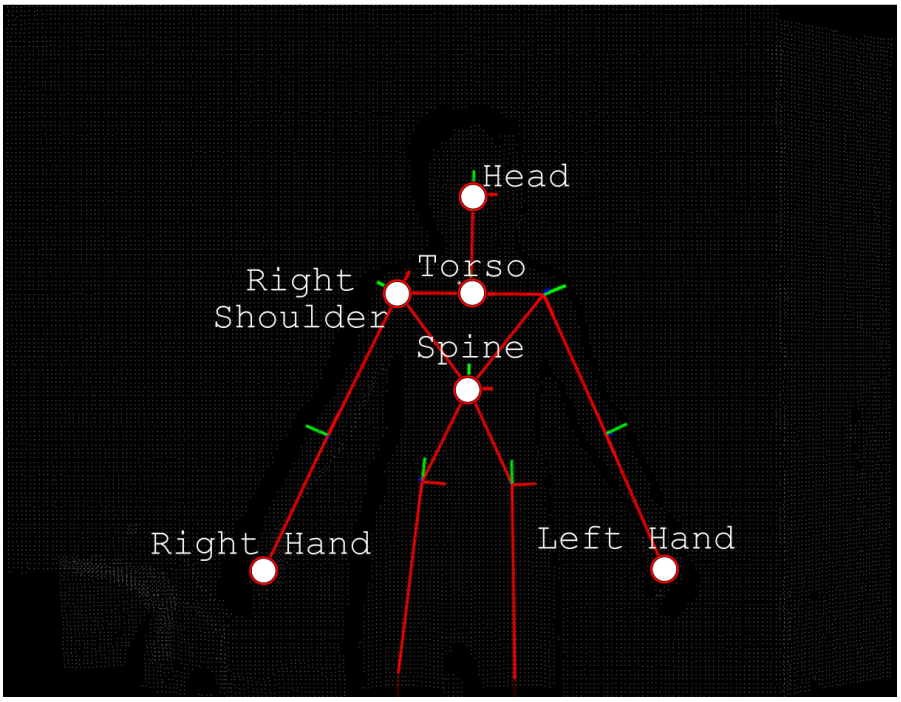


Figure 15: Skeleton with the Joints used in the application highlighted

removed. After initialization, at time t, the Mapping Engine uses Joint Coordinates K_t, with P, to provide the Parameter Value set E_t that the Synthesis Engine uses to generate auditory feedback to the producer; Figure 16 shows an overview of the process. The timed metadata consist of the coordinates K_t, along with the values E_t, and they are delivered to the client, to reproduce the sonic result and create visualizations.

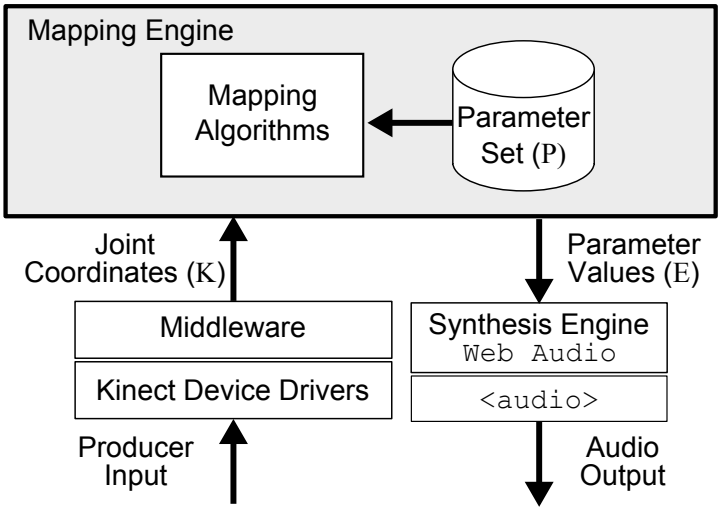


Figure 16: Audio Engine architecture

Because the server and the client use the same `Web Audio` based Synthesis Engine, the audible output is identical at both ends, al-

though if the user wants to change any of the unlocked values, P' can be used. The modification can be on the value mapping (e.g. by setting a different Min/Max), or by setting her own input (e.g. by using a controller via the `Web MIDI` API). At the same time, since P' does not contain the Parameter Map for the locked Parameters (e.g. Frequency), the client is oblivious to how the specific value from E_t is produced, hence unable to modify it.

Figure 17: Configuration panel of the Unveil reverb-removal tool

The flexibility of changing the Reverb depth, even during runtime, is a strong example of functionalities that would be computationally expensive to perform from the browser, if the client received directly the audio stream [59]. Even in a standalone DAW setup (that has sufficient processing power), to use a tool removing the reverb, as the one[5] shown in Figure 17, would require excessive technical knowledge and experimental configuration.

Regarding the technical aspects of our implementation, we are using functionalities of the `Web Audio` W3C specification [2], that is natively supported by all the major browsers. In order to achieve optimal performance for the audio synthesis, the `Web Audio` specification defines an audio routing graph, that is processed by optimized low-level code of the browser engine. We achieve further performance improvement by using the `Web Worker` API to run parts of the application in separate threads, a process described in Section 3.3.2. The specification defines the `AudioWorklet` interface and its `AudioWorkletNode`[6], for dedicated audio threads, but it is not finalized yet (nor implemented by any browser)[7], so we are not able to thoroughly test a multi-threaded audio engine.

5 UNVEIL, by Zynaptic – http://www.zynaptiq.com/unveil/

6 originally in the specification they were designed as AudioWorker and AudioWorkerNode respectively

7 as of the time that the platform was designed and implemented. Currently, only Firefox supports *some* experimental features

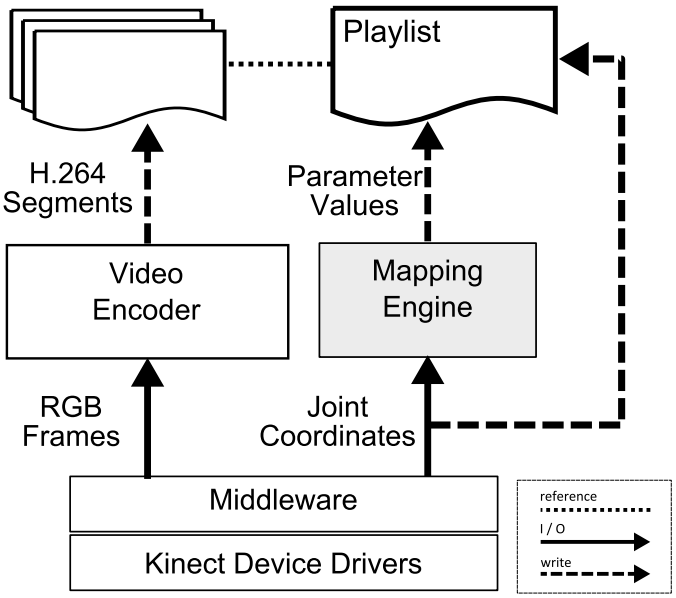


Figure 18: Application data handling during runtime

3.3.2 *Data Exchange*

Regarding the data exchange process, the only reference point required by the client application during setup, is a playlist file maintained on the server. The playlist starts with the locations of the application Parameter List and Parameter Map, that are used to initialize the Audio and Synthesis Engines. It also contains the location of the file with the initialization information regarding the codec used for the video stream from the Kinect.

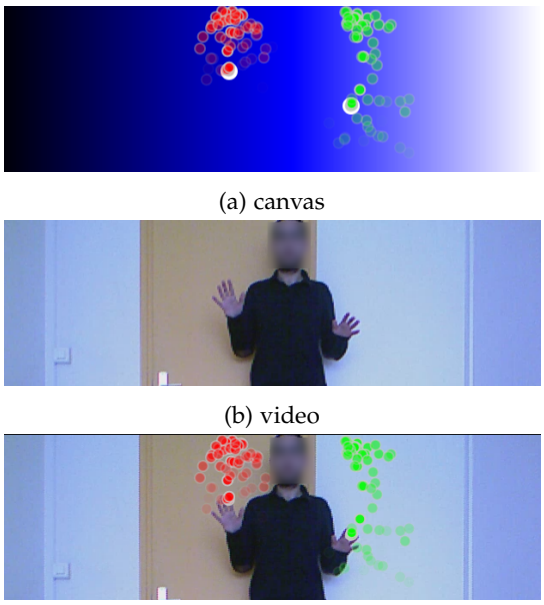
The server is responsible for updating the playlist by adding the Joint Coordinates (K_t) and Parameter Values (E_t) whenever the Mapping Engine provides new output (at approx. 30 Hz). To achieve synchronization, the timestamp of the Coordinates provided by the Kinect middleware is preserved, as is the case with the video frames timing. Note that the Coordinate frames and the video frames are not aligned (details on Section 3.4), but preserving source timing information is essential to implement techniques for counter-balancing the mis-alignment, or add filtering etc.

The recorded video is encoded in fragmented H.264 (MPEG-4 AVC) [51] and the playlist references to the resulting segments. To balance between file size and latency, we chose a segment duration of 1 s, thus a new location is added on the playlist at 1 Hz rate. Figure 18 shows an overview of the server data update mechanism. On the client side, both the playlist and the video segments are fetched from within the browser using the `XMLHttpRequest` API (AJAX).

The video segments and the timed metadata can be packaged in mp4 containers, for later offline distribution. With the timing capabilities of the mp4 file format, we are able to achieve frame-accurate

synchronization, while using a widely supported format. For consuming mp4 files from the browser, libraries that analyze and extract data such as mp4box.js[8] can be utilized.

3.3.3 *Using Timed Metadata to Generate Visualizations*



(a) canvas

(b) video

(c) mixed

Figure 19: Visualization examples

For rendering input-based visualizations, the application uses the `<canvas>` element. Vector graphics are rendered on the `canvas` by using its `CanvasRenderingContext2D` API. Figure 19a shows an example content-based visualization, with gradient based on the current Stereo Panning, pointers on the projected coordinates of the performer's hands and dynamically generated sprites according to the sound frequency. More specifically, we create a canvas-wide black-to-white gradient; then, the Audio Panning min/max values [-100%, 100%] are mapped to a blue gradient "stop", with possible values [0, `canvas.width`], in the illustrated example the Audio Panning value is close to 0% - i.e. we have Stereo: left and right audio output has similar volume - which reflects to the blue gradient being centered to the canvas. Similarly, the current Audio Frequency [0, 22 kHz] is mapped to the sprite generation frequency [10, 30 Hz].

Alternatively, for rendering the video of the performance, the `<video>` element is used instead. Since we have a continuous video stream, the frames are buffered and loaded using the `MediaSource` API. Figure 19b shows an application screenshot displaying the performance video, at the same time as the previous figure.

8 https://github.com/gpac/mp4box.js/

Finally, since the timed metadata stream is synchronized with the video stream, it is possible to combine the `<canvas>` and `<video>` elements to create composite customizable visualizations, as the one in Figure 19c. We mention more details on the synchronization level in the following section.

3.4 SYSTEM PERFORMANCE

When we designed the platform we were aware from various studies [114] [112] that there is a latency between the video frames and the corresponding joint coordinates. The source of this delta is the processing time of the point cloud and video frames, required by the middleware, to calculate the joint coordinates. Depending on the study, the average measured latency for tracking one skeleton is in the region of 30 to 106 ms, with observed maximum at 156 ms (which can reach up to 490 ms when tracking two skeletons) [65].

Also, the average skeleton data acquisition frequency varies in the range of 18 Hz, for two skeletons from three tracked users, to 30 Hz for one skeleton from one immobile user tracking. Because the values are application and hardware-dependent we did an evaluation for ourselves.

Using a first generation Kinect connected to a PC with 4-core CPU at 3.60 GHz and 16 GB of RAM, we measured the average acquisition rate to be at 27 Hz, and the lowest observed at 22 Hz. Note that these values exclude the first and last few seconds of capturing, where there is very abrupt behaviour on the data due to sensor calibration.

The latency mentioned in the single skeleton scenario is borderline acceptable for live applications, but to compensate for spatial jitter and increase precision of the coordinates, a filter might be needed. Adding a filter for higher accuracy on the joint coordinates would add an extra latency of about 2 frames [36]. Considering that those 2 frames, for a capture rate of 25 Hz would add an extra 80 ms, would disqualify Kinect for live audiovisual control.

For our application, we propose using unfiltered values on the production end to keep the latency within acceptable limits for control applications [59] [14] (i.e. classified as "Critical"), and using the filtering on the values sent to the consumers (i.e. "Strict" synchronization) where we can compensate for the difference, as detailed later in Chapter 6.

On the client, originally we tested adding the Joint Coordinates and Parameter Values as WebVTT cues and processing them in a timely fashion using the `oncuechange` event of the `textTrack` element. However the delay between the timestamp of the cue and the time it is actually fired can reach up to 250 ms. Thus, we utilize the `Web Worker` API, to run a background thread, dedicated to fetching and parsing the Joint Coordinates and Parameter Values.

Concerning bandwidth requirements of our implementation, the transmitted data for a minute of audio content is less than 200 kB in size, while an audio file of the same duration, even in a lossy format such as the *mp3* can be more than 2 MB (for a bitrate of 320 kbps).

Similarly, for the visualizations, we transmit only the timed metadata and the video recording, that is approximately 35 MB per minute. In the typical distribution chain, to support all three visualization paradigms mentioned, three video streams should be produced (i.e. ~ 105 MB per minute), and that is without support for any modifications to the output.

3.5 DISCUSSION

In this chapter we studied using the values from a *specialized* device (the Joint Coordinates from the Kinect) alongside with the processed values (the Parameter Values from the Mapping Engine) as timed metadata, to extend the capabilities of an audiovisual application. We presented how timed metadata can increase flexibility in audiovisual synthesis, reduce overall bandwidth consumption and reduce client-side computation complexity. We proposed an initial architecture for developing interactive distributed audiovisual control applications, applied to a Kinect-based scenario, as presented in the `ACM Audio Mostly` international conference [87]. Instead of the server directly sending the output to the client, it sends the mapped input values, alongside the application parameters, in a timely manner. This way, the data processing overhead is divided between the server and the client, and the content consumer is able to customize the final output.

From a communications perspective, our approach is bandwidth efficient, comparing to traditional sound synthesis applications. This is a direct result of the minimal size that the input and parameter sets have, as opposed to the larger audio files that are transmitted otherwise. For sound editing, or hybrid editing and synthesis, in order to maintain flexibility, we might introduce an overhead because both the audio source and the parameters are transmitted, but it is less than 10% of the total file size.

The only scenario that our proposal offers significant disadvantages would be if applied to live mixing, because all audio sources should be transmitted (and their number might vary a lot). However, even in that case, the master-track effects can be distributed as timed metadata, while the per-track effects and mixes already applied to the output audio file. This is a reasonable compromise, because the actual creative product in this case is the way the entanglement of the audio sources is performed, thus there is no need to expose it to possible modifications.

For the experimental part, we used segment duration of 1 s, targeting a balance between introduced delay and required throughput sav-

ings. The associated metadata produced until each segment is ready are transmitted alongside the respective segment. However, for low-latency scenarios, shorter segments might be used in combination with a low-latency protocol. This approach however might introduce out-of-order delivery of frames and (new) asynchronies between the timed metadata and the AV frames, in addition to the pre-existing coming from the middleware processing. To overcome synchronization issues of this nature we propose a client-side buffer-based solution in Chapter 6.

We have also studied if the architecture can be extended to accommodate more demanding immersive scenarios covering applications with server-side generated 3D visualizations and/or real-time audio input. Some of the challenges rising from such use cases include supporting audio editing capabilities synchronized with the audio synthesis and scene navigation with dynamic 3D Sound generation.

In order to support these complex scenarios, we propose packaging all required data (3D Scenes, audio input/output, video input/output, Joint Coordinates and Parameter Set) in mp4 containers. This packaging solution can also be used with DASH and it is suitable for both simple data, such a XML-based Parameter Set [27] and more exotic, such as 3D-based visualizations [26].

4

SPATIOTEMPORAL NAVIGATION FOR EXTENDED VIDEO STREAMS

We demonstrated in the previous chapter how we can extend multimedia content creation with *specialized* devices, by transmitting the device input and the partially processed values, instead of the final output. In this chapter we study extending systems using timed metadata from *common* devices. More specifically, we will examine how we can extend the navigational capabilities of a video collection, by adding spatial and temporal seeking functionalities. Also, this scenario is applicable to User-Generated Content (UGC) systems, where the audiovisual content can be provided by several different sources/users.

Through the past years, there is a surge in platforms that gather and distribute UGC. This multimedia (video, audio, text) UGC content is often challenging to classify and index. The predominant way to achieve this is by attaching tags and descriptions of the recordings. This information can be used to search for relevant recordings, as well as summarize and describe the contents of the UGC.

Several platforms (like flickr [40] and Instagram [55] - among others), allow the user to attach a location (on top of the textual description and tags) to the corresponding recording. Thus, when the user is consuming the multimedia content, she can be aware of the premises and/or the central location of the recording. Additionally, this location can be used to perform a coarse landmark based search. However, in such platforms, this information is without any timing data, used only to give an approximation of the location and it is often missing altogether.

Other platforms, like Google Maps [43], use an actual map to display the available media. Again, the location can be registered by the user thus inheriting some of the drawbacks mentioned in the previous paragraph or captured from the GPS sensor of the device. Regarding the retrieval of the recordings, the user typically navigates to the location on the map and then can browse through the available content.

In this work we merge the two approaches to propose a new paradigm that combines the two main properties:

- provide the user with location awareness during playback
- use map-based visualization for selecting location-relevant content

We elaborate by demonstrating two new paradigms for navigating through video collections and implementing two respective plat-

forms. The first (Spatiotemporal Video Navigation - Section 4.2), suitable for *on-demand* delivery, provides a new feature that is to *navigate to a specific point in the timeline* of a video, based on the location and orientation of the camera. By navigating in time we are able to allow the user to skip directly to the relevant part of the recording, thus enabling long duration mobile videos to be usable in scenarios where the user is interested in a portion of the available recording.

With the second platform (`SWAPUGC` - Section 4.3), we show that spatiotemporal navigation can also be applied in cases where the content consumer follows a specific event (i.e. a specific region at a specific timeframe) via multiple recordings. This approach can be applied to both *on-demand* and *live* scenarios.

In order to achieve the aforementioned capabilities, we use timed metadata gathered during the recording, i.e. the location and orientation data of the device, and instead of placing pointers ("pins") on the map, we place oriented indicative markers. The markers are representative to a corresponding time in the video file and permit robust navigation. Also, the map view is updated during video playback to include the vicinity of the currently displayed location.

4.1 STATE OF THE ART

In this Section we overview some existing platforms for sharing geotagged multimedia content. We selected three cases to demonstrate, according to relevance and popularity criteria. First, the Instagram sharing platform, which is a popular application and website for sharing UGC photographs and videos. Second, Google Maps that is a map service that also offers the capability to browse through user-submitted photographs according to the target location. Then, GeoUGV platform that is the result of a research project and enables recording sensor-enriched UGC videos and navigating through the collection using a map interface. Finally, before we present our contribution, we overview some other useful tools and techniques that can have auxiliary uses in such applications.

4.1.1 *Instagram and Google Maps*

One of the most widespread platforms for sharing geotagged multimedia content is Instagram. Instagram is a mobile application and website that allows the user to share multimedia content. Originally created for sharing photographs, it was extended to accommodate video-sharing functionalities. Even though it is mainly used for searching with tags, one of the features of the platform, is to search according to location. We performed a sample query for media rel-

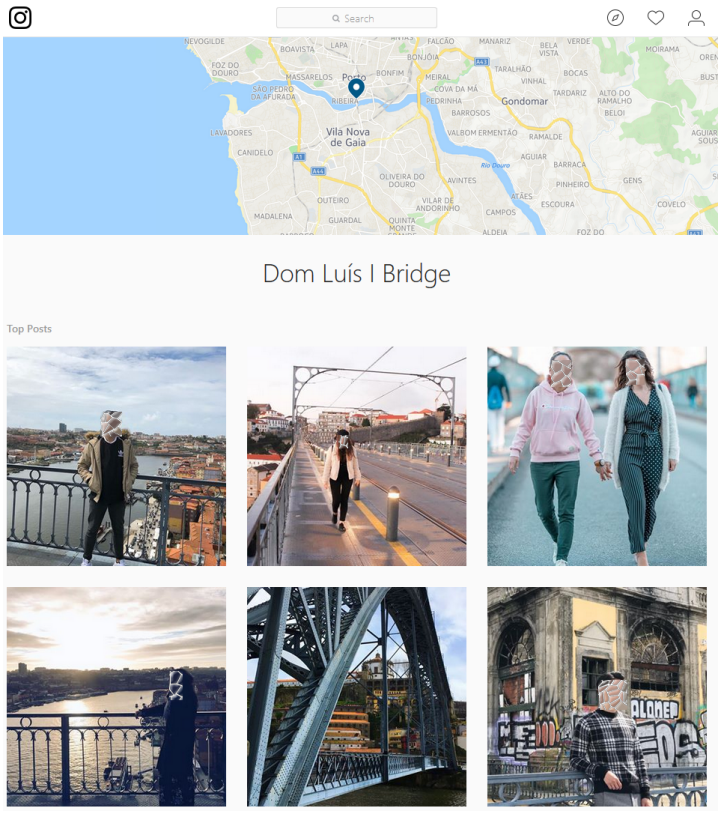


Figure 20: Instagram screenshot of "Bridge Dom Luis" results

evant to the bridge "Dom Luis", located in Porto, Portugal, and the results[1] are shown in Figure 20.

All of the "Top Results" of the query are photographs - corresponding to the main target medium of the platform, and all of the images are shot from the top of the bridge (i.e. the bridge itself is not visible).

Similarly, we performed the same query on Google Maps, with the results shown[2] in Figure 21. We can observe that for Google Maps, from the first ten results, in two of them (green highlight on Figure 21) the bridge is barely visible due to large distance; and in three of them (red highlight on Figure 21) the bridge is not visible at all - possibly due to camera facing, or wrong location data.

Both approaches lack a mechanism to identify whether the location accompanying the recordings is accurate, thus are able to provide only coarse localization capabilities in browsing. Even if the location is accurate, the lack of camera facing information does not permit to filter results with criteria like "Media containing the bridge in the image" or "Photographs taken from the top of the bridge". The GeoUGV platform was designed to address the two aforementioned shortcoming, while functioning as a prototype to facilitate moving from image-centered hosting platforms, to video distribution.

1 as appeared in April 2018 that the screenshot was taken
2 as appeared in April 2018 that the screenshot was taken

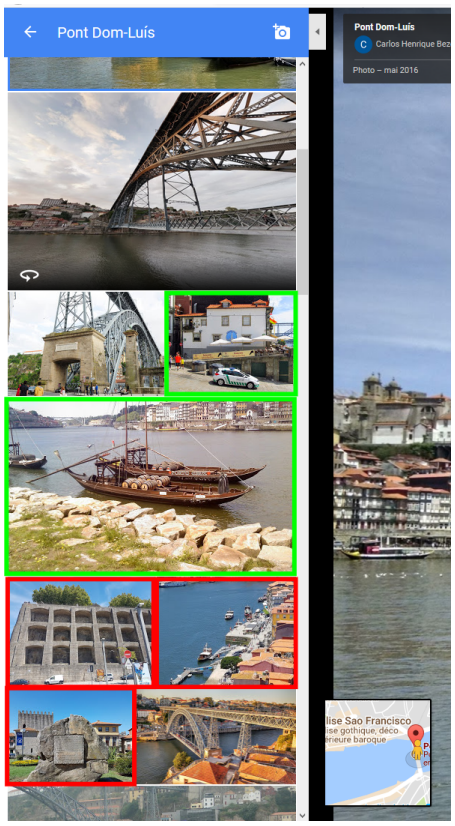


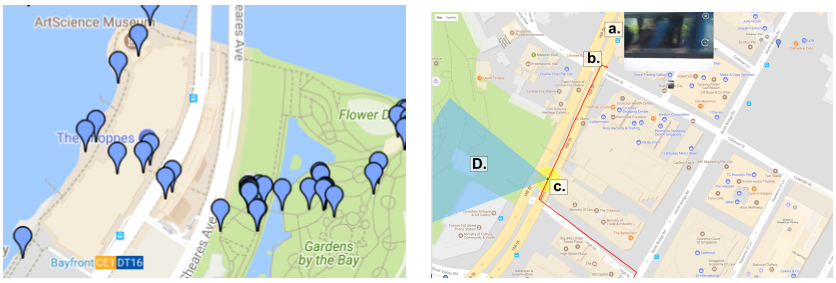
Figure 21: Google Maps screenshot of "Bridge Dom Luis" results

4.1.2 *GeoUGV*

GeoUGV [66] was[3] a platform that aimed at providing to users the capability to navigate through a collection of videos using maps. It consists of a smartphone application that records the videos and the corresponding location/orientation data, and a browser-based client for browsing the collection of recordings. The user is able to find videos according to their location — indicated by pointers on a map, as shown in Figure 22a. The pointer location on the map is set by using the first location sample recorded with the video.

After the user selects the video (by clicking on it), the video playback starts in a window (annotated as [a.] in Figure 22b), and the geospatial trace of the recording is displayed on the map by a red-colored line (starting at point annotated as [b.] in Figure 22b). There is also a pointer indicating the current location of the camera (visible as a dot on the trace - located at [c.] in Figure 22b) and an estimated Field-of-View (FoV) [10], indicated by a blue-colored overlay starting at the pointer position (annotated as area [D.] in Figure 22b). As the location updates so does the pointer position and respectively the FoV with the matching orientation values.

3 Since March 2018 and as of the time of writing of this manuscript (February 2019) the GeoUGV platform is offline (`http://api.geovid.org/v1.0/web/viewer`)



(a) Video listing on the map (b) Video playback interface

Figure 22: GeoUGV captures

This platform, even though developed in an academic environment and not as a consumer product, introduced several novel contributions on the field of geospatial navigation of videos. It is the only one that supports functionalities enabling to follow the spatial progress of the video in time. Also, it introduced querying of videos according to their FoV coverage of a Point of Interest (PoI). Finally, the geospatial timed metadata were captured and uploaded in an automated way directly from the sensors by the recorder application, thus eliminating fraudulent or mistaken entries.

However, there are downsides in the approach, that we are trying to address with our proposal. First, even though the FoV can be used for indexing and searching for relevant videos in databases, the visualization is not always accurate — as in the screenshot where the FoV visualization shows the video capture covering a "green" area of almost two building blocks (blue-colored area [D.] in Figure 22b), but the view is actually obstructed by a nearby building (as seen in the preview window [a.]). Secondly, the map visualization is updated during playback, but the user cannot navigate in the video by using the map interface (e.g. by clicking on a specific point on the trace). Finally, before starting playback, the consumer has no indication on the duration of the video, the course followed during the recording, or the facing of the camera. As an example, a user might be standing close to the desired PoI on its East side and quickly moving away for it, or continue recording with the camera facing on the East thus never recording the desired scene. Therefore, the content consumer is not able to identify the video as irrelevant prior to watching (part of) it.

4.1.3 *Other Tools and Techniques*

Concerning other approaches on browsing UGC video collections, there has been studies on the domain of multi-view video [6], that target concurrent recordings with the motivation of following a moving object (person). The four proposed interfaces are "Swap " (ges-

(a) Swap gesture (b) Map overlay

(c) Marker overlay (d) Carousel preview

Figure 23: Interfaces of multi-view video navigation (from [6])

ture based), "Map " (overlay), "Marker " (overlay) and "Carousel " (preview); screenshots of these approaches are shown in Figure 23.

The authors were focused on interaction techniques, thus we do not have any interesting architecture designs or system performance and scalability considerations mentioned. However, they conducted a user study which indicates that the Marker technique scores highest according to the "Overall QoE" criterion, followed by Map and Swap. On comments by the users, Swap is considered unhelpful since it does not have any spatial information, and similarly for Carousel, users comment that they just scroll through the videos until they find one that they like, without considering the camera positioning.

With regards to the criteria that measure spatial awareness of the user (i.e. "Distance perception", "Direction perception", "Learnability") Map and Marker rank high, with users commenting that both can be improved (e.g. by choosing more indicative shapes for the markers/pointers). Since our approach is close to those two paradigms, we have considered this input thoroughly, to make an interface that is easy to navigate, yet providing intuitive markers with clear spatial indications.

A feature that can be essential in some use cases of automated navigation is automatically identifying relevant recordings. GeoUGV uses FoV estimation [10], that is computationally inexpensive, but it can suffer from view-blocking (as mentioned in the previous section). A different approach is using image features [19], but due to its processing requirements it is not scalable nor supports low-latency scenarios. A solution that balances precision with complexity would be to fuse the two approaches by using FoV estimation for coarse location calcu-

lation, and then apply image features extraction, on sampled frames, for validation of the results.

In addition to identifying spatially-relevant recordings, the quality of the temporal information of the streams must be verified. This can be done during the recording, if the devices are interconnected, and a clock synchronization protocol (like NTP) is used. Alternatively, an extra processing step correcting potential misalignment of the timing information is required. For example, in the multi-view video study presented in this section, the authors use the ConCor+ algorithm [4], but other techniques that use audio/video features can be applied instead [20]. These approaches are computationally expensive but can be used to synchronize, with sufficient accuracy, streams from any device. The accuracy of the attached timing information is especially important in the scenario where the user is following a specific event, to avoid unexpected temporal "jumps" when switching between views.

Typically the actual video content upload is trivial, because we are are discussing on-demand scenarios. Specific provisions can be taken for outlier use cases. For example, in constrained environments, techniques like NEWSMAN [98] can be applied. Originally designed for uploading regular and breaking news, NEWSMAN can provide incentives on using middlewares for scheduling and transcoding of outgoing streams. The main principle is to implement schedulers in the middleboxes to facilitate uploading of recordings. The outline of the proposed scheduler is illustrated in Figure 24. First, the recordings are prioritized according to annotated "importance" level and subsequently ranked in the queue according to the assigned priority. Then, the network is evaluated in order to generate relative bitrates for the queued recordings. Finally, the generated prioritised bitrates (referred to as "jobs") are scheduled for upload.

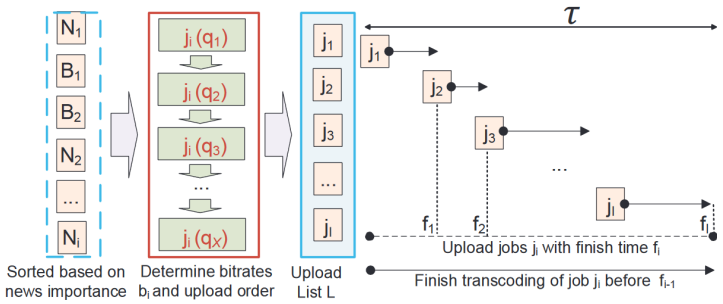


Figure 24: NEWSMAN Middlebox scheduler architecture overview (from [98])

We implemented our own platform for playback of spatially annotated videos with orientation information. Our platform consists of a smartphone recorder application for the Android OS [4] and a browser-based client for navigating through the recordings [5]. The novelty of our implementation comparing to the previous approaches is that instead of placing a single generic pointer on the map to indicate a recording location, we use the accompanying location/orientation samples to visualize several keypoints with oriented markers.

The previous approaches were based on the assumption that the recording devices have a fixed location, which might not suffice to identify relevant recordings when the cameras are mobile. Even GeoUGV which can be used to virtually follow the spatial trace of the recording on a map does not differ in indicating the video locations. Therefore we show the keypoints (as recorded by the GPS sensor) on the map, even if the video is not selected, as shown in the screenshot in Figure 25.

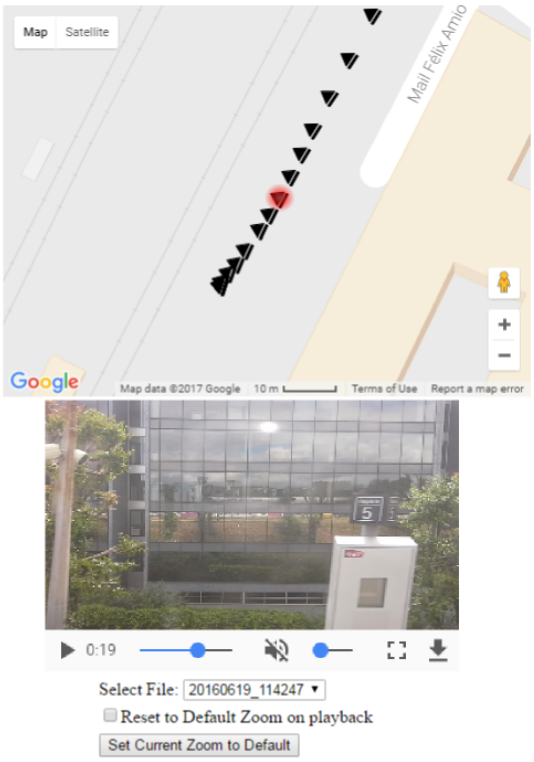


Figure 25: Screenshot of Spatiotemporal Video Navigation

By using one marker per sample, instead of a continuous trail, the user is able to identify the location (from the marker placement), and infer the velocity of the producer (from the concentration of the mark-

4 https://git.io/SNR
5 https://github.com/emmanouil/Spatiotemporal-Navigation

ers), at any given point. During playback the map centering is updated to match the location of the current video content and the exact location is indicated by highlighting (in red at the screenshot) the currently active token.

Also, instead of trying to calculate the FoV and display it, we just indicate the direction (facing) of the camera, by using the data from the orientation sensors. This is visualized by shaping the marker as a triangle (to imitate the camera view cone) starting from where the camera is located during the recording. This way, the user can have an incentive on the direction of the camera at any given moment (as opposed to previous work that it is only for the current frame), which is essential when searching for a video recording a specific area. Also, we avoid misleading calculations of FoV, as in Figure 22b, where we have on the map a broad field of view, while in the video the view is obstructed by a building.

With the discrete markers, we were also able to implement temporal navigation functionalities. The user, by clicking on a map maker, not only selects a spatially relevant video, but at the same time he skips to the temporally relevant part of the video. So, we also add a time dimension to the video navigation.

Finally, by having the points visible at all times, as opposed to display the trace of a video after clicking on a single point, the user has a more representative view of the video content and timeline, before the selection of it. At the same time, the density of the tokens indicates if the producer emphasized at a specific location during the recording (dense markers), or if she is just "passing by" that area (sparse markers).

However, there are some downsides to our proposal. More specifically, in case that there are multiple recordings in a geographically limited area, the resulting congestion of markers on the map might be confusing to the content consumer. Even when we remove some markers from the view, as the user zooms out on the map, the problem persists when the number of available recordings is significant. The negative effects can be eased by using per-video color-coded marks, but this approach also has scaling limitations.

4.2.1 *Architecture and Implementation*

In order to examine the capabilities of timed metadata for spatiotemporal navigation of video streams, from common devices in on-demand scenarios, that we demonstrated in this chapter, we developed a software suite dubbed "Spatiotemporal Video Navigation", comprising of two parts:

- **Spatiotemporal Navigation Client** [6]: browser-based client capable of displaying videos with spatiotemporal navigation capabilities. Also, the client repository contains a python module for processing and formatting the recorded streams (SN-Parser).
- **SN-Recorder** [7]: android application for recording videos with sensor data.

The input of the client for each recording (bundle) consists of the following files:

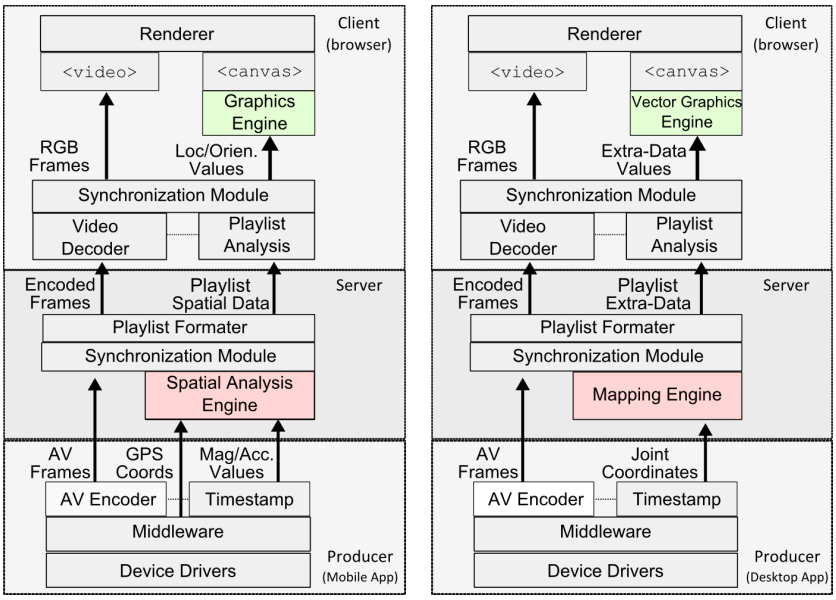
- *Descriptor*, with the required synchronization information, the location of the other files and other optional information (such as recorder ID, recording device type ...).
- *Playlist* of the video segments for the video stream.
- *Location File* with timestamped Latitude/Longitude pairs.
- *Orientation File* with timestamped Yaw/Pitch/Roll triplets.

From a system design aspect, the implementation is based on the original architecture we presented in Chapter 1. We can see the outline of the architecture for Spatiotemporal Video Navigation in Figure 26a. For comparison, we show the architecture for the Kinect-based Synthesis platform (studied in Chapter 3) in Figure 26b. Both systems use different types of metadata, to extend in different ways the corresponding systems, but they share similarities, e.g. in their synchronization engines that both use timestamps and frame numbers to achieve inter-stream synchronization.

Regarding the differences between the two platforms, the Mapping Engine of the Kinect system (previously detailed in Section 3.3) is replaced by a Spatial Analysis Engine that is responsible for formatting the recorded spatial data and extracting information based on the Location/Orientation data (accuracy, bearing etc.). Because both the Mapping and the Spatial Analysis engines input the sensor timed metadata and output the transformed timed metadata, they can be macroscopically seen as different engine "modules" of the same architecture.

On the client side, again we use the `video` HTML element to render the captured video and a `canvas` element for the map, markers, icons and map overlays. Because we are using external APIs (in this case the Google Maps API), the graphics engine, during the initialization stage, works by sending synchronous requests at the remote service. After the initialization, the platform can be robust and lightweight, because we are using only native HTML elements and APIs without any custom plugins.

6 https://github.com/emmanouil/Spatiotemporal-Navigation
7 https://github.com/emmanouil/Spatiotemporal-Navigation-Recorder



(a) Spatiotemporal Video Navigation Architecture

(b) Kinect-based Control Architecture

Figure 26: Outlines of implemented architectures

Our platform can be easily modified to accommodate visualization modifications. As an example, we are currently rendering all the available GPS points (an average of one sample per 1-2 s), however it can be modified to display markers only on location changes, or at specific time intervals. Another extension is the interpolation of samples (according to time or distance criteria), when the GPS trace is sparse due to signal loss, since the orientation is sampled at a fixed rate.

Spatiotemporal Video Navigation is an experimental platform that we built - based on GeoUGV, as a mean to examine the flexibility and feasibility of our proposed architecture. Its goal was to give us more insights on the characteristics, requirements and behaviour of the *common* devices class applications and explore the possibilities of AV systems with spatial awareness. Due to the time limitations of the Thesis, we did not perform evaluation tests on this incarnation, but we did evaluate the SWAPUGC platform, that is the evolutionary offspring of Spatiotemporal Navigation.

4.3 Software for Adaptive Playback of User-Generated Content (SWAPUGC)

In the previous section we demonstrated how a user can navigate through videos in the spatial and temporal domain. Here, we examine a different paradigm which is *navigating through the different views of simultaneous recordings*. Several views overlapping in time

and space are available and the client can switch between them, either on command of the user or in an automated way.

Again the client should be aware of the spatial characteristics of each view, in order to be able to synthesize a representation of the available recordings. However, in this case, the inter-bundle synchronization requirements are more demanding. In Spatiotemporal Video Navigation switching between views with misaligned timing information (i.e. the reference clocks defer by several seconds) would not penalize the QoE - therefore we had *loose* synchronization requirements. However, in this case, switching from a view to another, while following the same space on a common timeline, would gravely disrupt the experience (as examined in the following Chapter 5). Thus, we specify the synchronization level to *critical*.

As mentioned earlier in this chapter, if the common timing reference is not inherently present, it can be obtained by means like feature extraction and matching. This common timing reference is used to identify and synchronize relevant streams.

Starting from Spatiotemporal Navigation Client we developed SWAPUGC [8] [9] as a tool to experiment on consumption scenarios for multiple recordings [85]. Both of the platforms take the same input, however SWAPUGC, when the strict synchronization requirements are met, is able to support live scenarios with multiple recordings and multiple video bitrates per recording (i.e. "representations" in adaptive streaming systems).

4.3.1 *Technical Description*

First, during the initialization phase, SWAPUGC fetches and parses the `JSON` descriptor file for each bundle. It contains synchronization information that is a required reference point in time, typically indicating the start of the recording. Also, the descriptor contains reference to the other files required, for retrieving the video, location and orientation. An example of other application-specific information that can be inside the descriptor file is the lens characteristics, used for applications estimating the Field-of-View [10]. After the timing information for each bundle is parsed, the earliest recording is identified and its timeline is used as a reference during the simulation.

Then, the playlist of each recording is fetched. The playlist contains the available video representations (i.e. the available qualities), and references to the byte-ranges or files of the representation segments. The justification of the architectural design choice to support multiple representations and segmented video files is to simulate a realistic distribution scenario like Adaptive Streaming over HTTP (e.g. DASH,

8 ACM MMSys archived version: https://github.com/acmmmsys/2018-SWAPUGC
9 active development repository: https://github.com/emmanouil/swapugc

HLS). Then, from the extracted information the video player is initialized, which, since our platform is browser-based, is a `video` HTML element. During playback we update the `video`, by feeding its buffer, using the `Media Source Extensions` (MSE) API.

After the setup of the video player, the Location and Orientation data are fetched, to be used during playback. When the initialization phase is completed, the earliest starting recording is selected as a reference, meaning that it is the initial view, and consecutive events are fired on its timeline. The interface outline depends on the implementation, but for the purpose of this example we consider a default setup with a map having markers indicating the location of the recordings and a video showing the currently active view, as shown in Figure 27 and explained in the following Section 4.3.2.

When playback starts, the default view is displayed and the respective location/orientation is updated according to the timestamped sensor measurements. As a recording (with the accompanying spatial information) becomes available the user can switch to it. Alternatively, the switching can occur via automated view selection policies - such as going through the available views at fixed intervals (round-robin).The selection policy can take into account the available spatial information to decide when to switch (e.g. when the camera moves close enough to the subject). The throughput between the server and the client can be throttled to emulate different network conditions, and by designing our platform to support representations of multiple video qualities, quality-adaptation algorithms can be applied. This way the stream selection algorithm can consider both networking metrics and spatial information to decide on the most suitable recording and quality. To the best of our knowledge, SWAPUGC is the first open source player allowing for dynamic view switches in a DASH context. In the following section, we describe the implementation of the SWAPUGC client application.

4.3.2 *Example Implemented Application*

We created the SWAPUGC client in order to test the scenario of simultaneous recordings consumption. Our application supports a manual switching functionality, in which the user clicks on a marker and the main view switches to the recording from that camera. Also, we created a naive view selection policy, in which a 10 s round-robin algorithm switches between views.

The interface of the application consists of a map and a video element. After loading the timing characteristics, SWAPUGC parses the location measurements and the initial markers are placed on the map indicating the recording location corresponding to the beginning of each -currently available- video. Also, the respective orientation value is parsed, in order to rotate the marker, to emulate the facing of the

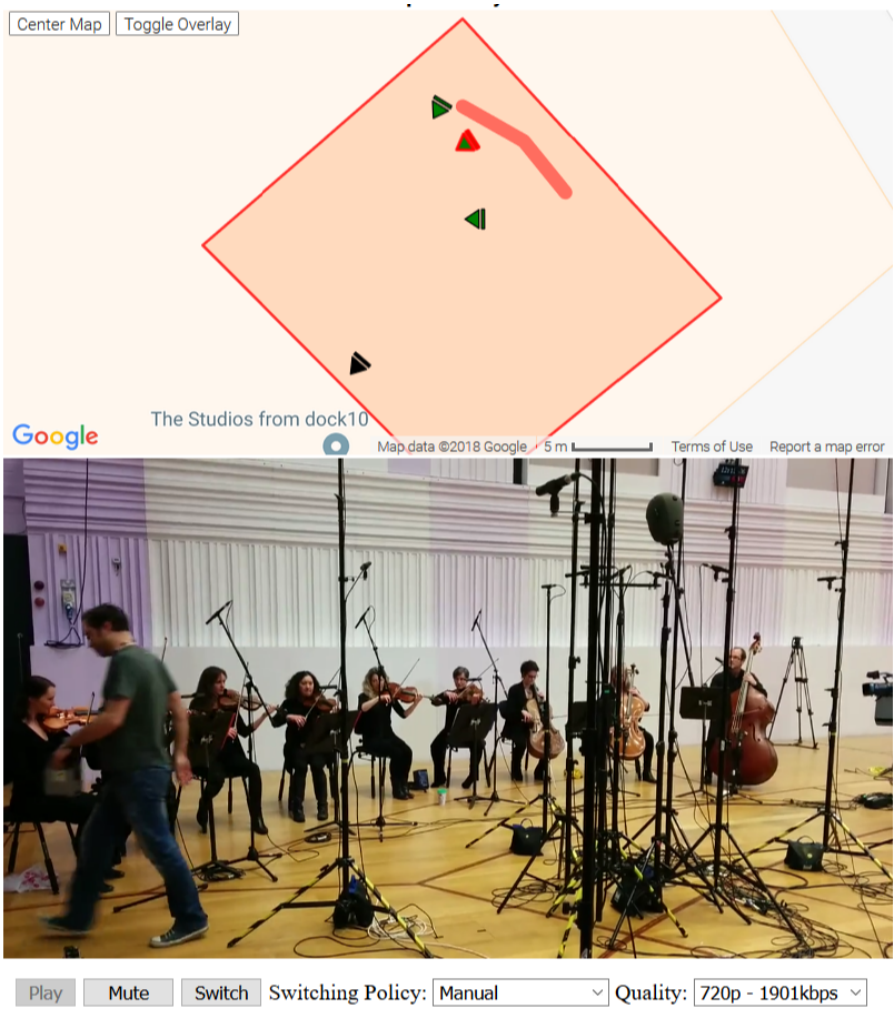


Figure 27: Screenshot of the SWAPUGC implemented application

recording device, as shown in Figure 27. Thus the markers *indicate* the field of view (they are *not* simulating it), and it is the same technique as previously shown in Figure 25, with the difference that instead of having one marker for a specific timestamp, we have one marker for a specific recording.

When the playback starts, the placed markers are updated, as new orientation / location measurements are parsed, corresponding to the timeline. If a recording becomes available at any time during playback a new marker is placed on the map, or removed if it becomes unavailable. When the user clicks on a marker, the video switches to the selected view.

On top of the click-to-activate view, which is based on the user, the platform can operate in an automated fashion. Since all the characteristics of the recordings are available, a view selection policy can be defined that will switch to the better matching view. Events can be fired to indicate when a view has become available/unavailable.

To test this application, we used a dataset provided by the ICoSOLE project[11], containing studio recordings of the BBC philharmonic[10]. More specifically, we used 4 recordings in total (from "Take #5"):

- `A002C001_140325E3` fixed studio camera - no sensor data
- `20140325_121238` roaming handheld camera - sparse orientation data
- `20140325_131253` roaming handheld camera - sparse orientation data
- `Take5_Nexus5` roaming handheld camera - orientation data

Because the recording session was indoors, the location data was severely inaccurate and we did manual annotation. Also, the recordings with id `20140325_121238` and `20140325_131253` had very coarse and sparse orientation data, thus were recreated, using the open-source Spatiotemporal Video Navigation Recorder[11]. A screenshot of the application running with the three recordings is shown in Figure 27.

We used GPAC [62], to create segmented video files so they can be distributed in a manner that would resemble a live scenario (potentially with multiple qualities for adaptive streaming). The segment playlist is in `mpd` format, which is natively supported by SWAPUGC, in order to be compatible with the MPEG-DASH standard. All of the aforementioned files and the source code are available in the archived SWAPUGC repository, under a Creative Commons Attribution Non-Commercial Share Alike license (CC BY-NC-SA 4.0). Also, a website hosting a demo of the application is available online[12].

4.4 DISCUSSION

In this chapter we demonstrated another way to extend the capabilities of a multimedia system, this time by using the timed metadata to achieve a novel approach to spatiotemporal video navigation. This video navigation paradigm is feasible due to the recording and distribution of the timed location and orientation metadata. We examined two different approaches, one for each of the two different paradigms for spatiotemporal navigation, the first for navigating (in time and space) through videos, and the second for following a specific event. We provided the implemented platforms (Spatiotemporal Video Navigation and SWAPUGC) as open-source projects, suitable for research purposes.

10 Available on the SWAPUGC repository is also the parser used to extract timing information and format the XML-based data of the dataset, to the SWAPUGC JSON-based format.

11 https://git.io/SNR

12 https://acmmmsys.github.io/2018-SWAPUGC/

For the first approach, where the paramount goal is to allow the user to navigate in the time of the video that is of spatial interest, by displaying discrete oriented markers on a map for the duration of a video recording, we are able to offer the user the ability to select the relevant video and timeshift to the respective point in time, thus in practice navigating to the specific frame of interest. This approach can be useful in cases where the producer of the content is not static at a fixed location, because the consumer will be able to preview the movement trail and camera orientation at all times. On the downside, this can lead to map overloading with markers that can obscure the map overview and overwhelm the user with available keypoints.

For the second scenario, where the target is to offer flexibility to the user when following a specific area at a specific time (i.e. an event), we use the same principle of rendering oriented markers, instead that for this case, there is only one marker per camera and the location/orientation is used to update its characteristics. This approach is focused in navigating through the views, to find the most suitable one (either manually or automatically).

In the following chapter we studied different adaptation policies that we tested using SWAPUGC. We also address UX issues we encountered during the development of Spatiotemporal Video Navigation and SWAPUGC, like video source selection etc.

Part III

IMPROVING VIDEO DELIVERY

For the final part of this manuscript we enhance video applications by using the timed metadata to improve delivery of the content. We are mostly focusing on User-Generated Content, though the techniques demonstrated can be applied to other sources too. First, we study using timed metadata for multi-variable adaptation in multi-view video delivery, and then, we present a buffering scheme for synchronous and low-latency playback.

5

TIMED METADATA-BASED ADAPTATION OF SENSOR-ENRICHED VIDEO STREAMS

In the previous Chapter, we studied how timed metadata from common devices can be used to extend the navigation capabilities of a video collection. Due to the popularity of mobile phones and cameras with embedded geospatial sensors, in events that attract masses, like concerts, plays, performances etc. several users might use their devices to record that event simultaneously. Social networks like Facebook and YouTube have provided the users with capabilities of live streaming these events. However, with the aforementioned platforms, viewers cannot select the streams with regards to the event, but with regards to the user that records it.

This approach seems to be evolving, since often the recordings come from cameras and/or smartphones capable of recording spatial information (location and orientation), which is used to identify recordings relevant in space and time (i.e. from the same event) [31]. Also, we saw in the previous chapter, that new platforms emerge (like Spatiotemporal Video Navigation and GeoUGV) that facilitate browsing through the selected relevant recordings. After identifying the relevant recordings, selecting the most "suitable" view during playback is a separate task. There has already been some work on the stream selection front, with approaches that evaluate the video quality according to the content, or the spatial sensor measurements [30], reviewed in the following section.

Since there are works on identifying the recordings and automating the view selection process, the final frontier towards a live distribution scenario is the actual streams delivery. As mentioned previously, for video delivery over the internet, it is common practice to apply HTTP *Adaptive Streaming* (HAS). Therefore, *in a realistic live distribution scenario of multi-stream UGC video, the client has to select the most suitable recording* and *the highest supported available bitrate of that recording*. Even though we are using HAS as an example throughout the chapter, the principles of our work can also be applied to setups that the recording and bitrate selection is transposed from the client to the server (e.g. RTP delivery).

In this chapter we examine all of the aforementioned factors and provide insights on techniques that use timed metadata for selecting the streams that would improve the user experience when viewing live events via UGC. The main contribution of our work is that we consider live scenarios, in which the content is consumed as it is produced, so we are proposing a scalable approach. In order to achieve

that, we focus on client-side solutions, because the server-side processing should be kept to a minimum, in order to increase the maximum number of concurrently connected users and reduce end-to-end delay. In our proposed architecture, the server should only be responsible for maintaining the synchronization between the gathered UGC streams, expose stream characteristics (metrics) and generate different video qualities/representations. To achieve that, we delegated functions to the content producers (e.g. image quality assessment) and the clients (e.g. stream evaluation).

On top of the proposed techniques to build a system supporting live streaming of multiple UGC recordings, we are identifying the user experience effects of the technical infrastructure. More specifically, HAS systems rely on evaluating the link between the server and the client, but we also study how the client should behave when the network connection of the producers to the server is unstable.

To the best of our knowledge, this is the only study that takes in account both content and network parameters, to optimize live delivery of multiple UGC streams of the same event. However, there are several works that contain elements that we use in our approach, and we review them in the following Section 5.1. Then, in Section 5.2, we analyse the algorithm used for the Stream Selection and in Section 5.3 the metrics we used as inputs. Following, in Section 5.4 are considerations for implementations of our proposal and in Section 5.5 is the setup and parameters used for the experimental part of our study. In Section 5.6 we review the results of the experiments and we conclude in Section 5.7 with a discussion on our proposal and future work on the domain.

5.1 STATE OF THE ART

There have been several studies on view selection, including for sensor-based UGC content. We are using this previous work to design our system, and extend it to make it suitable for live scenarios and to apply network condition provisions.

On stream selection, we cannot use content-based solutions that require computationally expensive image analysis algorithms [69] [118] and/or training [96]. Nonetheless, because we use some of the proposed techniques (e.g. for algorithm modeling, scene length etc.), we review these solutions in this section.

MoViMash [96], is a framework for creating a video mashup (i.e. a video sourced by multiple cameras) of music performances using as design principles, Video Quality, View Quality, Diversity and Learning. Our approach is close to MoViMash because we are using similar principles, but, whenever possible, we replace the computationally expensive image recognition algorithms used (especially in Video and View Quality assessment) with sensor-based estimation.

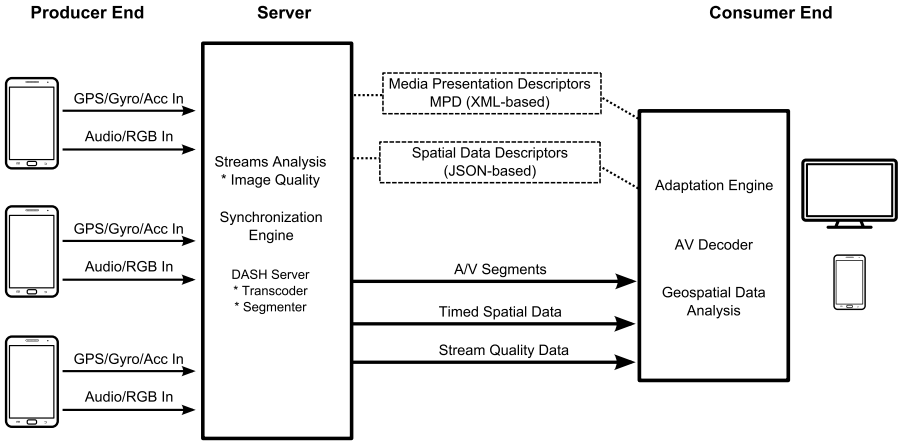


Figure 28: System architecture overview

Also, concerning the architectural implementation, since we consider live scenarios, we do not use classification and class prediction (machine learning-based) of the videos, but we keep modified versions of the pre-filtering and ranking methods. More specifically, we follow the suggestion of the authors for first evaluating "relevant" views, according to discrete objective criteria (e.g. currently selected view is excluded from the available views), then rank the remaining views according to the metrics and finally decide on the next view and duration of the scene.

The aforementioned proposals also offered pointers on the cinematic guidelines we follow. We picked criteria that can be applied to a wide range of videos, by cross-selecting guidelines from example applications in relevant works varying from music performances [96], to sport events [8]. These criteria include achieving view diversity, calculate scene duration and general video editing directives (such as keeping the 180-degree rule[1] or avoiding jump-cuts[2]). We omitted audio-related directives, because we do not perform any audio quality evaluation and switching, but instead we maintain one audio source for the duration of the experience.

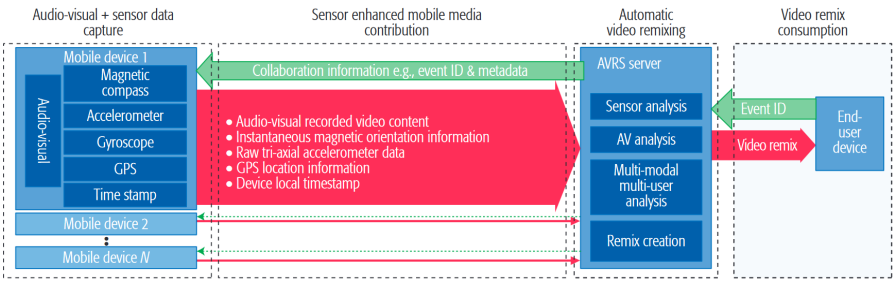


Figure 29: AVRS end-to-end system design (from [69])

1 https://en.wikipedia.org/wiki/180-degree_rule
2 https://en.wikipedia.org/wiki/Jump_cut

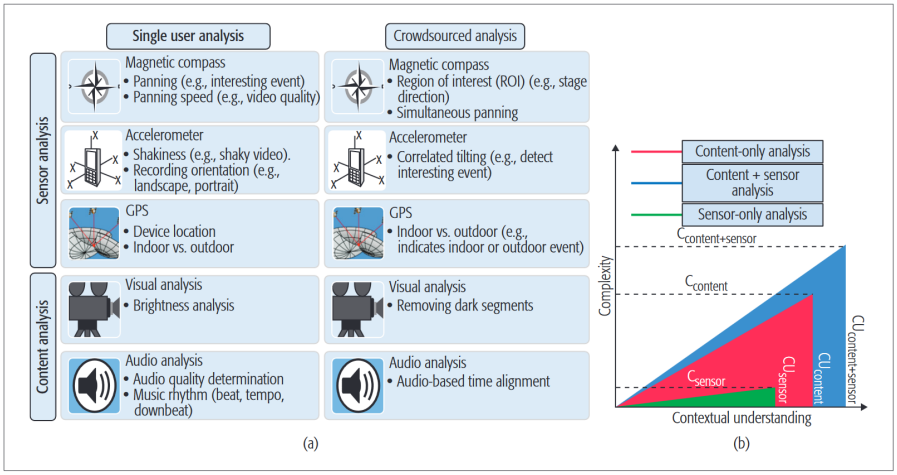


Figure 30: (a) Sensor, Content analysis methods and (b) comparison (from [69])

Automatic Video Remixing System (AVRS) [69] is a proposal for creating multi-camera "remix" videos of events. Even though it is targeting offline distribution scenarios, where a single video "remix" file is generated on the server and is sent to the clients, the proposed system design (Figure 29) for the sensor-based mode is similar to the one we used (Figure 28). Also, this work illustrates the comparison between sensor-based and content-based stream evaluation, and validates our choice to focus on the former, to minimize system complexity in order for our proposal to be scalable (Figure 30).

The sensor-based mode of AVRS is based on previous works on the domain of multi-modal information extraction for UGC videos [31]. The latter work also offers insights on detecting relevant recordings and Regions of Interests (RoIs) — techniques that can be used with the stream selection algorithms. However, regarding the camera-selection algorithms, we couldn't validate the proposals, since the datasets (one described as "real-world" and one "obtained in a simulated scenario") used in the paper are not available. Nonetheless, there has been extensive work on the domain, for identifying RoIs, specific event time and location, highlights etc. [21]. Our recommended method is to use a purely sensor-based approach, because it has very low computational complexity, yet is accurate enough. This method first applies low-pass filter on the gyroscope to exclude noisy values, then estimates the FoV (assuming a conservative view width of 90 degrees) and discards outlier recordings [31].

Indexing and retrieval of the relevant videos is an area that we do not focus on this work, but there are approaches that can be used to build such applications. The proposed methods [10] [31] are based on calculation (not estimation) of the FoV, that suffer from the drawbacks we mentioned on the previous chapter. However, FoV calculation can be used to quickly provide a set of most relevant recordings, and then

filter out those that are not suitable by other means that can be more "expensive" (e.g. with image feature extraction).

We also consider the *blurriness* of an image as a stream quality metric. We are using blurriness because artefacts/distortions from other causes (compression, shakiness etc.) can be either deduced from other characteristics (e.g. use bitrate to estimate compression level) or sensor-based metrics (e.g. shakiness). Regarding the specifics, we calculate the blurriness level using Laplacian edge detection techniques – that is we measure the variance of the second derivative of the grayscale version of the frame [81]. Other blur detection techniques can be used, on the condition that they are computationally inexpensive to allow the system to scale for several sources [82] [29].

All of the aforementioned works address aspects that are relevant to our approach and we incorporate these elements to our proposal, but to the best of our knowledge, there is no previous complete work on designing systems for live adaptive delivery of UGC content.

5.2 PROPOSAL OVERVIEW

In this section we propose and overview a model for view selection that is based on the state of the art. We elaborate more on the details of the metrics in the following section and the implementation-specific details are in Section 5.4. Regarding the required sensors for our model to be applicable, we assume that all the recording devices provide the location (e.g. GPS sensor) and orientation (e.g. Accelerometer / Gyroscope) data; features that are common to modern smartphones and digital cameras. The location measurement consist of Latitude, Longitude and Altitude values, and the orientation of Pitch (a.k.a. Pan - X axis), Yaw (a.k.a. Tilt - Y axis), Roll (Z axis). From the aforementioned sensor data we can calculate metrics for Shakiness, Roll & Tilt, detailed in the following section, and approximate the Field-of-View (FoV) and Region-of-Interest, as in Appendix A.2. As a visual quality metric, we are sampling frames of each recording and perform a blurriness-based image quality estimation. The last metric we use for the stream evaluation is Link Reliability estimation, to indicate the quality of connection between the recorders and the server.

To summarize, we consider the following objective criteria for stream evaluation:

- **Field-of-View (FoV) estimation** (location/orientation - sensor-based)
- **Shakiness** (orientation - sensor-based)
- **Roll & Tilt** (orientation - sensor-based)
- **Bitrate** (metadata)

- **Image Quality** (metadata)
- **Link Reliability (LR) estimation** (metadata)

All of the aforementioned criteria, are based on values gathered (i.e. orientation and/or location for Shakiness, Roll & Tilt, FoV estimation), or computed (Bitrate, Image Quality, Link Reliability estimation). The values used for the metrics are gathered on the production end and then they are sent to the server, where they are used to calculate comparative metrics (e.g. Image Quality).

We define a view change/switch event as the time that the client choses to perform a transition between recordings. During playback, the time for the view change event and the suitable available views are selected according to subjective cinematic criteria:

- **Scene Duration**
- **View Angle**
- **View Distance**

Also, unscheduled view changes might occur if the currently selected view becomes unsuitable (i.e. does not film the RoI), or unavailable (i.e. the content producer disconnects).

We detail the algorithm for the stream selection in the next section, but it can be summarized as follows:

1. First (filtering phase), mark as irrelevant recordings that do not film the RoI (using FoV estimation).
2. Then (ranking), we order the remaining streams according to the metrics.
3. Finally (switching), when a view change event occurs, we switch to the highest ranked view that meets the cinematic criteria at a bitrate that can be supported by the client connection.

The ranking is updated as new data arrives, so it is available if an unexpected premature view change event occurs, otherwise the view change happens according to the scene duration defined by the cinematic criteria.

Finally, we address the case that the recorder-to-server network of a stream becomes unstable soon after it has been selected after a switch. This case has not been thoroughly studied in the literature, since previous work is focused on offline scenarios, or live scenarios without considering the actual video delivery [31][69][96][118]. The switch-and-unstable case can be divided in two sub-cases.

1. The network connection between the server and the active recording (i.e. the producer) fails, thus the stream is disconnected.

2. The network connection between the server and the active recording (i.e. the producer) degrades, potentially drastically changing the position of the recording in the ranking.

For the degradation sub-case, the client can either stick to the recording, with its new lowered quality, until the next switch, or speedup the next switch, risking to disrupt the user experience with the abrupt change of view. Because both courses of action have downsides, we included this scenario in the user-study to examine whether it is preferred to stay in the current stream or speed up the switch in such cases.

5.3 STREAM SELECTION POLICIES

Regarding the stream selection policy we used our own version of the MoViMash [96] algorithm, adapted for live scenarios. As mentioned in Section 5.1, we removed from the stream selection engine processes that might affect the scalability of the system (e.g. online learning), or that are application-specific (e.g. classification), but we kept the filtering, ranking and switching steps, with their criteria described in the following subsection. The stream selection engine is responsible for *filtering* the irrelevant streams, *rank* the remaining streams and finally *switch*, at the right time, to the one with the highest score.

5.3.0.1 *Filtering*

For the filtering phase, each stream is considered to be relevant if it is filming the *Region-of-Interest* (RoI), and the rest are excluded from the pool of available streams. Because in order to calculate accurately the FoV, several camera characteristics (e.g. zoom level, camera angle, lens aperture etc.) are required but are not always available, when they are not present, we are using an estimation of the FoV - based only on the location and orientation of the device.

5.3.0.2 *Ranking Metrics*

After the filtering, the available streams are ranked according to their score. The score is calculated on the metrics for Shakiness, Roll & Tilt, Image Quality, Bitrate and Link Reliability.

Shakiness (S_s), is the metric that indicates instability (i.e. sharp and/or constant movement) of the camera, and can be calculated using the accelerometer of the device. More specifically, the recommended algorithm is [31]:

1. Set a sampling window and shakiness thresholds (Ts_{min}, Ts_{max}).
2. Filter the values through a 10 Hz high-pass filter, for all axes (Pan - X, Tilt - Y, Roll - Z). This step is required to identify the type of movement – that is to separate panning from shaking.

3. For each set of samples:
    a) Compute the sample variance of the three filter components (let Ex, Ey, Ez).
    b) Compute the median value of Ex, Ey and Ez, which is going to be the shakiness value (M_{shake}). This step is required to further eliminate the possibility of measuring a fast panning (that happens in one axis) from shakiness
4. If $M_{shake} < Ts_{min}$ there is no shakiness ($S_s = 0.0$), if $M_{shake} \geqslant Ts_{min}$ there is shakiness ($S_s = 0.5$), if $M_{shake} \geqslant Ts_{max}$ there is strong shakiness ($S_s = 1.0$)

Higher shakiness values indicate worse perceived video quality, thus when calculating the final score, we use the complement to 1 of the value ($1 - S_s$).

Roll & Tilt (S_t) is a metric extracted from the orientation sensors of the device. The Roll, represents the angle between the left-right horizontal axis of the camera and the ground [3]. Tilt (Yaw) indicates the angle between the font-back horizontal axis of the camera and the ground – i.e. if the camera is shooting straight ahead should be equal to 0°, if it is shooting downwards it should be negative and if it is shooting upwards positive. As the absolute Roll angle increases the perceived quality degrades, and we empirically found that a view is "very disturbing" if the difference between the parallel to the ground value reaches ±10°. The ±10° is very conservative comparing to MoViMash [96] that has a similar approach to Roll[4] measurement, with the difference that they degrade the score for values up to ±30° and if the Roll is higher, they completely remove the video from the pool. We consider each degree of deviation to reduce the score per 0.1, thus for an angle a_R, the Roll & Tilt score (St) is calculated as shown:

$$S_t = \begin{cases} 0, & \text{if } a_R = 0 \\ |a_R| * 0.1, & \text{if } |a_R| \in [1, 10] \\ 1, & \text{if } |a_R| \in (10, 80) \end{cases} \quad [5]$$

For the Tilt, to extract an accurate metric, the elevation is needed to identify the actual deviation from the main view axis. However, because often the GPS sensor, that provides the altitude, is used indoors we do not have accurate enough measurement of the altitude. Therefore, we only consider "extreme" values (i.e. filming floor or ceiling). When we have such an "extreme" value (e.g. $|a_T| > |\pm 70°|$) the Roll

3 this angle is also called "head tilt", which is *not* the same as the camera tilt
4 mentioned as "Tilt" in the paper – meaning "head tilt"
5 for angles wider than 80°, the score starts to decrease since the video orientation switches from/to portrait/landscape

& Tilt score is zeroed ($S_t = 1$). Same as with S_s, S_t is also a negative metric, thus to calculate the final score we also use the complement to 1 of the value ($1 - S_t$).

Bitrate (V_b) is a metric to evaluate how the highest quality representation of a stream compares to the highest quality representations available from the rest of the streams. At a given time t the server receives streams of bitrates $\{Bh_1, Bh_2, \ldots, Bh_n\}$, for n available streams. The V_b score for a stream x with a maximum bitrate Bh_x is calculated as following:

Let $Bmin = min\{Bh_1, \ldots, Bh_n\}$ and $Bmax = max\{Bh_1, \ldots, Bh_n\}$:

$$V_b x = (Bh_x - Bmin)\frac{1}{Bmax - Bmin}$$

This means that if the currently available bitrate is the lowest (globally) it scores $V_b = 0$, if it is the the highest (globally) $V_b = 1$, or it is mapped respectively for values in between.

Image Quality (I_q) is a metric to indicate whether the video has "clean" (sharp/crisp) image. To calculate it, a frame from each stream is sampled every second, and the *blurriness* (I_b) is measured by taking the variance of the second derivative of its grayscale version [81], as shown:

Let I(m, n) the grayscale representation of an $M \times N$ frame and L(m, n) the second derivative operator [6]:

$$I_b = \sum_{m}^{M} \sum_{n}^{N} [|L(m,n)| - \bar{L}^2]$$

where $\bar{L}$ is the mean of absolute values:

$$\bar{L} = \frac{1}{NM} \sum_{m}^{M} \sum_{n}^{N} |L(m,n)|$$

Then, the measurements from all streams are normalized between 0 and 1, as previously, with 0 assigned to the lowest measurement and 1 to the highest. The final value of Image Quality for a stream is the complement to 1 of I_b, i.e. $I_q = 1 - I_b$.

Link Reliability (LR) is a composite metric consisting of an unordered pair $LR = \{\mu_{bitrate}, \sigma^2\}$, with $\mu_{bitrate}$ being the average of the highest bitrate available for the stream over time (denoted previously as Bh) and σ^2 the number of quality switches per minute (i.e. the times that the value of Bh changed). We did not invent $\mu_{bitrate}$ and σ^2, since they were already proposed to evaluate Link Reliability of HAS systems [77], but instead we removed aspects irrelevant to our use-case, such as the Buffer Level.

6 we used OpenCV that approximates it with a Laplacian mask – https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/

The $\mu_{bitrate}$ is a history-based metric and refers to the highest bitrate offered by the stream producer averaged over a period of time (i.e. for a number of segments) and is computed with the following equation:

$$\mu_{bitrate} = \frac{\sum_{i=0}^{N} Bh_i * t_i}{t_n}$$

where (for s_i = Segment i),

$i \in [0, N]$	Segment Index	t_i	Duration of s_i
t_n	Current time (duration)	Bh_i	Bitrate of s_i

The number of quality switches σ^2 is the amount of times that the maximum bitrate offered by the producer has changed and is calculated as following:

$$\sigma^2 = \sum_{i=0}^{N} g(s_i) \quad g(s_i) = \begin{cases} 1, & \text{if } i = 0 \\ 1, & \text{if } B_h i - 1 \neq Bh_i \\ 0, & \text{else} \end{cases}$$

The sensitivity of the LR metric can be optimized by adjusting the granularity of $\mu_{bitrate}$ and σ^2 when setting a window sample size. For example, for window size of z segments: $i \in [N - z, N]$, instead of the full history $i \in [0, N]$. Finally, because LR is used as a single metric for the ranking, we propose expressing the pair as $L_r = (\mu_{bitrate}/B_{max}) * \frac{1}{\sigma^2}$, with B_{max} being the highest bitrate (Bh) received from the source. We assume that LR is calculated on the server, but it can be implemented on the client, with the server providing the individual elements of the metric as timed metadata.

5.3.0.3 *Final Ranking*

For the ranking, the overall Score (S) of each stream is calculated as following:
$S = a_1(1 - S_s) + a_2(1 - S_t) + a_3 V_b + a_4 I_q + a_5 L_r$, with the weights $a_1 = a_2 = a_3 = a_4 = a_5 = 0.20$.
The selected weights of each score in this work are equal for two reasons. First, for our study to be consistent with previous work that uses similar ranking algorithms (e.g. MoViMash [96]). Secondly, because in our case, we manually annotated missing data (due to incompleteness of the dataset - detailed in Subsection 5.5.2), thus their accuracy is verified. In other implementations the weights can be adjusted according to the reliability of the data (and/or the sensitivity of the underlying hardware). For example, when the recording device is measuring the orientation using accelerometer, that is providing noisy measurements, Shakiness can have a lower weight. As a counter example, when the orientation is measured via gyroscope, that is subjectable to drift, Roll & Tilt can have a lower weight.

5.3.0.4 *Switching*

During playback a switch occurs at an interval between 6s and 12s. This interval range is based on generic cinematic guidelines, but can be refined for specific applications (e.g. shorter cuts for sport events, beat-matched cuts for concerts etc.). The precise scene duration - i.e. time of the switch (within this range), depends on the position of the stream on the ranking.

When a switch event occurs, the stream selection engine picks the highest ranked stream (other than the one currently playing) and checks the following conditions:

1. **Diversity**: The view selected before the currently active is skipped from this selection.

2. **Angle Difference**: If the viewing angle is similar to the one of the current stream we might have a "jump cut effect", or if it is close to $\pm 180^{\circ}$, might cause confusion [8].

3. **Diversity by Distance** (conditional): If none of the streams meets the Angle Difference criteria, we select the view with the longest distance from the currently selected (to avoid image-feature comparison between all streams [99] [96]).

5.4 IMPLEMENTATION CONSIDERATIONS

This is a first approach in streaming multi-source multi-stream UGC content. As a result, we examined the basic elements of such systems. In practice, modifications might be required according to the implementation specifications. Following are some proposals that can be evaluated according to the delivery chain.

5.4.1 *Generating Server-Side Representations*

The content consumer should have multiple representations (bitrates) available, to chose according to the policy of the HAS algorithm. In a traditional scenario, having multiple representations from the recorder to the server and perform typical DASH would suffice. However, because in UGC typically we encounter devices with constrained energy, networking and/or processing capabilities, we propose DASH with Scalable Video Coding (SVC) [100]. Alternatively, LiViU [117] can be used to select the optimal protocol for streaming the recording to the server in the highest quality, and then perform the transcoding on the server side for DASH delivery to the client [105]. Each approach has different advantages, notably by performing DASH with SVC, the users can opt for a higher latency with chances that they will receive better quality, while with LiViU the recorder has higher probability of staying in a high bitrate by switching protocols.

Regardless of the approach, the important is that any given moment a) the server is aware of the highest bitrate representation received by the recorder, in order to b) generate the respective representations for the client. Then we perform HAS from server to client.

Due to the variability of the highest available bitrate on the server (e.g. due to network throughput drops), the highest available bitrate to the client changes accordingly. However, for the duration of the streaming, the server must keep all of the representations listed available to the client. The reason for this is that if representations are to be excluded, the adaptive streaming descriptor/manifest must be updated to match the changes which typically happens at regular intervals [7] while in a UGC scenario with several recorders can potentially happen very frequently.

HLS does not mention anything relevant to dynamically changing available representations in the standard, but it is currently common practice for the server to create the manifest once and not make any changes thereafter – and for the client to load it and not check for updates.

MPEG-DASH allows modifications of the manifest only at the end of a period (or after a predefined update interval) [8], which means that every time there is a change in the available representations the period must conclude, the server must update the manifest, and the clients must load the new descriptor and reconfigure. This can be problematic since it will potentially happen very often (as the system scales), and at the beginning of each period players typically empty their buffers, thus causing constant global rebuffering events.

A naïve solution that could work for any client would be to mirror the low-bitrate representation in place of all the others, when the high-bitrate representation is not available to the server. This would technically work to typical HAS clients, without implementing any new functionalities. However, with this approach, the adaptation algorithms (that are responsible for selecting the most suitable representation) will effectively not work. The reason is that adaptation algorithms typically work by estimating the networking conditions, either by monitoring the buffer occupancy levels, or by asserting the available throughput. Therefore, when the mirroring of the representation is occurring, they will try to switch up on representations constantly, given that the traffic can be easily handled. Also, when the actual high bitrate representation becomes available again, the player might stall since the estimation was based on the "deceiving" mirrored requirements.

Alternatively, because there is already a communication channel between the server and the client for the timed metadata, it can be

7 https://stackoverflow.com/questions/33123631/adaptive-streaming-player-playlist-update-interval

8 https://dashif.org/docs/DASH-IF-IOP-v4.2-clean.htm#_Toc511040767

used to signal changes in the available representations. This approach is already indirectly implemented with the V_b metric that indicates the bitrate.

5.4.2 *Metrics Generation and Distribution*

Until now, we have provided recommendations concerning the type and method of logging the metrics, but we intentionally omitted exposing explicit details in the way they should be processed and distributed. The reasoning behind this decision is that several approaches exist, each with different advantages and disadvantages, that can be examined at this point, now that the system, metrics and policies are already explained. The approaches vary from the client only receiving the final Score (S) of each stream and select the one that has the highest, to receiving all the information available (e.g. Location, Orientation, Sampled Frames, Bitrates etc.) and calculate the metrics locally. The obvious advantage in the first case is the small computational and network overhead for the client, and in the second case is the flexibility on the client-side (e.g. to apply different policies).

We propose a hybrid approach in which all of the metrics are generated on the server, based on the video and timed metadata received from the recorder. Then, all the metrics (including the final Score), the LR components ($\mu_{bitrate}$, B_{max} and σ^2) and the geospatial information are available to the clients. This is similar to the approach proposed at Chapter 3 where timed metadata were used for control of multimedia applications. This technique maintains most of the advantages:

- **Low Throughput Requirements** (compared to generating metrics on the clients): The Sampled Frames that have a considerable size and are required for the I_q metric are not transmitted to the clients, nor processed there.
- **Flexibility in Metrics Generation** (compared to sending only the final Score to the clients): Because all of the metrics are transmitted to the client and the network and geospatial information is available, different metrics can be generated in several ways
  - **Different Algorithms for Existing Metrics**: custom algorithms for interpreting metrics can be implemented (e.g. modified Link Reliability).
  - **Introducing New Metrics**: testing algorithms using different metrics, can be achieved by updating or parameterizing only the software running on the client, without any modifications to the server (e.g. introducing a "mobility" metric based on the geospatial traces).

– **Per Device Adaptation**: by using information available to the device (e.g. signal strength, WiFi or GSM connection etc.) stream/representation selection policies can be adjusted accordingly, by calculating the Score only with regards to the supported bitrates.

This approach introduces some data redundancy, because all the metrics for all streams are transmitted (instead of just the final Score), but it can be useful in several cases. All the aforementioned flexibility is maintained, and processing power at the clients is conserved (which can reduce the energy consumed by the devices) because most of the heavy computing is performed on the server, that can be of importance when the clients are portable devices. Also, by generating the metrics on the server, new history-based, or other inter-source comparison metrics can be introduced.

Concerning the distribution of the aforementioned metrics, there can be two approaches; either having the metrics available in a per producer asset, or bundle the metrics from all the sources together, in a pre-defined frequency. The first approach offers the advantages of permitting to the client to fetch metrics only for a subset of the sources, that can offer some features like black-listing unreliable (e.g. classified as such due to frequent disconnect events) or low-quality (e.g. devices equipped with a low-fidelity camera) sources. Otherwise, bundling all the metrics together (e.g. every segment duration), reduces the number of HTTP requests, for the standard scenario where all the metrics are required, but creates issues when the producer-to-server delays are not uniform for all the sources.

5.5 EXPERIMENTAL SETUP

We simulate a live distribution scenario in which simultaneous recordings of a concert are streamed from the producer to the server and then to the client (consumer), where the view selection engine selects the most suitable view. The quality of the connections from the producers to the server and from the server to the client might vary, impacting the maximum bitrate offered for the uploaded streams.

We experiment with three scenarios. For the first scenario, we test our stream selection algorithm with perfect networking conditions to identify its theoretical impact.

Then, for the second scenario, we introduce network degradations on the link between the server and the client, in order to study how our stream selection works in HAS systems.

Finally, we introduce degradations on the link between the producers and the server. This scenario allows us to examine how the stream selection policy adapts to variable sources.

5.5.1 *Client*

To test the stream selection policies, we created a client-side application that supports playback of geotagged UGC videos. We used the SWAPUGC [85] open-source platform (presented in Section 4.3) to build our client. SWAPUGC supports the live MPEG-DASH profile for segmented video files, thus we can simulate a live delivery scenario.

Our client works as following: First, descriptor files of the recordings are parsed. The descriptor files contain information about the recording, such as its timing characteristics and the location of the spatial (location/orientation) and video (DASH) descriptor files. The video descriptor files contain information regarding the duration, available qualities and timing of the video streams. To finish the initialization of the client, the first location/orientation pairs are parsed.

When the playback starts, a default view is initially selected. As time progresses, callbacks are fired when new timed metadata arrives, in the following three cases:

- **Stream update**, when the bitrate or the availability of a recording changes.
- **Sensor update**, when a new location/orientation arrives.
- **Metric update**, when a new metric measurement arrives (e.g. Image Quality).

In our implementation, we have different sampling frequency for different timed metadata, in practice, the updates can be grouped to happen in one event (e.g. every second). The stream selection engine can use this information to evaluate whether it should change to a different stream, or remain on the currently active.

5.5.2 *Setup*

To evaluate the stream selection policies, a relevant dataset of recordings had to be used. We chose a dataset that contains recordings of the BBC philharmonic, from several cameras, both UGC and professionally recorded [11]. The dataset contains synchronization information of the video streams [9]. However, the spatial information available does not have any synchronization information and in some recordings is missing altogether, therefore we had to recreate it. In order to re-record the orientation information we used the open-source Spatiotemporal Video Navigation Recorder, and the location data was manually annotated.

9 Available on the SWAPUGC repository is also the parser used to extract the timing information of the videos and format the XML-based data of the dataset, to the SWAPUGC JSON-based format.

In total we used 7 recordings (from "Take #5" of the dataset). One of the views is a fixed studio camera at the back of the room, that is used as initial view at the start of the tests and then excluded from the pool of streams. Figure 31 shows a visualization of the cameras at the start of the Take (with the studio camera and the RoI highlighted in red).

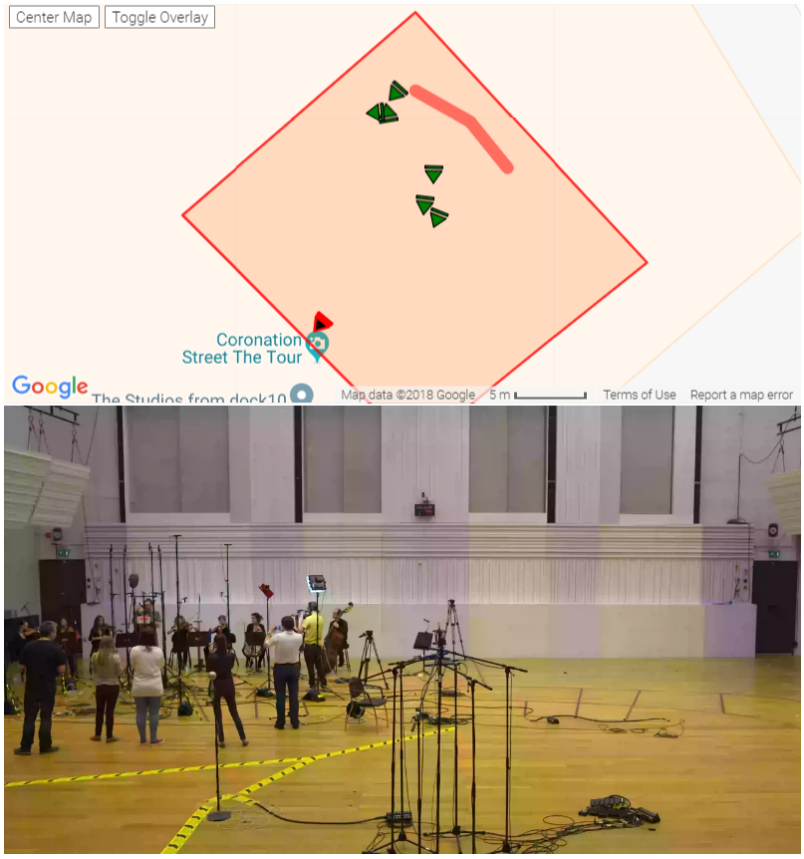


Figure 31: Visualization of available views

Finally, we used GPAC [62], to create segmented video files so they can be distributed in a manner that resembles a live scenario (with multiple qualities for adaptive streaming). We generated two representations for each stream, one for "high" and one for "low" quality, with bitrates 1800kbps and 400kbps respectively [10], and 2s-long ISO-BMFF segments. Initially we had an "intermediate" representation at 1000kbps but we screened them to experts and saw insignificant differences between it and the "high". The provided videos are not of good enough quality to generate a higher bitrate.

The DASH adaptation we implemented in SWAPUGC is a simple conservative buffer-based algorithm, that requests the highest bitrate, until the buffer contents drop below a predefined threshold, in which case switches to requesting a representation with lower bandwidth

10 we used the Youtube encoder settings as reference for the selected values: https://support.google.com/youtube/answer/2853702

requirements. Note that *the specific adaptation policy we designed has consideration for multiple content sources* (detailed in Appendix A.1).

The source code and all of the aforementioned metadata and descriptor files are available in the SWAPUGC repository, under a Creative Commons Attribution Non-Commercial Share Alike license (CC BY-NC-SA 4.0). The original audio and video files are hosted on the ICoSOLE project website, under a Creative Commons Attribution Non-Commercial Share Alike license (CC BY-NC-SA 4.0).

5.5.3 *Evaluation*

We want to examine whether our stream evaluation technique actually works and how our policy should behave in unstable network conditions, therefore we conducted a user study. For the user study we used the aforementioned recordings of Take 5, that yield a total overlap duration of 210s. In order to keep the duration of the video short and avoid bias due to fatigue of the participants, we showed 80s of each video simulating a selection policy.

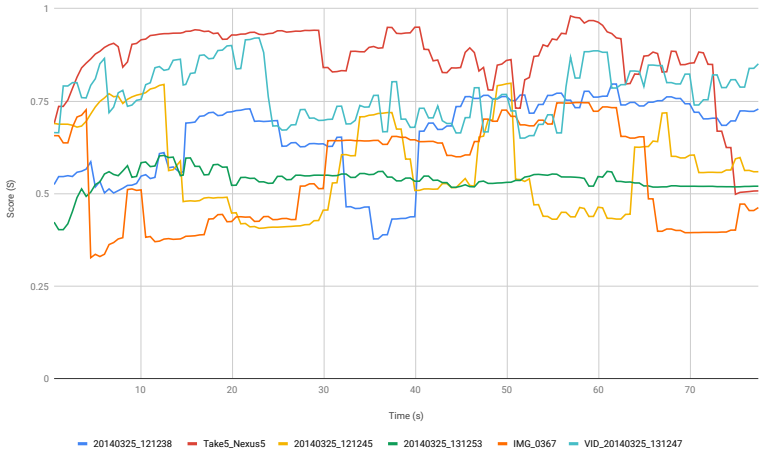


Figure 32: Scores over time for all the videos used

Due to the high variance in the quality of the videos and during each video (due to high mobility of the users), the stream selection is non-trivial. Figure 32 shows the Score, generated by our algorithm, for all the videos used in the experiment. We can observe that in this particular set a recording (with ID "Take5_Nexus5") scores constantly higher than the rest, however it can be selected only once every three changes to maintain the diversity criterion of the cinematic rules. Another observation on the recordings is that all of the videos have rapid changes in the ranking, because their individual criteria have frequent changes (e.g. when a view becomes suddenly very shaky, the overall Score for that stream is immediately reduced by 0.20).

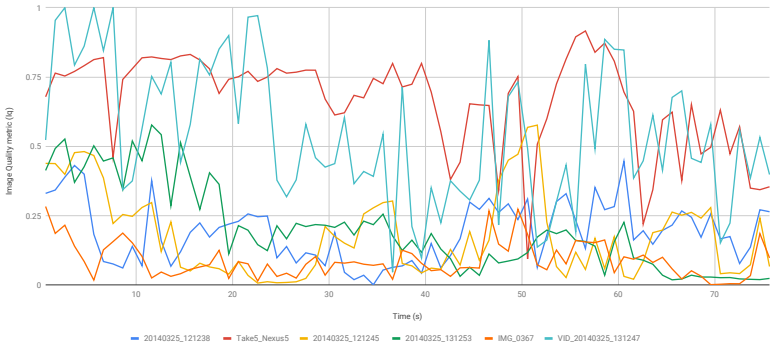


Figure 33: Image Quality metric (Iq) over time for all the videos used

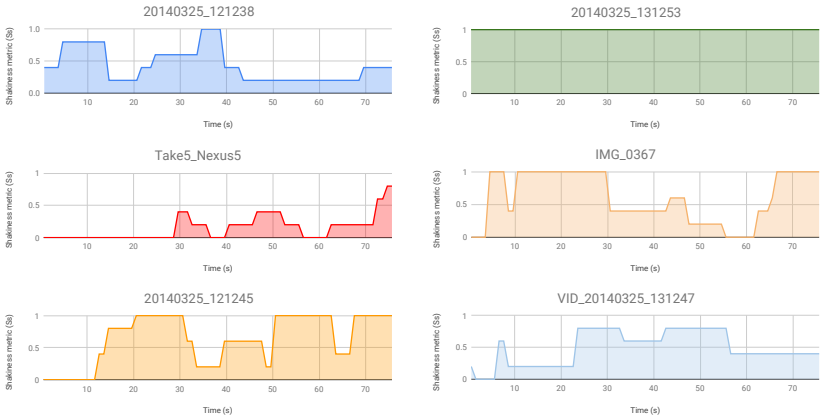


Figure 34: Shakiness metric (Ss) per video, over time for all the videos used

To break down the Score, we plotted the metrics that remain the same for all runs. Figure 33 shows the computed Iq metric (non-normalized) over time for the videos used. We can observe that overall there is high variance in this metric, even though some videos have consistently low or high values. For example the video with ID `20140325_121245` was mostly out of focus, thus its Iq score stays low, thus validating the metric.

Figure 34 shows plots of the Shakiness metric (Ss) for each video. Note that this is the metric value, not the complement to 1, thus higher value means worse quality. As mentioned before, the orientation sensors measurements of the videos were sometimes missing and even those that exists did not have any timing information, thus we had to manually annotate all the orientation-based metrics. This is why the Shakiness metric has discrete values (we used 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0). The same apply to the Roll & Tilt metric in Figure 35.

Regarding the videos, first, we generate two simulated runs to test the effectiveness of our stream selection policies under perfect networking conditions. Therefore, we generated a baseline, that follows only the cinematic criteria and the videos are in the highest bitrate.

Figure 35: Roll & Tilt metric (St) per video, over time for all the videos used

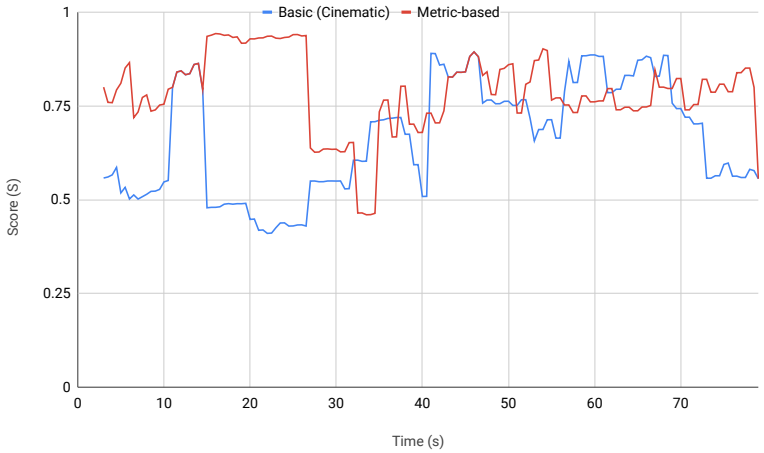
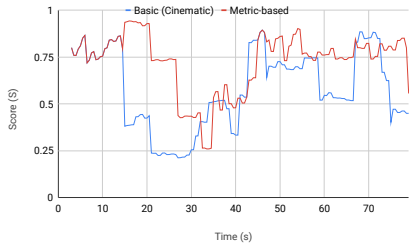
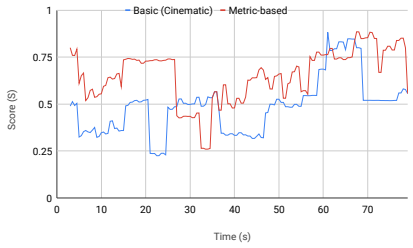


Figure 36: Scores over time for the streams selected by cinematic-only and proposed algorithms

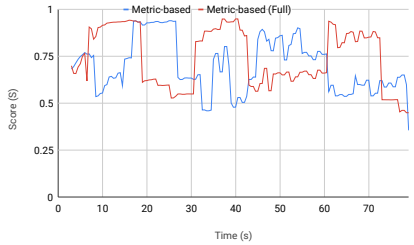
And an other run, in which both the cinematic criteria and our proposed stream selection policies are in effect, again without any network degradations. This pair aims to identify whether the elementary principles of our proposed approach are sound – i.e. if the MoViMash-based algorithm with the modifications we applied outperforms a simple cinematic-based stream selection algorithm. Since we have a "perfect" network, the V_b metric has a fixed value of 1 and L_r rapidly converges to 1. For comparison, the Cinematic-only selected streams average a Score of 0.67, and our stream selection algorithm 0.79; the evolution of the Score for selected streams over time is plotted on Figure 36. We can observe that even if the stream-selection is based on the full criteria, at times the semi-random algorithm outperforms our proposal (e.g. at the period between 57 and 68 seconds); this is expected to occasionally happen due to the view diversity criteria that exclude direct view repetition. This means that if there are only two high-ranked views in the pool, the selection will not be alternating



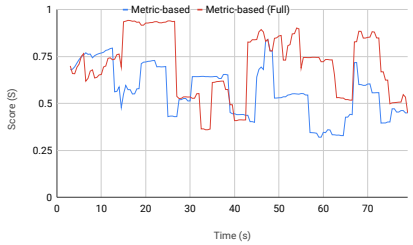
(a) Server-Client Degradations (Trace 1)



(b) Server-Client Degradations (Trace 2)



(c) Producer-Server Degradations (Trace 3)



(d) Producer-Server Degradations (Trace 4)

Figure 37: Scores over time for streams selected by respective algorithms, paired by network trace used

between them, instead the algorithm every third switch will select a different one - even if it scores significantly lower.

Then, in order to test the behaviour of our stream selection policies in realistic scenarios, we generated two more pairs of simulations in which there are network degradations from the server to the client – each pair with a different network trace (Fig. 37a and 37b). For these two simulation pairs, we implemented the basic DASH buffer-based adaptation algorithm described in the previous subsection. In the first simulation of each pair, again we use the basic, cinematic-based stream selection algorithm, while in the second we use our full metric-based stream selection algorithm, both using the same DASH policy. For this scenario, we would also like to identify whether our approach is compatible with "traditional" DASH servers, that do not support any signaling to the clients, therefore we keep the values of V_b and L_r fixed for all streams.

Finally, the last two generated pairs contain network degradations from the recorders to the server and all of the simulations use our stream-selection algorithm (Fig. 37c and 37d). Because, the HAS standards do not support such multi-source scenarios, if a high bitrate is not available anymore and instead is replaced by an upscaled lower quality video, the client cannot be aware of that (at least just by reading the `mpd` in case of MPEG-DASH). Therefore, for the first run of each pair we keep the traditional server functionalities (no signaling) and the client uses the multi-stream version of the buffer-based DASH algorithm (detailed in Appendix A.1). The second run of each

ID	# of runs	Adaptation	Degredations
0101	1	Basic (Cinematic)	None
0102	1	Metric-based (no V_b, L_r)	None
0201	2	Basic (Cinematic)	Server-Client
0202	2	Metric-based (no V_b, L_r)	Server-Client
0301	2	Metric-based (no V_b, L_r)	Recorder-Server
0302	2	Metric-based (Full)	Recorder-Server

Table 4: Simulation characteristics

pair, is aiming to identify whether the full version of our algorithm, with the proposed server signaling, improves the overall performance, so the client is aware of the actual available bitrates and the V_b and L_r metrics are put in effect.

To be able to identify the different runs of the same pair the naming conversion we follow is concatenating the ID with the run number. As an example the entry "020101" uses the same trace as "020201", "020102" with "020202" etc. The total of the 10 simulations used for the experiment is summed up on Table 4. Each run of the same pair, for the simulations with network degradations (i.e. with IDs 02X and 03X), uses the same network trace. In order to ensure that the results are not trace-specific, for the aforementioned simulations, with IDs 02X and 03X, we generated two different pairs, with two different traces. To sum up, the name of the entry is in the format XX_1-XX_2-XX_3, where XX_1 indicates the ID/number of the pair, XX_2 is the ID/number within the pair and XX_3 indicates the ID/number of the run/trace.

Regarding the network traces used, originally we wanted to use real-life measured traces, however, because we require both upstream (for the recorder to server link simulation) and downstream (for the server to client link simulation) throughput, we were not able to find a suitable dataset. The real-world recorded traces available, usually do not offer uplink measurements (which is required for the recorder-server link simulation), and when they do [89] [15] it is inaccurate. The reason for this is that the applications used to record the traces (for example, G-NetTrack Pro[11] [89]) record the uplink throughput utilized, not the actually available. In order to make an accurate recording the authors should be stressing the uplink for the duration of the recording, however they commonly put load on the downlink only, thus the recorded uplink throughput is usually extremely low (consisting only of the requests for the downloads). For that reason we generated the network traces for the simulation, using a custom python script that inputs a network quality variable representing the

11 http://www.gyokovsolutions.com/

(a) Server-Client Throughput (Trace 1)

(b) Server-Client Throughput (Trace 2)

(c) Producer-Server Throughput (Trace 3)

(d) Producer-Server Throughput (Trace 4)

Figure 38: Visualization of the representation supported by the simulated throughput, per trace over time

probability to support the high or low bitrate representation and outputs the network state (with granularity of 500ms).

On Figure 38 we plotted the generated traces. The plots show the supported bitrate representations over time (*not* actual throughput values), with "1" indicating low bitrate and "3" high. Subfigures 38a and 38b are for the simulations for which we alter the throughput on the server-client link and Subfigures 38c and 38d are for the recorder-server links, on per-recorder plots.

As for the audio, because we do not evaluate the audio quality or volume of the streams, we maintain the same audio source (from one camera) for all the simulations, so we will not alter the overall perception of the videos [13] [58]. In practice, this can be achieved by "late-binding" the selected audio source and exposing only that audio source on the HAS manifest. Audio has a low bitrate - comparing to the video files - thus it is not exposed as much to the network fluctuations.

5.6 USER STUDY

For the user study, all of the participants evaluated all of the simulated runs. The order of the pairs and the order within a pair was random, to avoid first-view (anchoring) bias and fatigue trends. Before the experiment started all of the users were told that they are about to watch different videos of a concert recorded from mobile phones and handheld cameras and we showed them two representative videos, one of "bad" quality and one of "good" quality, both manually edited by a video expert. The "good" quality was a first-pass edit of the videos, so it would be better than the simulated, but not so good that would create unrealistic expectations on the users.

The users answered a questionnaire aiming to measure the Quality of Experience in terms of View Diversity, Image Quality and Overall Pleasantness [99]. The statements (listed in Table 5) were presented after each simulation and the participants rated them in a five-point Likert-type scale according to the level of agreement or disagreement (1. "Completely Disagree" to 5. "Completely Agree"). The statements were mixed positive and negative, in order to discourage the participants from directing the scores [48] [47]. Prior to formulating the final version of the questionnaire we ran the experiment with 4 participants, using their feedback to verify the clarity of the statements and understanding of the experimental process. The results from these test runs are excluded from the presented findings, because based on the feedback we re-formulated the statements for the final version.

Statement	QoE Aspect
S1. "The image quality of the video was very bad"	Image Quality
S2. "It was difficult to see the details (faces, hands etc.) in the video"	Image Quality
S3. "I enjoyed watching the video"	Overall Pleasantness
S4. "I found watching the video tiring"	Overall Pleasantness
S5. "I found the selected views/cameras presenting accurately the action"	View Diversity
S6. "I found the view/camera changes too often"	View Diversity

Table 5: Evaluation statements and target QoE aspect

In total, we had 21 volunteers for the final user study, both male and female, with ages from 23 to 58 years old. Three of the volunteers had professional-level experience with video editing, and seven had "some" experience. All of the volunteers watched the simulated runs on the same 15" inch screen, with some using speakers and others headphones (according to their preferred habitual method).

5.6.1 *Results*

In this section, we analyze the responses of the participants. For the sake of clarity, on all of the following numbers and visualizations, the negative statements have reversed values (i.e. in all figures, the higher the score - the better). Note that we are able to do this because in this subjective study we are interested only in the relevant comparative values, otherwise it should be strongly advised against this practice, because positive statements have weaker effect than nega-

tives (explaining why 'S3' and 'S5' have globally lower score than the rest) [47].

video ID	S1	S2	S3	S4	S5	S6	AVG
010101	3.18	2.86	2.09	2.68	1.90	2.31	2.50
010201	4.09	4.04	3.59	3.90	3.77	4.00	3.90
020101	1.95	2.09	1.95	2.77	2.31	2.86	2.32
020201	3.27	3.31	3.09	3.77	3.50	3.72	3.44
020102	1.50	1.68	1.86	2.27	2.95	3.13	2.23
020202	2.54	2.59	2.54	3.09	3.09	3.50	2.89
030101	2.36	2.36	2.36	2.86	2.95	3.31	2.70
030201	3.81	3.77	3.50	3.86	3.72	3.68	3.72
030102	2.40	2.63	2.13	2.22	2.36	2.86	2.43
030202	3.63	3.63	3.04	3.45	3.13	3.22	3.35

Table 6: Mean statement scores per generated videos (simulated runs)

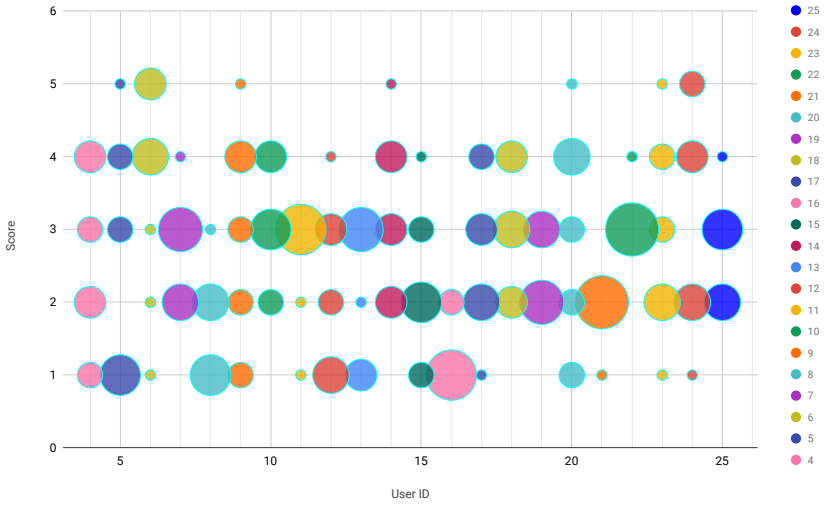


Figure 39: Scores given for S3, for all videos, by user ID

On Table 6 we can see the mean scores for all statements and the average score, per video ID (maintaining the pair order). The scores have an average standard deviation (STD) of 0.99. The high STD value is expected, because the aesthetic standards of each user differ, thus some users might have a better overall opinion on the videos than others. This is visible in Figure 39 where we plotted the responses of all the users for the Statement S3, for all videos. We can observe that most respondents (11 out of 21) used only two or three consecutive values from the 5-point scale for their evaluations (i.e. [1,3], [2,4],

or [3,5]), and only 5 used the full 5-point scale. The same trend is observed for the rest of the questions as well.

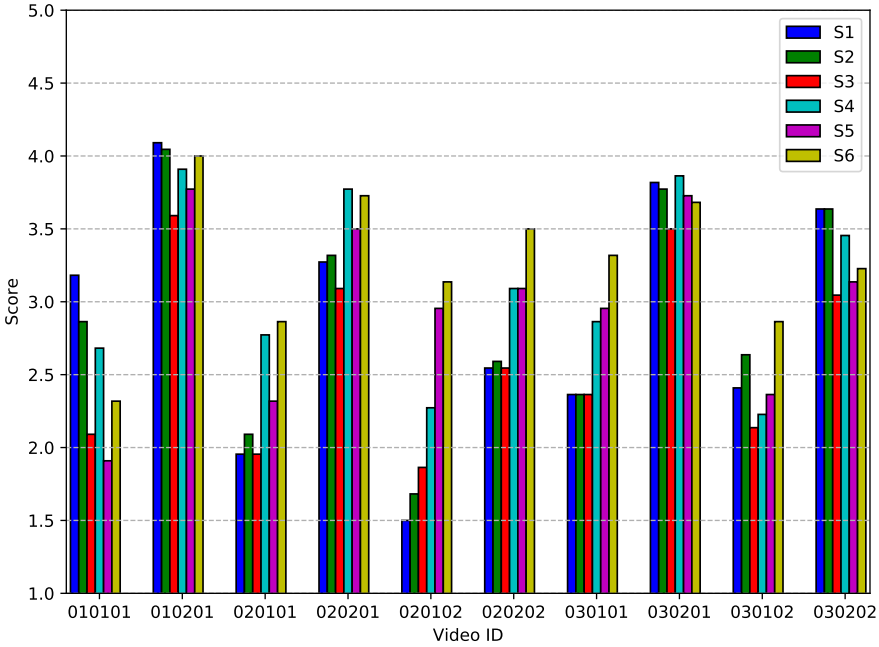


Figure 40: Average scores per statement, by video ID

The statement scores from Table 6 are also plotted in Figure 40. We can observe, that overall, the simulations that use our algorithm (0102X and 0202X), outperform those that rely only on the cinematic criteria (0101X and 0201X).

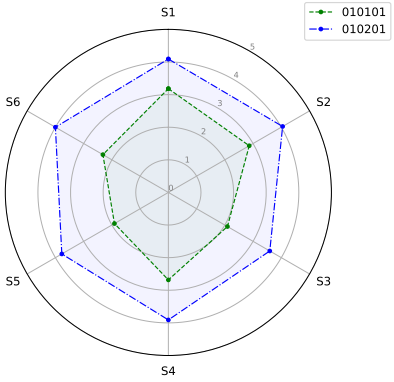
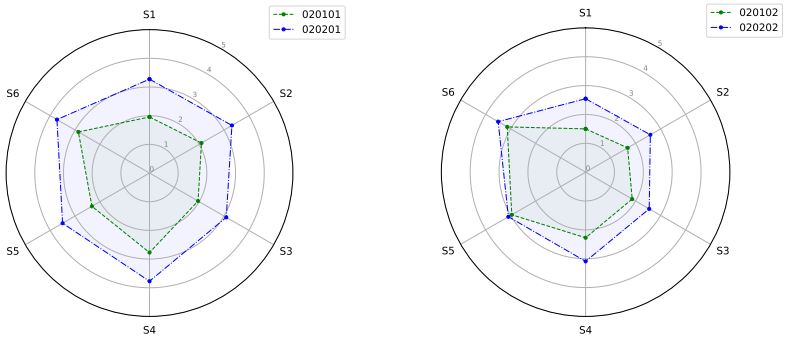


Figure 41: Average scores for Cinematic and Metric-based adaptation (perfect network)

More specifically, for the first pair, that targets measuring the elementary effectiveness of our view selection algorithm, with perfect networking conditions, the baseline cinematic-based solution scores an average of 2.50, and our algorithm 3.90. The specific scores for each statement, for pair-wise comparison are plotted in Figure 41.



(a) Scores for first pair/trace

(b) Scores for second pair/trace

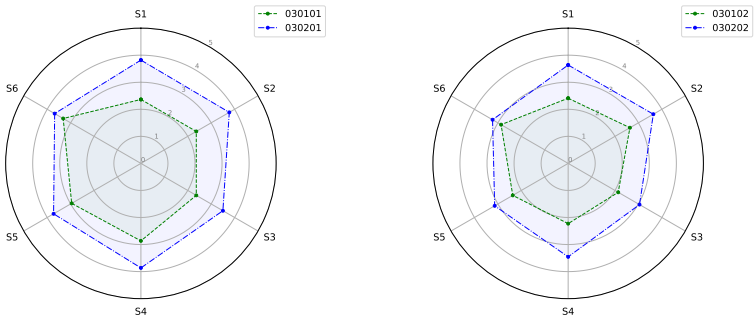
Figure 42: Average scores for Cinematic and Metric-based adaptation (server-client degradations)

The results confirm our assumption that our algorithm can select views with better-perceived Image Quality (S1, S2) and improves the Overall Pleasantness (S3, S4) of the videos. Furthermore, even though both algorithms use the same cinematic criteria the users evaluated our approach on view selection as more relevant (S5). We attribute this improvement on the effect of including the rest of the metrics (thus the overall camera quality) in the camera selection process at a switch. This phenomenon also impacts the perceived frequency of camera changes (S6), even though for the duration of the videos the amount of switches is similar (10 for 010101 and 8 for 010201). Both algorithms have the same boundaries for the scene length, but in the baseline video the selected value is random, while in our proposal it is affected by the overall metrics score. Because our algorithm is apt in evaluating the stream quality, it results to "good quality" scenes being longer than the "bad quality" scenes.

Seeing that our proposed algorithm outperforms the baseline in ideal networks, we next compare the runs that simulate link degradations. Figure 42 shows the scores for all statements of the baseline cinematic algorithm against our proposed metric-based, for two different network traces. Both approaches use the same HAS buffered-based adaptation algorithm.

Again, the metric-based algorithm scores higher in all of the examined criteria, however we can observe that the impact depends on the trace. Comparing to the 3.90 average score for the first test, the scores for the second were 3.44 and 2.89. This fluctuation is justified by the overall drop of quality for the final output, therefore even if the algorithm selects a "good" view, the perceived quality is lower due to the encoding artefacts of the low-bitrate representations. This assumption is backed by the comments of the participants, that often expressed difficulty to identify differences and accurately evaluate the statements when the quality of the final output was low. Note

that the participants where unaware of which runs had simulated varying throughput or not.



(a) Scores for first pair/trace (b) Scores for second pair/trace

Figure 43: Average scores for Metric-based and Full adaptation (recorder-server degradations)

The final test was to evaluate the impact of signaling metrics from the server to the client, therefore in the first scenario we have the metric-based algorithm used so far, and in the second its full version, including the V_b and L_r metrics, that require signaling to work as designed. Both clients use the same HAS buffer-based adaptation algorithm (the multiple sources version described Appendix A.1). For these test runs, we introduced network degradations on the recorder-server network. The results are plotted in Figure 43.

The introduction of the server-client signaling, thus the implementation of the V_b and L_r metrics, improves the performance of the algorithm by almost 1 point on average. Using these metrics, the client avoids switching to a stream from an "unstable" recorder, minimizing the possibility of having to stream a low-quality representation. Signaling can be used when the currently selected stream becomes unavailable. When the client is aware that a link issue occurs (i.e. the high-bitrate representation is not available), it can re-evaluate the ranking with the new bitrate and decide to haste the switch (if the current scene duration exceeds the minimum limit set by the cinematic criteria).

Finally, regarding the feedback from the participants, most of the users stated "shakiness" as being the most annoying issue with the videos, followed by the lack of steady cameras, cameras changing often and fast direction, camera not filming the RoI et.al. Also, some participants stated that depending on the target application, they would use such algorithms to follow an event. Example target applications mentioned were "live streaming of the event" and "content preview/overview", on the other end, applications like "learning/training (musical instrument)" and "promotional videos" were deemed unsuitable for such techniques. All of the comments from the participants are listed in the Appendix A.3.

In this Chapter, we proposed a stream selection mechanism for multi-view multi-quality UGC videos with accompanying geospatial data. We presented the individual elements (cinematic and quality metrics) and the details of the respective algorithms. We conducted a user study to measure the effectiveness of our base algorithm and how it adapts to various networking infrastructures.

From the results of the user study we can conclude that our proposed metrics-based algorithm offers improved QoE compared to just following cinematic criteria. It can also work in platforms with simple servers (i.e. that do not support signalling), by excluding the V_b and L_r metrics. However, to reach its full potential, it requires server signalling.

Regarding the future work, the two fronts that we would like to work after the end of this Thesis are the Delivery and the User Experience (UX). On the latter, we would like to study the aural aspects of the platform. First, in our tests, since we do not have a proposed audio evaluation algorithm, the client gets the audio signal from only one camera. We would like to examine lightweight audio evaluation techniques to integrate in our proposal to select the optimal audio source. The second audio-related issue we would like to address, is finding a suitable, lightweight audio analysis algorithm that will give us information on the event (i.e. if it is dialog-based, music-oriented etc.) in order to potentially integrate respective directives to the cinematic criteria, for example switching on tempo if the event is a concert [118].

Similarly, using the camera information, on top of the RoI, we can identify the class of the event. For example, in sport events the viewing angle of the UGC cameras tends to vary a lot (and in bursts), because most of the cameras tend to follow the action - that is more quickly paced than other events. This can be used to adjust the cinematic criteria accordingly.

As for the Delivery, the streaming adaptation policy is an aspect deserving further study. In our implementation, we used a buffer-based adaptation policy, however it would be useful to examine how different policies affect the implementations of our proposal. For example, throughput-based adaptation policies are based on estimating the current throughput capabilities, therefore streams with bitrates that cannot be supported by the estimated available throughput can be excluded in the filtering phase.

Also, because our system uses streams from varying sources, at varying qualities and networking conditions, it would be interesting to make a comparison with algorithms used in Tiled Video Streaming [120]. When very big videos (e.g. 360°, 4k etc.) are streamed, are often split in "tiles" that are available in different bitrates and the

client selects the most appropriate according to the viewport (tile) and connection (bitrate). Therefore, there are system-design similarities between adaptation for UGC content streaming and Tiled Video Streaming and future contributions to either field, might be applicable to the other.

6

BUFFER MANAGEMENT FOR SYNCHRONOUS AND LOW-LATENCY PLAYBACK OF MULTI-STREAM USER GENERATED CONTENT

So far we have studied an architecture for delivery of video with timed metadata and we examined this architecture by addressing scenarios and respective methods for streaming and presentation of extended AV streams produced from common and specialized scenarios for live and on-demand delivery. A significant portion of these scenarios concerns gathering and distributing User-Generated Content (UGC) – as in Chapters 4 and 5.

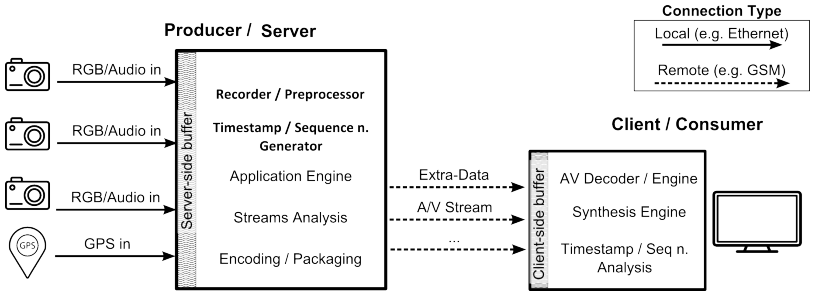
We have mentioned that in UGC scenarios, there are differences comparing to the traditional distribution systems. In the "traditional" scenario, the recording sensors are directly connected to the server (e.g. Kinect-based control in Chapter 3), but in the "UGC" scenario the sensor data is recorded (and potentially processed) locally and send to the remote server over an unstable connection. Because of this uncertainty on the content production side special consideration should be granted to improve the overall experience on the client. For example, in the previous chapter, we introduced signaling from the server to the clients on the status of the recorders, novel adaptation algorithms, and generation of different bitrates both at the recorder and on the server side.

We show our proposed architecture for "traditional" multi-stream systems in Figure 44a, and the "UGC" version illustrated in Figure 44b.

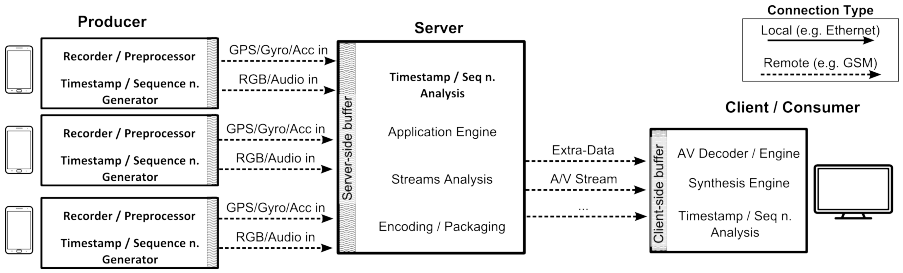
Content from different sources might arrive with different delays - depending on the respective remote connection, potential local data processing etc. Even from the same source, depending on the connection quality to the server, a wide spectrum of delays can be observed when bursts of (delayed) data arrive as the user reconnects.

This diversity of delays, in combination with an unreliable upload mechanism (e.g. over UDP), can also cause out-of-order delivery of frames. Sources for such incidents can be when the user is uploading the content using a mobile broadband connection and a switch from a high to low latency cell occurs (i.e. handover[1]), or when a stream source is a multi-hop sensor network gateway etc. In the latter case, the end-to-end delay can be in the order of minutes *per hop* for low-power wireless sensor networks (e.g. slotted IEEE 802.15.4) [93]. Also, due to the non-uniform nature of recording devices, different types and/or qualities of data (both AV and sensor) might exist. Addition-

1 also mentioned in literature as handoff



(a) Traditional Server-Client schema



(b) UGC Producer-Server-Client schema

Figure 44: Extended AV stream Architectures Overview

ally, in contrast to AV streams where losing frames on delivery can create issues in multiple other frames during playback (because they might be required by the decoder), this is not always the case with the timed metadata.

All of the above issues create challenges in delivering the content streams with low-latency and consuming them in a synchronous manner. We focus on tackling those challenges with client-side solutions, because in UGC scenarios there is no control on the production end, other than extracting information on the status of the recorders and signal it to the clients (as we did previously, in Chapter 5).

To address the challenges, we propose a client-side buffer management scheme for multiple streams. The priority of our scheme is to *achieve synchronization between the streams, while reducing the number of buffering events*[2], and has provision for two consumption modes, that facilitate different secondary goals:

- **Synchronized**, emphasizing on eliminating *frame losses*.
- **Low-latency**, that reduces *buffering duration*.

The *Synchronized* playback can be used in cases where fidelity during playback is paramount (e.g. when watching a performance). *Low-latency* is targeting scenarios where reducing delay is prioritized over lossless playback (e.g. for security systems monitoring). We also

2 as "buffering event" or "rebuffering event" we define the incident that the buffer of the client is empty, thus the player is in "stalling" state; and "buffering duration" is the duration that the player stays in "stalling" state

propose solutions for applications that require switching or compromising between the two modes (e.g. support both quick response and thorough investigation of an emergency).

In the next section we review the state of the art on UGC-compatible synchronization algorithms. Then, in Section 6.2, we demonstrate our proposal for synchronous and low-latency consumption of extended AV streams. Finally, we close this Chapter with a discussion in Section 6.3.

6.1 STATE OF THE ART

There are studies and surveys on multimedia content synchronization [16] [34] [56] [50] [28]. We considered all of the previous work as foundation for our work. Due to the criteria mentioned in the introduction, we filtered irrelevant methods. More specifically, regarding the synchronization type, *intra-stream* techniques are not in the scope of this Thesis, since it has been thoroughly studied [56], thus we assume accurate media sampling of our streams. Also, *inter-destination* synchronization is not relevant, because we consider the streams to be consumed from a single device. Our work is suitable for *inter-stream* synchronization, dealing with streams of various characteristics from different sources.

On top of defining the synchronization type of our focus, we used the classification method utilized in such studies [16] [56] to identify specific techniques relevant to our work. Regarding the synchronization system location, we exclude most *preventive* and *reactive* "Source Control" techniques, like "deadline-based transmission scheduling" or "adjusting transmission rate" that require feedback from the clients to the server in order to change transmission scheduling and rate respectively [91] [90] [57]. This feedback can be used to adjust transmission [90] or playback [57] of the content. This is a point-to-point design which can not be done in a live UGC scenario where the server does not produce the content and there are several clients consuming it. However, such techniques can be used for on-demand delivery, high latency systems - by implementing a server-side buffering scheme, or hybrid systems in which the content comes both from sources directly connected to the server and UGC (e.g. project Cognitus [3]).

Alternatively, if server side buffering is used, or for on-demand delivery, synchronization can be achieved by packaging all of the available streams in a single container [26]. This solution, unless used in systems with a bounded number of users/streams, is not suitable for the server-to-consumer part of an UGC platform (which is the part that we focus on this chapter), since it would require packaging content from several producers in a single extremely large file. However it can provide frame-accurate synchronization, with minimal effort,

3 www.cognitus-h2020.eu

on the producer-to-server part, especially if a widespread format like `mp4` is used.

Since we do not have control on the production of the content, we only use *basic* "Source Control" techniques in our proposal, that concerns providing synchronization information (i.e. timestamps, frame numbers and source identifiers) already present in UGC systems.

Our "Receiver Control" solution, uses *buffering techniques* and/or *reactive skips* to offer synchronization. Such methods have been used as part of systems in a variety of works, from modeling event-based synchronization [1], to complementing agent-based adaptive systems [68] and protocols for time, error and synchronization control [119], among others. The event-based synchronization model [1] does not suffice to cover UGC scenarios because it is based on multiplexing "download" and "display" commands in the video stream. This approach assumes that at the time a video frame is sent, the representation time for the respective extra data frame is known and it is ahead enough on the timeline to allow transmission and processing of the video frame, request the extra data frame, transmit and process of that extra data frame, and all that before the display event is fired. The agent-based system [68] is also not suitable for our targeting UGC systems because it assumes a fixed and known frame rate on the production end, and requires intensive bidirectional communication between the server and the clients.

Our work is not competing with any of the aforementioned relevant work, instead it aims to complement these approaches when applied in multi-source multi-stream scenarios. In many use cases it can not be used alone to offer solutions for all the synchronization problems that occur. Such an example case is when simultaneous recordings do not share a common clock and have to be synchronized on the server, prior to distributing to the clients. For that example, an extra (inter-bundle) synchronization layer must be applied server-side (e.g. using audio feature extraction [104]) and our proposal can be used on the client.

Finally, regarding the server-side processing, even though the previous example where a post-production resynchronization layer is in place adds significant overhead on the server, it has applications when common time reference is not feasible and/or the content consumer can tolerate large end-to-end delays. In contrast, the techniques we propose minimize that overhead, in order to also be applicable in delay-sensitive scenarios.

6.2 CLIENT-SIDE BUFFERING SCHEME

Because we were unable to find a generic timed metadata dataset with accompanying video, large enough to have statistical signifi-

cance, we generated a test data set with example timed metadata and video streams.

Each stream file has frame information for a total duration (T_{Dur}) of 100s with a 30Hz frame rate frequency (F). We selected this frame rate because it is close to the frame rate of the Kinect for joint coordinates and to common frame rates for video. Listing 1 shows an example sample describing a frame, containing the Frame Number (`FRN`), Arrival (`T_arr`), Display/Composition (`T_disp`) time, and `Delay`.

The video frames generated (as the one shown in the Listing 1) have a fixed 100ms delay. We selected a fixed video delay since this can be the case when an encoder is used. The timed metadata frames have a delay varying from D_{min}= 200ms to D_{max}= 3200ms, with an average of D_{avg}= 1700ms. The delay range was selected to fit values close to those that we experimentally observed in the Kinect-based control application demonstrated in Chapter 3 and of relevant literature [119], on the D_{min}end, to delays that are observed in single-hop wireless networks using low-energy communication protocols like `IEEE 802.15.4` [93] or `BLE` [23], on the D_{max} end.

In total, for the simulations, we generated 1000 extra-data stream files, out of which 500 with a uniform delay distribution and 500 with a normal distribution. We opted for the generated samples to follow two different, but common, distributions so we can examine any potential effects from the form of the samples on our proposed techniques. A visualization of the data set frame delays is shown in Figure 45. For future work we plan on testing our proposal against other common distributions (e.g. Poisson) and actual data sets.

```
1 "FRN":544,
2 "T_disp":18133.33,
3 "T_arr":18233.33,
4 "Delay":100
```

Listing 1: Frame sample contents

To accommodate a broader spectrum of application scenarios we include cases such as a distributed processing environment (e.g. cloud computing), or multiple sources over a network (e.g. wireless sensor network), thus in the simulations we allowed out-of-order frame delivery. That is, at time t_{curr}, a frame FRN_n with $Delay = D_n$ will arrive before a frame FRN_{n-x} (where $x \in \mathbb{Z}_{>0}$), if $D_{n-x} + (x * \frac{1}{F}) < t_{curr}$. Because of the out-of-order arrival of frames, in this chapter when we refer to a buffer containing such frames, we count its size (B_{size}) or length (B_{len}) in consecutive frames, starting from the next frame scheduled for consumption. When the buffer size or length is measured by counting all the frames (for size), or the difference between the first and the last timestamp (for length), we refer to *fragmented* buffer size (B_{size_frag}) and length (B_{len_frag}) respec-

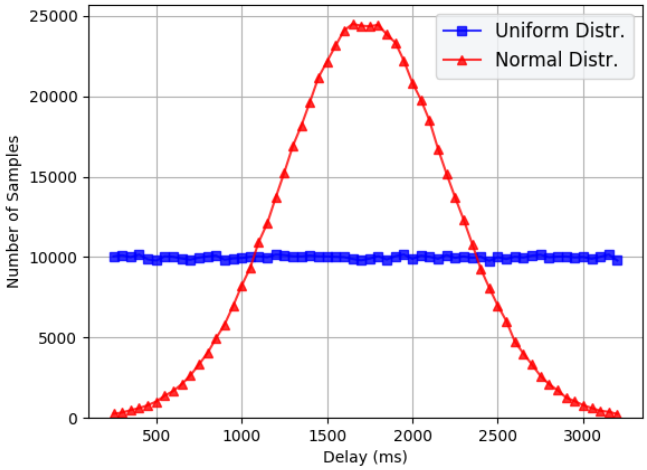


Figure 45: Delay distribution of generated frames

tively. More information on fragmented and non-fragmented buffer size/length, on Appendix A.4.

6.2.1 *Synchronized Playback of Streams*

We used the aforementioned data set to feed a multimedia client simulator [4]. For the simulations we are using the video stream (that has a fixed delay), as reference clock.

The client has two buffers, one for the video (B_V) and one for the extra-data (B_M). Upon the arrival of a frame at `T_arr` (relative to the simulation start time), it is pushed in the respective buffer, and it is removed when the playback timeline reaches `T_disp`. To achieve synchronization between the two streams, their playback pauses whenever either buffer is empty (i.e. $B_{Msize} = 0$ *or* $B_{Vsize} = 0$), causing a *Rebuffering* event [107].

In order to avoid rebuffering during playback, that would disrupt the user experience, we set an initial buffer length requirement (`Binit`). Initial playback commences when the duration of the frames inside each buffer (B_{len}) equals to the respective `Binit` threshold. Thus, the initial buffering duration is not fixed; instead, it depends on the time required to obtain the defined length of continuous data in the buffer, which in turn is affected by the stream delay characteristics. Because the generated video frames have a fixed delay, we set a constant $B_{Vinit} = 100$ms and we ran 40s-long simulations for $B_{Minit} = \{100, 300, ..., 1500\}$ms. The B_{Vinit} value can be any constant, for example, it can be set to match the requirements of a codec using prediction techniques (e.g. for B-frames).

4 source code of the simulator and dataset generator available at: https://github.com/emmanouil/Buffer-Tests

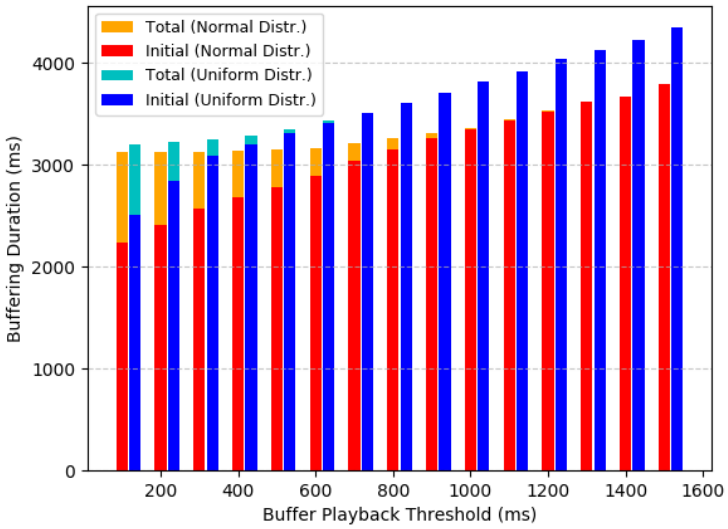


Figure 46: Buffering duration to `Binit` value

Figure 46 shows a stacked bar chart with the resulting buffering duration averages (in ms - y axis), in respect to the B_{Minit} threshold (in ms - x axis). The dark colored bars indicate the initial buffering duration, while the corresponding visible light colored part indicates the duration of rebuffering events. We can observe that for the uniform distributed samples the time required to reach B_{Minit} is longer than the respective for the normal distribution. However, in both cases, when the initial buffering duration exceeds the maximum sample delay (i.e. $T_{B_{Minit}} > D_{max}$), no rebuffering events occur.

As a result, if the D_{max} is known in advance (e.g. from the system specifications), it can be utilized to avoid rebuffering events. The buffering policy can be adjusted, so instead of setting the initial buffering criterion (`Binit`) to the desired content duration for the buffered stream, to be set as fixed buffering time. If this initial buffering duration is set to be equal or greater than the known D_{max}, no rebuffering events should occur, unless the D_{max} value increases later.

USES WITH ASYNCHRONOUS PLAYBACK This technique can also be used in systems for which synchronous and asynchronous consumption modes are required, such as environmental monitoring deployments. As an example, the user might be remotely monitoring an area, though a low-latency video stream coupled with low-power (high-delay) temperature sensor measurements. By default, on the consumption-end, low-latency is desired, for immediate response on emergencies (e.g. a fire). In order to achieve that, both the video and the temperature frames are rendered upon arrival - thus not synchronously, since they have different delays.

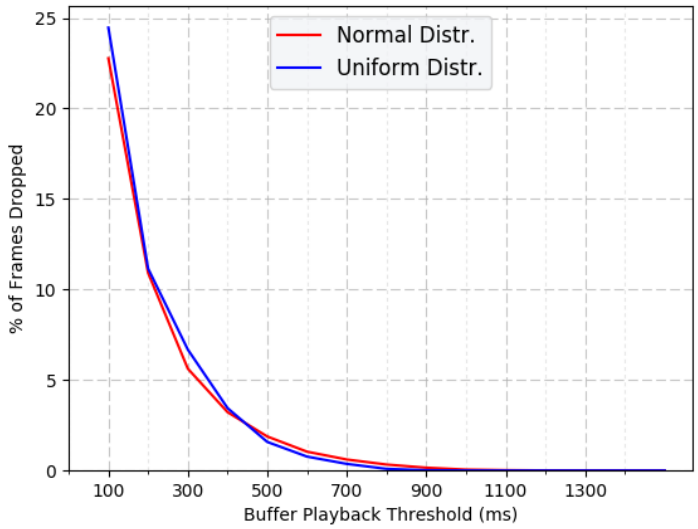


Figure 47: Percentage of dropped (to consumed) frames per `Binit` value

However, if the purpose is to identify the origin or spread of the fire, synchronous playback of the streams is crucial. The user can seek to the desired time and consume the content in a synchronous manner. If the time difference (Δ_t) between the current and rewind point is less than D_{max}, a buffering event will occur for a duration equal to D_{max} minus Δ_t.

6.2.2 *Proposal For Low-Latency Playback of Streams*

In the synchronous consumption mode, the impact on initial buffering can be significant, adding an overall delay to the stream playback. This can cause synchronized playback to be unsuitable for use-cases where it is preferred to consume the content in a low-latency mode. Some systems, such as the Kinect-based synthesis described in Chpater 3, are designed is such way that can afford to skip some timed metadata frames. For these systems, to achieve a tradeoff between reducing the stream playback latency (i.e. due to initial buffering) and increasing the system bootstrap time, frames that arrive late, can be dropped.

In that scenario, initial buffering can be adjusted either to a buffer threshold B_{init}, or to a maximum acceptable delay – defined by the specification of the system (instead of D_{max}). Afterwards, when playback starts, delayed frames are discarded, eliminating the rebuffering events.

We ran simulations using the same dataset and parameters as before, but dropping frames instead of rebuffering, and in Figure 47 we display the percentage of frames dropped to total frames consumed

for different B_{init} values. By accepting an example frame drop rate of 7% (which is less than a frame per second), we can set `Binit` to 300 ms, thus reducing the initial buffering duration for the normal distribution sample from 3200 ms to 2600 ms, or for an even lower drop rate of 3% to 2800 ms.

This buffering technique can be applied to low-latency systems, or systems that support both low-latency and synchronous playback. In the latter case, if the player is in low-latency mode and the user wants to switch to synchronous, a single rebuffering event occurs for a duration equal to D_{max} minus the initial buffering. For the reverse scenario, that the switch is from synchronous to low-latency, the player time-shifts to the desired latency value and the late frames are dropped thereafter.

MULTIPLE STREAMS - MULTIPLE DELAYS There are cases that multiple qualities of the same stream can arrive, depending on the underlying hardware and type of data. For example, in the UGC video with geospatial matadata scenario (like in Chapters 4 and 5), some devices are equipped with gyroscope sensors that directly provide high-quality orientation values, while others calculate them using measurements from the magnetic field sensor and accelerometer. Even though the processing delay is negligible, the resulting values must be smoothed (normalized) over time to match in accuracy those that would have been provided via a composite gyroscope sensor [73]. Similarly, in a stream of 3D object reconstructions using a smartphone [71], the fidelity of the 3D models is proportional to the number of samples used for the reconstruction, therefore to the production delay. In such cases, multiple streams of the same data can be provided, and the quality (accuracy / fidelity) of each data stream is proportional to its delay.

For such cases, application of our proposal for *synchronous* consumption is straightforward, since to produce streams of predefined qualities, the processing requirements are also predefined; therefore the D_{max} for each extra-data stream is known in advance. However, there can be an issue if the user, while consuming a stream, would like to switch to a stream of a different quality/delay. If the selected stream has lower quality (and therefore lower delay), the switching is seamless. However, when changing to a higher delay stream, triggering a rebuffering will degrade the over User Experience. In that case, on top of switching the stream, we can change the buffer behavior as well, from *synchronous* to *low-latency*, thus avoiding the rebuffering event.

Similarly, a client rendering frames only in low-latency mode, with multiple streams available, can switch to a different stream. If the client is playing a low-latency stream and switches to one with higher

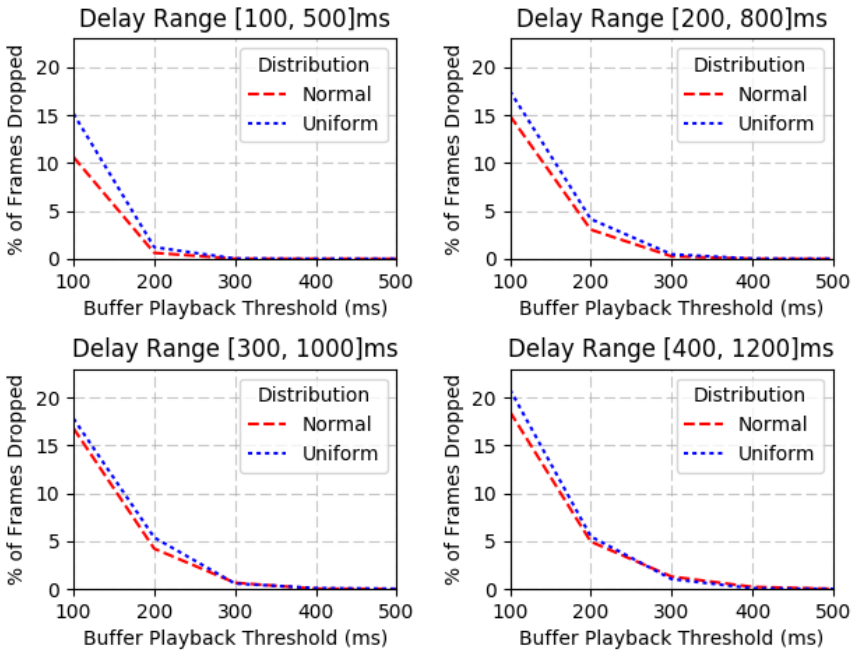


Figure 48: Percentage of dropped (to consumed) frames per `Binit`, for various delays

latency, the percentage of dropped to consumed frames will increase, while in the oppose case, it will decrease.

Example drops rates for streams with various delays are shown in Figure 48. We can observe that, if for example the `Binit` was set at 300 ms, even in the highest delay range of [400, 1200] ms, the frame losses are below 3% of the total number of frames, and even if we are not aware of the initial buffering time that would be required for a different stream (e.g.[100, 500] ms), when the switching to the low-latency stream occurs, the percentage of frames dropped will be lower.

6.3 DISCUSSION

We demonstrated that by signaling the maximum production delay of a stream (D_{max}) to the client, the buffer size can be adjusted to eliminate rebuffering events, thus offering a smooth synchronized playback experience.

For systems that have low-latency requirements, the policy can be adjusted to a maximum initial buffering duration (or to a minimum required buffer length - `Binit`) and dropping late frames thereafter, instead of rebuffering. The same principle can be applied for seamlessly switching between streams with different characteristics.

We also examined how those two modes can co-exist in a platform that supports synchronized and low-latency streaming.

Our work can be applied to applications that utilize extended AV streams. Especially in a UGC streaming scenario, that feedback to the content producer is not supported (therefore the quality of the transmitted extra data is not known to the producer), using the aforementioned methods can lead to lower end-to-end delay, while maintaining synchronized and low-latency consumption capabilities. This applies to examples given previously (and to our applications from previous chapters), where filtering data over time results to higher quality output and our buffering scheme can be applied.

Our work can be extended, to include delay estimation schemes, in case that signaling the D_{max} is not feasible and synchronous consumption is required. There has already been research on this field [103] and some of the algorithms can be used in conjunction with our proposals.

Some specific aspects of delay estimation are close to the application scenarios that we have already presented in this manuscript. For example, previous work for consuming Adaptive Video streams over HTTP in mobile networks [70] uses GPS error metrics and GSM network quality to assert whether the user moves from an outdoors (good service) to an indoors (dubious service) setting. Those measurements can be recorded on the producer end in the UGC with geospatial data scenario (as in the applications demonstrated in Chapters 4 and 5), thus used for signaling potential upcoming changes in the D_{max}.

Finally, it would be of interest as future work to propose the D_{max} signaling to be integrated in relevant standards, part of the current streaming ecosystem, such as MPEG-DASH, IETF RTP etc. [121] [34].

Part IV

CONCLUSION AND DELIVERABLES

7 CONCLUSION

In this Thesis, we studied the requirements, apparatus and environment to propose techniques for enhancing video applications through timed metadata. In the Introduction we mentioned that we were looking to answer questions like the following:

- *What kind of added value does the timed metadata offer to AV applications?*
  We identified that the timed metadata with respect to the AV systems can:
  - Enable new use-case scenaria.
  - Improve the performance of current (and future) AV systems by optimizing client-side delivery and server-side processing/repackaging of AV content .
  - Allow implementation of different end-user applications using the same metadata.
- *Where and how should the timed metadata be processed?*
  Even though there is not a clear answer to this question, we recommend to gather everything on the recorder-side. Then, process the metadata (including synchronization provisions) on the server. In most cases, the extra data are light (compared to the AV data), for such cases transmit all of them. Finally, the rendering and presentation aspects should be handled on the client.
- *Are there common building blocks / techniques that can be studied and then applied to different application scenaria?*
  Following the aforementioned guidelines, the resulting elementary building blocks are the same for most extended AV applications, adjusted to the respective specifications. Techniques (e.g. buffering, adaptation etc.) can be applied to applications that use timed metadata of specific properties.

As the main motivation for this Thesis was to examine the feasibility of distributing all types of timed metadata in everyday applications and their potential usefulness, we begun by identifying the characteristics, challenges and classes of such systems. Despite the variety of cases encountered, we outlined an architecture pattern that can facilitate recording, distribution and consumption of various data types, for several use cases. Based on this architecture and the aforementioned work, we conducted the studies and implemented relevant applications thereafter. These contributions can be used as a

reference for identification of relevant future work, or as foundations for possible extensions.

The main body of our work is split in two parts, the first on using timed metadata to extend the capabilities of audiovisual systems and the second to improve content delivery of such systems. For the first part, we analyzed the specifications and requirements for two categories of applications. First, for multimedia control, using inputs from a specialized device and second for spatiotemporal video navigation, using inputs from common devices.

In both cases, we built relevant systems, in order to demonstrate the features enabled from the timed metadata and in the process we got further understanding on the limitations and extensions of extended AV systems. For example, in Chapter 3, we studied audiovisual synthesis applications and we demonstrated that accompanying timed metadata can increase flexibility and reduce client-side computational complexity, with the producer maintaining control over the final output. We also identified possible issues with such applications, like potential asynchronies between frames or modalities — a challenge that we address in Chapter 6.

Similarly, in Chapter 4, we studied how timed metadata can be used to enable spatiotemporal video navigation. Again, we showed extensions, like enabling visualization and navigation of UGC videos, and we identified shortcomings, e.g. marker congestion occurring in content-crowded regions.

For the second part, we proposed techniques for optimizing video delivery by using timed metadata. The timed metadata are used to facilitate the selection of the relevant videos and the selection of appropriate bitrates of these videos. Then, we demonstrated methods for buffer management for accommodating delivery of timed metadata in different scenarios. These methods can be applied to the use cases we already studied and examined via proof of concept platforms, and to use cases that we considered but we did not have the chance to thoroughly examine, like sensor networks.

7.1 Outlook and potential application fields

Our work has the potential to be integrated in existing video-centric platforms that utilize timed metadata (e.g. GeoUGV), or to seed extensions in fields that use timed metadata with video, but have no current considerations for delivery optimizations. An example of the latter would be telemedicine, that Kinect is used for rehabilitation and recovery monitoring [80], but without regards for systems aspects.

To further elaborate, while staying on the telemedicine example, for the initial part of the Thesis, we have argued that such applications can benefit from adopting our proposed classification and architecture design proposals, and at the same time, by using international

standards, can lead e-healthcare to a more interoperable and modular direction.

Then, we appeal to the concept of liberating the systems for future applications by selecting the appropriate level of abstraction on the transmitted modalities. Distributed applications of telemedicine can benefit in this aspect by transmitting and archiving data and synchronization information provided from the cameras and other recording devices (BCIs, Kinect etc.). An example benefit would be facilitating implementation of new client-side features, retroactively validating new diagnostic techniques and/or facilitating collaboration between different applications, including platforms from different vendors. Finally, due to the diversity of modalities and environments that is prevalent in the telemedicine domain, distribution issues occur (out-of-order delivery, frame drops etc.) and our recommendations from Chapter 6 can be applied to ease their effect on the receiving end.

Telemedicine is just one example of a field that our work can contribute. Even simple media players for multiview video can be enhanced through the appropriate distribution and utilization of timed metadata. For example, in Chapter 4 we showed how to facilitate the selection of the appropriate video and the navigation to the relevant point in the timeline. Also, by applying the principles of Chapter 5, the media player can improve the User Experience by using the timed metadata to automatically select the appropriate view and quality.

7.2 RESEARCH PERSPECTIVES AND FUTURE WORK

Concerning the future work on our proposals, there can be exploratory research in breadth and depth of the relevant systems as well as extensions on the interoperability with academic and industry practices.

To increase the impact area of our proposals, we could examine different environments that we did not have the chance to do so during the PhD. One such example mentioned in several instances throughout the manuscript is the sensor networks. Even though we take provisions according to the specifications of different sensor networks, and we do consider borderline cases, if we had the time we would have extended our work by studying real life deployments of different configurations (wired / wireless, main / battery / harvesting powered, personal / local / wide area etc.).

The sensor network setup is especially interesting to be studied for live scenarios, because of the traditionally large delays that occur on the data collection. The nodes tend to have large (to save power) and varying (according to their depth in the network) delays therefore are not commonly applied to scenarios with live consumption of the content. Even though we did study such applications (e.g. in Chapter 6), a large-scale experimental work would be very interesting. A proposed example of such work would be for a smart cities deployment

where a plethora of cameras and smaller sensor networks are interconnected and the data gathered is used in different scenaria (energy monitoring, traffic optimization etc.) each with different priorities and specifications. It is also noteworthy that often in such systems (in contrast to the systems we studied), the AV is not the main content, and occasionally can be missing completely.

To increase the impact effect of our proposals, experiments with different configurations of the studied scenarios would be beneficial. For example, in the buffering proposals, if time was not of the essence, we would have studied more delay ranges and distributions.

Additionally, specific configurations could advance the quality of the obtained results. Such an example is with the UGC view and stream selection work, that we could try different coefficients to the metrics used for the final score, according to the underlying videos used. For example, we could study how the coefficients of the metrics and the cinematic criteria can be adapted according to the video content type (sport event, concert etc.). Moreover, since to our knowledge, our work is the first that considers both sensor measurements and network evaluation criteria for stream and quality selection, there is potential for improving our contributions by fine tuning (or altering) the metrics. Due its novelty, we would focus especially at the Link Reliability metric, since it is newly introduced and there is no other work on UGC stream selection with a similar recorder-to-server connection evaluation metric.

Another research aspect of the UGC stream work that deserves examination would be the application of our proposals in immersive scenarios. Having already tools and techniques to identify the quality of the streams and their spatiotemporal relevance can potentially enable the integration of these UGC streams in immersive environments. For example, the entanglement of UGC videos in virtual worlds (or VR exploration of virtual representations) can be a possibility, if the proper QoE studies are conducted.

8 DELIVERABLES

8.1 PUBLICATIONS

List of papers published during the Thesis in peer-reviewed venues.

[00] Emmanouil Potetsianakis and Jean Le Feuvre.*"View and Quality Selection For Live UGC Video Streaming"*, ACM Transactions on Multimedia Computing, Communication, and Applications (TOMM). *[submitted]*

[85] Emmanouil Potetsianakis and Jean Le Feuvre.*"SWAPUGC: Software for Adaptive Playback of Geotagged UGC"*, MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.

[88] Emmanouil Potetsianakis and Jean Le Feuvre. *"Buffer Management for Synchronous and Low-Latency Playback of Multi-Stream User-Generated Content"*, IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR), Miami, FL, USA, April 2018.

[84] Emmanouil Potetsianakis. *"Streaming and Presentation Architectures for Extended Video Streams"*, Adjunct Publication of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video (TVX), 2017, pp. 129–132.

[87] Emmanouil Potetsianakis and Jean Le Feuvre. *"Streaming of Kinect Data for Interactive In-Browser Audio Performances"*. Proceedings of the 11th Audio Mostly Conference. ACM, 2016.

8.2 SOFTWARE

This section lists the open source software developed during this Thesis, mentioned on this manuscript. Note that it contains complete software implementations of our own design; contributions to external repositories are not included.

8.2.1 *Kinect-based Control*

Kinect-based Control is a set of tools for recording and playback of Kinect data for control of multimedia applications using MS Kinect as an input. The repository contains the recorder and an example client application for modular audio synthesis with accompanying visualizations (as presented in Chapter 3).

CURRENT STATUS :
Inactive. Project concluded in 2017.

LANGUAGES/PLATFORMS USED :
Recorder: C++ (MS Kinect SDK, libgpac, ffmpeg) – Client: Javascript (p5.js)

REPOSITORIES :
Main repository: https://github.com/emmanouil/MOOOK/

8.2.2 *Spatiotemporal Video Navigation*

Spatiotemporal Video Navigation is a platform for selecting the appropriate video, at the desired time, according to geospatial criteria of the recording (as presented in Chapter 4). It consists of three parts:

- **Recorder**: An Android app for recording videos and the accompanying geospatial data.
- **Parser**: A Python module for parsing and formatting the recordings.
- **Client**: A web-based client/viewer for the recordings.

CURRENT STATUS :
Maintained, but not under active development.

LANGUAGES/PLATFORMS USED :
Recorder: Java (Android SDK) – Parser: Python – Client: Javascript, Google Maps JS API – Example videos: MP4Box, ffmpeg

REPOSITORIES :
Client (main repository): https://github.com/emmanouil/Spatiotemporal-Navigation
Recorder: https://git.io/SNR

8.2.3 *SWAPUGC*

Soft**W**are for **A**daptive **P**layback of **U**ser-**G**enerated **C**ontent is a set of tools used for implementing platforms for playback of spatially annotated videos. It supports playback of multiple concurrent streams, with multiple representations (bitrates) per stream, and adaptation algorithms (metadata and/or bandwidth-based).

CURRENT STATUS :
Under development. Currently changing the format to a set of libraries and the parser scripts.

LANGUAGES/PLATFORMS USED :
Parsers (for SNR and ICoSOLE recordings): Python – Blur estimation: Python (OpenCV) – Demo Client and Browser Scripts: Javascript, Google Maps JS API – Example videos: MP4Box, ffmpeg

REPOSITORIES :
Main repository: https://github.com/emmanouil/SWAPUGC/
Recorder: same as Spatiotemporal Video Navigation
Demo (hosted by ACM MMSys): https://acmmmsys.github.io/2018-SWAPUGC
Dataset (provided by ICoSOLE project): http://icosole.lab.vrt.be/

8.2.4 *Buffer Simulator*

A simulator for conducting simulations of buffering techniques. It contains a frame generator (to simulate the streams), the buffer/network simulator and visualization scripts.

CURRENT STATUS :
Inactive. Project concluded in 2017.

LANGUAGES/PLATFORMS USED :
Data generator and Simulator: Javascript (Nodejs) – Analysis and Visualization scripts: Python (Numpy, Matplotlib)

REPOSITORIES :
Main repository: https://github.com/emmanouil/Buffer-Tests

BIBLIOGRAPHY

[1] Hironobu Abe, Hiroshi Shigeno, and Ken-ichi Okada. "Media synchronization method for video hypermedia application based on extended event model." In: *Multimedia and Expo, 2006 IEEE International Conference on*. IEEE. 2006, pp. 1289–1292.

[2] Paul Adenot, Chris Wilson, and Chris Rogers. "Web Audio API." In: *W3C, October* 10 (2013).

[3] Demosthenes Akoumianakis, George Vellis, Ioannis Milolidakis, Dimitrios Kotsalis, and Chrisoula Alexandraki. "Distributed Collective Practices in Collaborative Music Performance." In: *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*. ACM. 2008, pp. 368–375.

[4] A. Al-Nuaimi, B. Cizmeci, F. Schweiger, R. Katz, S. Taifour, E. Steinbach, and M. Fahrmair. "ConCor+: Robust and confident video synchronization using consensus-based Cross-correlation." In: *2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP)*. 2012, pp. 83–88. DOI: 10.1109/MMSP.2012.6343420.

[5] Usman Ali and Muhammad Tariq Mahmood. "Analysis of Blur Measure Operators for Single Image Blur Segmentation." In: *Applied Sciences* 8.5 (2018), p. 807.

[6] Silviu Apostu, Anas Al-Nuaimi, Eckehard Steinbach, Michael Fahrmair, Xiaohang Song, and Andreas Möller. "Towards the design of an intuitive multi-view video navigation interface based on spatial information." In: *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*. ACM. 2013, pp. 103–112.

[7] *Apple HTTP Live Streaming (HLS)*. https://developer.apple.com/documentation/http_live_streaming. Accessed: 2019-01-01.

[8] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. "Automatic Editing of Footage from Multiple Social Cameras." In: *ACM Trans. Graph.* 33.4 (July 2014), 81:1–81:11. ISSN: 0730-0301. DOI: 10.1145/2601097.2601198. URL: http://doi.acm.org/10.1145/2601097.2601198.

[9] Artec. *Artec Eva 3D Scanner Specification*. https://www.artec3d.com/portable-3d-scanners/artec-eva. Accessed: 2019-08-02.

[10] Sakire Arslan Ay, Roger Zimmermann, and Seon Ho Kim. "Viewable scene modeling for geospatial video search." In: *Proceedings of the 16th ACM international conference on Multimedia*. ACM. 2008, pp. 309–318.

[11] Werner Bailer, Chris Pike, Rik Bauwens, Reinhard Grandl, Mike Matton, and Marcus Thaler. "Multi-sensor concert recording dataset including professional and user-generated content." In: *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM. 2015, pp. 201–206.

[12] Victor Baños-Gonzalez, M Shahwaiz Afaqui, Elena Lopez-Aguilera, and Eduard Garcia-Villegas. "IEEE 802.11 ah: A technology to face the IoT challenge." In: *Sensors* 16.11 (2016), p. 1960.

[13] John G Beerends and Frank E De Caluwe. "The influence of video quality on perceived audio quality and vice versa." In: *Journal of the Audio Engineering Society* 47.5 (1999), pp. 355–362.

[14] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. "The effects of loss and latency on user performance in unreal tournament 2003®." In: *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. ACM. 2004, pp. 144–151.

[15] Ayub Bokani, Mahbub Hassan, Salil S Kanhere, Jun Yao, and Garson Zhong. "Comprehensive mobile bandwidth traces from vehicular networks." In: *Proceedings of the 7th International Conference on Multimedia Systems*. ACM. 2016, p. 44.

[16] Fernando Boronat, Jaime Lloret, and Miguel García. "Multimedia Group and Inter-Stream Synchronization Techniques: A Comparative Study." In: *Information Systems* 34.1 (2009), pp. 108–131.

[17] Tim Bray. *The Javascript Object Notation (JSON) Data Interchange Format*. Tech. rep. 2017.

[18] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. *Extensible markup language (XML) 1.0*. 2008.

[19] Axel Carlier, Lilian Calvet, Duong Trung Dung Nguyen, Wei Tsang Ooi, Pierre Gurdjos, and Vincent Charvillat. "3D interest maps from simultaneous video recordings." In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 577–586.

[20] Anna Llagostera Casanovas and Andrea Cavallaro. "Audio-visual events for multi-camera synchronization." In: *Multimedia Tools and Applications* 74.4 (2015), pp. 1317–1340.

[21] Huihui Chen, Bin Guo, Zhiwen Yu, and Qi Han. "Toward real-time and cooperative mobile visual sensing and sharing." In: *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*. IEEE. 2016, pp. 1–9.

[22] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor C Leung. "Body area networks: A survey." In: *Mobile networks and applications* 16.2 (2011), pp. 171–193.

[23] Keuchul Cho, Woojin Park, Moonki Hong, Gisu Park, Wooseong Cho, Jihoon Seo, and Kijun Han. "Analysis of latency performance of Bluetooth low energy (BLE) networks." In: *Sensors* 15.1 (2015), pp. 59–78.

[24] Anthony Churnside, Chris Pike, and Max Leonard. "Musical movements—gesture based audio interfaces." In: *Audio Engineering Society Convention 131*. Audio Engineering Society. 2011.

[25] Cyril Concolato, Jean Le Feuvre, and Romain Bouqueau. "Usages of DASH for Rich Media Services." In: *Proceedings of the Second Annual ACM Conference on Multimedia Systems*. MMSys '11. San Jose, CA, USA: ACM, 2011, pp. 265–270. ISBN: 978-1-4503-0518-1. DOI: 10.1145/1943552.1943587. URL: http://doi.acm.org/10.1145/1943552.1943587.

[26] Cyril Concolato, Jean Le Feuvre, and Emmanouil Potetsianakis. "Synchronized Delivery of 3D Scenes With Audio and Video." In: *Proceedings of the 20th International Conference on 3D Web Technology*. ACM. 2015, pp. 245–248. DOI: 10.1145/2775292.2775324. URL: http://doi.acm.org/10.1145/2775292.2775324.

[27] Cyril Concolato and Emmanouil Potetsianakis. "In-Browser XML Document Streaming." In: *XML Prague* (2015), pp. 197–205.

[28] Ricardo Mendes Costa Segundo and Celso Alberto Saibel Santos. "Systematic Review of Multiple Contents Synchronization in Interactive Television Scenario." In: *ISRN Communications and Networking* 2014 (2014).

[29] Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. "The blur effect: perception and estimation with a new no-reference perceptual blur metric." In: *Human vision and electronic imaging XII*. Vol. 6492. International Society for Optics and Photonics. 2007, p. 64920I.

[30] Francesco Cricri, Igor DD Curcio, Sujeet Mate, Kostadin Dabov, and Moncef Gabbouj. "Sensor-based analysis of user generated video for multi-camera video remixing." In: *International Conference on Multimedia Modeling*. Springer. 2012, pp. 255–265.

[31] Francesco Cricri, Kostadin Dabov, Igor DD Curcio, Sujeet Mate, and Moncef Gabbouj. "Multimodal extraction of events and of information about the recording activity in user generated videos." In: *Multimedia tools and applications* 70.1 (2014), pp. 119–158.

[32] Eduardo Cuervo, Alec Wolman, Landon P Cox, Kiron Lebeck, Ali Razeen, Stefan Saroiu, and Madanlal Musuvathi. "Kahawai: High-Quality Mobile Gaming Using GPU Offload." In: *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM. 2015, pp. 121–135.

[33] *DVB Companion Screens and Streams (DVB-CSS) part 2: Content ID and Media Synchronisation*. Tech. rep. TS 103 286-2 v1.2.1, 2017.

[34] M Oskar van Deventer, Hans Stokking, Matt Hammond, Jean Le Feuvre, and Pablo Cesar. "Standards for Multi-Stream and Multi-Device Media Synchronization." In: *IEEE Communications Magazine* 54.3 (2016), pp. 16–21.

[35] M Oskar van Deventer, Jean-Claude Dufourd, Sejin Oh, Seong Yong Lim, Youngkwon Lim, Krishna Chandramouli, and Rob Koenen. "Media orchestration between streams and devices via new MPEG timed metadata." In: *SMPTE Motion Imaging Journal* 127.10 (2018), pp. 32–38.

[36] Matthew Edwards and Richard Green. "Low-latency filtering of kinect skeleton data for video game control." In: *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*. ACM. 2014, pp. 190–195.

[37] John Eidson and Kang Lee. "IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems." In: *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*. Ieee. 2002, pp. 98–105.

[38] Jeremy Elson and Kay Römer. "Wireless sensor networks: A new regime for time synchronization." In: *ACM SIGCOMM Computer Communication Review* 33.1 (2003), pp. 149–154.

[39] Steve Faulkner, Arron Eicholz, Travis Leithead, Alex Danilo, and Sangwhan Moon. *HTML 5.2*. 2017.

[40] *Flickr*. www.flickr.com. Accessed: 2018-04-04.

[41] Jules Françoise, Norbert Schnell, and Frédéric Bevilacqua. "A multimodal probabilistic model for gesture–based control of sound synthesis." In: *Proceedings of the 21st ACM international conference on Multimedia*. ACM. 2013, pp. 705–708.

[42] Gianluca Giorgolo and Frans AJ Verstraten. "Perception of 'speech-and-gesture' integration." In: *AVSP*. Citeseer. 2008, pp. 31–36.

[43] *Google Maps*. https://maps.google.com. Accessed: 2018-04-04.

[44] Michael Gurevich. "JamSpace: A Networked Real-Time Collaborative Music Environment." In: *CHI'06 extended abstracts on Human factors in computing systems*. ACM. 2006, pp. 821–826.

[45] Abdelkrim Hadjidj, Marion Souil, Abdelmadjid Bouabdallah, Yacine Challal, and Henry Owen. "Wireless sensor networks for rehabilitation applications: Challenges and opportunities." In: *Journal of Network and Computer Applications* 36.1 (2013), pp. 1–15.

[46] Jia Hao, Guanfeng Wang, Beomjoo Seo, and Roger Zimmermann. "Keyframe presentation for browsing of user-generated videos on map interfaces." In: *Proceedings of the 19th ACM international conference on Multimedia*. ACM. 2011, pp. 1013–1016.

[47] James Hartley. "Some thoughts on Likert-type scales." In: *International Journal of Clinical and Health Psychology* 14.1 (2014).

[48] James Hartley and Lucy R. Betts. "Four layouts and a finding: the effects of changes in the order of the verbal labels and numerical values on Likert-type scales." In: *International Journal of Social Research Methodology* 13.1 (2010), pp. 17–27. DOI: 10.1080/13645570802648077.

[49] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. "A buffer-based approach to rate adaptation: Evidence from a large video streaming service." In: *ACM SIGCOMM Computer Communication Review* 44.4 (2015), pp. 187–198.

[50] Zixia Huang, Klara Nahrstedt, and Ralf Steinmetz. "Evolution of Temporal Multimedia Synchronization Principles: A Historical Viewpoint." In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9.1s (2013), p. 34.

[51] ISO/IEC International Standard 14496-10:2003. *Information Technology — Coding of Audio-visual Objects — Part 10: Advanced Video Coding*.

[52] ISO/IEC International Standard 14496-12:2008. *Information Technology — Coding of Audio-visual Objects — Part 10: Part 12: ISO Base Media File Format*.

[53] ISO/IEC International Standard 14496-14:2018. *Information technology – Coding of audio-visual objects – Part 14: MP4 file format*.

[54] Adobe Inc. "HTTP Dynamic Streaming Specification - Version 3.0." In: (2013).

[55] *Instagram*. www.instagram.com. Accessed: 2018-04-04.

[56] Yutaka Ishibashi and Shuji Tasaka. "A Comparative Survey of Synchronization Algorithms for Continuous Media in Network Environments." In: *Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on*. IEEE. 2000, pp. 337–348.

[57] Yutaka Ishibashi, Shuji Tasaka, and Yoshiro Tachibana. "Media synchronization and causality control for distributed multimedia applications." In: *IEICE Transactions on Communications* 84.3 (2001), pp. 667–677.

[58] Satu Jumisko-Pyykkö. ""I would like to see the subtitles and the face or at least hear the voice": Effects of picture ratio and audio–video bitrate ratio on perception of quality in mobile television." In: *Multimedia Tools and Applications* 36.1-2 (2008), pp. 167–184.

[59] Jarkko Juslin. *Audio Synthesis and Effects Processing with Web Browser in Open Web*. 2016.

[60] Theodoros Karagkioules, Cyril Concolato, Dimitrios Tsilimantos, and Stefan Valentin. "A Comparative Case Study of HTTP Adaptive Streaming Algorithms in Mobile Networks." In: *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. NOSSDAV'17. New York, NY, USA: ACM, 2017, pp. 1–6. ISBN: 978-1-4503-5003-7. DOI: 10.1145/3083165.3083170. URL: http://doi.acm.org/10.1145/3083165.3083170.

[61] Sang Kyun Kim and Marius Preda. "MPEG-V international standard and the internet of things." In: *SIT 2014-FIT 2014: International Conference on Software Intelligence Technologies and Applications & International Conference on Frontiers of Internet of Things*. IET. 2014, pp. 166–173.

[62] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. "GPAC: Open Source Multimedia Framework." In: *Proceedings of the 15th ACM International Conference on Multimedia*. MM '07. Augsburg, Germany: ACM, 2007, pp. 1009–1012. ISBN: 978-1-59593-702-5. DOI: 10.1145/1291233.1291452. URL: http://doi.acm.org/10.1145/1291233.1291452.

[63] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. "Toward a practical perceptual video quality metric." In: *The Netflix Tech Blog* 6 (2016).

[64] J. Lichtenauer, M. Valstar, J. Shen, and M. Pantic. "Cost-Effective Solution to Synchronized Audio-Visual Capture Using Multiple Sensors." In: *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*. 2009, pp. 324–329. DOI: 10.1109/AVSS.2009.92.

[65] Mark A Livingston, Jay Sebastian, Zhuming Ai, and Jonathan W Decker. *Performance measurements for the Microsoft Kinect skeleton*. Tech. rep. NAVAL RESEARCH LAB WASHINGTON DC, 2012.

[66] Ying Lu, Hien To, Abdullah Alfarrarjeh, Seon Ho Kim, Yifang Yin, Roger Zimmermann, and Cyrus Shahabi. "GeoUGV: User-generated mobile video dataset with fine granularity spatial metadata." In: *Proceedings of the 7th International Conference on Multimedia Systems*. ACM. 2016, p. 43.

[67] He Ma, Sakire Arslan Ay, Roger Zimmermann, and Seon Ho Kim. "Large-scale geo-tagged video indexing and queries." In: *GeoInformatica* 18.4 (2014), pp. 671–697.

[68] Sunilkumar S Manvi and P Venkataram. "An agent based synchronization scheme for multimedia applications." In: *Journal of Systems and Software* 79.5 (2006), pp. 701–713.

[69] Sujeet Mate and Igor DD Curcio. "Automatic Video Remixing Systems." In: *IEEE Communications Magazine* 55.1 (2017), pp. 180–187.

[70] Sami Mekki, Theodoros Karagkioules, and Stefan Valentin. "HTTP adaptive streaming with indoors-outdoors detection in mobile networks." In: *arXiv preprint arXiv:1705.08809* (2017).

[71] Natan Micheletti, Jim H Chandler, and Stuart N Lane. "Structure from motion (SFM) photogrammetry." In: (2015).

[72] Microsoft. *Kinect For Windows SDK*. https://docs.microsoft.com/en-us/previous-versions/windows/kinect/. Accessed: 2019-08-02.

[73] Greg Milette and Adam Stroud. *Professional Android sensor programming*. John Wiley & Sons, 2012.

[74] David Mills et al. *Network Time Protocol*. Tech. rep. RFC 958, M/A-COM Linkabit, 1985.

[75] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. "No-reference image quality assessment in the spatial domain." In: *IEEE Transactions on Image Processing* 21.12 (2012), pp. 4695–4708.

[76] Mario Montagud, Fernando Boronat, Hans Stokking, and Ray van Brandenburg. "Inter-destination multimedia synchronization: schemes, use cases and standardization." In: *Multimedia systems* 18.6 (2012), pp. 459–482.

[77] Christopher Müller, Stefan Lederer, and Christian Timmerer. "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments." In: *Proceedings of the 4th Workshop on Mobile Video*. ACM. 2012, pp. 37–42.

[78] *Nine Inch Nails remix project*. http://www.ninremixes.com/multitracks.php. Accessed: 2019-03-03.

[79] Gabrielle Odowichuk, Shawn Trail, Peter Driessen, Wendy Nie, and W Page. "Sensor fusion: Towards a fully expressive 3D music control interface." In: Aug. 2011, pp. 836–841. DOI: 10.1109/PACRIM.2011.6033003.

[80] Ioannis Pachoulakis, Nikolaos Xilourgos, Nikolaos Papadopoulos, and Anastasia Analyti. "A Kinect-based physiotherapy and assessment platform for Parkinson's disease patients." In: *Journal of medical engineering* 2016 (2016).

[81] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. "Diatom autofocusing in brightfield microscopy: a comparative study." In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 3. IEEE. 2000, pp. 314–317.

[82] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. "Analysis of focus measure operators for shape-from-focus." In: *Pattern Recognition* 46.5 (2013), pp. 1415–1432.

[83] Christopher J Plack, Andrew J Oxenham, and Richard R Fay. *Pitch: Neural Coding and Perception*. Vol. 24. Springer Science & Business Media, 2006.

[84] Emmanouil Potetsianakis. "Streaming and Presentation Architectures for Extended Video Streams." In: *Adjunct Publication of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video*. TVX '17 Adjunct. Hilversum, The Netherlands: ACM, 2017, pp. 129–132. ISBN: 978-1-4503-5023-5. DOI: 10.1145/3084289.3084298. URL: http://doi.acm.org/10.1145/3084289.3084298.

[85] Emmanouil Potetsianakis and Jean Le Feuvre. "SWAPUGC: Software for Adaptive Playback of Geotagged UGC." In: *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM. 2018, pp. 456–459. DOI: 10.1145/3204949.3208142. URL: http://doi.acm.org/10.1145/3204949.3208142.

[86] Emmanouil Potetsianakis, Emmanouil Ksylakis, and Georgios Triantafyllidis. "A Kinect-based Framework For Better User Experience in Real-Time Audiovisual Content Manipulation." In: *Telecommunications and Multimedia (TEMU), 2014 International Conference on*. 2014, pp. 238–242. DOI: 10.1109/TEMU.2014.6917767.

[87] Emmanouil Potetsianakis and Jean Le Feuvre. "Streaming of Kinect Data for Interactive In-Browser Audio Performances." In: *Proceedings of the Audio Mostly 2016*. ACM. 2016, pp. 258–265.

[88] Emmanouil Potetsianakis and Jean Le Feuvre. "Buffer Management for Synchronous and Low-Latency Playback of Multi-Stream User-Generated Content." In: *IEEE International Conference on Multimedia Information Processing and Retrieval*. MIPR '18. Miami, FL, USA: IEEE, 2018. DOI: 10.1109/MIPR.2018.00022.

[89] Darijo Raca, Jason J Quinlan, Ahmed H Zahran, and Cormac J Sreenan. "Beyond throughput: a 4G LTE dataset with channel and context metrics." In: *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM. 2018, pp. 460–465.

[90] P Venkat Rangan, Srinivas Ramanathan, and Srihari SampathKumar. "Feedback techniques for continuity and synchronization in multimedia information retrieval." In: *ACM Transactions on Information Systems (TOIS)* 13.2 (1995), pp. 145–176.

[91] P Venkat Rangan, Srinivas Ramanathan, Harrick M Vin, and Thomas Kaeppner. "Techniques for multimedia synchronization in network file systems." In: *computer communications* 16.3 (1993), pp. 168–176.

[92] Charlie Roberts and J Kuchera-Morin. *Gibber: Live Coding Audio in The Browser*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 2012.

[93] G. Romaniello, E. Potetsianakis, O. Alphand, R. Guizzetti, and A. Duda. "Fast and energy-efficient topology construction in multi-hop multi-channel 802.15.4 networks." In: *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2013, pp. 382–387. DOI: 10.1109/WiMOB.2013.6673388.

[94] Michael del Rosario, Stephen Redmond, and Nigel Lovell. "Tracking the evolution of smartphone sensing for monitoring human movement." In: *Sensors* 15.8 (2015), pp. 18901–18933.

[95] Michele A Saad, Alan C Bovik, and Christophe Charrier. "Blind image quality assessment: A natural scene statistics approach in the DCT domain." In: *IEEE transactions on Image Processing* 21.8 (2012), pp. 3339–3352.

[96] Mukesh Kumar Saini, Raghudeep Gadde, Shuicheng Yan, and Wei Tsang Ooi. "MoViMash: online mobile video mashup." In: *Proceedings of the 20th ACM international conference on Multimedia*. ACM. 2012, pp. 139–148.

[97] H Schulzrinne, S Casner, R Frederick, and V Jacobson. "RFC 3550." In: *RTP: a transport protocol for real-time applications* 7 (2003).

[98] Rajiv Ratn Shah, Mohamed Hefeeda, Roger Zimmermann, Khaled Harras, Cheng-Hsin Hsu, and Yi Yu. "NEWSMAN: Uploading Videos over Adaptive Middleboxes to News Servers In Weak Network Infrastructures." In: *International Conference on Multimedia Modeling*. Springer. 2016, pp. 100–113.

[99] Prarthana Shrestha, Hans Weda, Mauro Barbieri, Emile HL Aarts, et al. "Automatic mashup generation from multiple-camera concert recordings." In: *Proceedings of the 18th ACM international conference on Multimedia*. ACM. 2010, pp. 541–550.

[100] Matti Siekkinen, Enrico Masala, and Jukka K Nurminen. "Optimized upload strategies for live scalable video transmission from mobile devices." In: *IEEE Transactions on mobile computing* 16.4 (2017), pp. 1059–1072.

[101] Iraj Sodagar. "The MPEG-DASH Standard For Multimedia Streaming Over The Internet." In: *IEEE MultiMedia* 4 (2011), pp. 62–67.

[102] M. Soleymani, M. Pantic, and T. Pun. "Multimodal Emotion Recognition in Response to Videos." In: *IEEE Transactions on Affective Computing* 3.2 (2012), pp. 211–223. ISSN: 1949-3045. DOI: 10.1109/T-AFFC.2011.37.

[103] Cormac J. Sreenan, Jyh-Cheng Chen, Prathima Agrawal, and B Narendran. "Delay reduction techniques for playout buffering." In: *IEEE Transactions on Multimedia* 2.2 (2000), pp. 88–100.

[104] Nikolaos Stefanakis, Stavros Chonianakis, and Athanasios Mouchtaris. "Automatic matching and synchronization of user generated videos from a large scale sport event." In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE. 2017, pp. 3016–3020.

[105] Denny Stohr, Stefan Wilk, Iva Toteva, Wolfgang Effelsberg, and Ralf Steinmetz. "Context not Content: A Novel Approach to Real-Time User-Generated Video Composition." In: *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2016, pp. 65–70.

[106] "Synchronized Multimedia Integration Language (SMIL) 3.0." In: ().

[107] Xiangchun Tan, Jörgen Gustafsson, and Gunnar Heikkilä. "Perceived video streaming quality under initial buffering and rebuffering degradations." In: *MESAQIN Conference (June 2006)*. Vol. 90. 2006.

[108] Christian Timmerer, Jean Gelissen, Markus Waltl, and Hermann Hellwagner. "Interfacing with virtual worlds." In: *Network and Electronic Media Summit* (2009).

[109] Aruna Tyagi, Sunil Semwal, and Gautam Shah. "A Review of Eeg Sensors used for Data Acquisition." In: *National Conference on Future Aspects of Artificial intelligence in Industrial Automation (NCFAAIIA 2012)*. International Journal of Computer Applications®(IJCA).

[110] Liviu-Octavian Varga, Gabriele Romaniello, Mališa Vučinić, Michel Favre, Andrei Banciu, Roberto Guizzetti, Christophe Planat, Pascal Urard, Martin Heusse, Franck Rousseau, et al. "GreenNet: An Energy-Harvesting IP-Enabled Wireless Sensor Network." In: *IEEE Internet of Things Journal* 2.5 (2015), pp. 412–426.

[111] Markus Waltl, Christian Timmerer, and Hermann Hellwagner. "A test-bed for quality of multimedia experience evaluation of sensory effects." In: *Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on*. IEEE. 2009, pp. 145–150.

[112] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy. "Evaluation of Pose Tracking Accuracy in the First and Second Generations of Microsoft Kinect." In: *2015 International Conference on Healthcare Informatics*. 2015, pp. 380–389. DOI: 10.1109/ICHI.2015.54.

[113] Zhou Wang and Alan C Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures." In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[114] D. Webster and O. Celik. "Experimental evaluation of Microsoft Kinect's accuracy and capture rate for stroke rehabilitation applications." In: *2014 IEEE Haptics Symposium (HAPTICS)*. 2014, pp. 455–460. DOI: 10.1109/HAPTICS.2014.6775498.

[115] Stefanie Wechtitsch, Marcus Thaler, Andras Horti, Albert Hofmann, Werner Bailer, Wolfram Hofmeister, Jameson Steiner, and Reinhard Grandl. "Automatic Selection of Live User Generated Content." In: *WSICC@ TVX*. 2016.

[116] Stefan Wilk, Stephan Kopf, and Wolfgang Effelsberg. "Video Composition by the Crowd: A System to Compose User-generated Videos in Near Real-time." In: *Proceedings of the 6th ACM Multimedia Systems Conference*. MMSys '15. New York, NY, USA: ACM, 2015, pp. 13–24. ISBN: 978-1-4503-3351-1. DOI: 10.1145/2713168.2713178. URL: http://doi.acm.org/10.1145/2713168.2713178.

[117] Stefan Wilk, Roger Zimmermann, and Wolfgang Effelsberg. "Leveraging Transitions for the Upload of User-generated Mobile Video." In: *Proceedings of the 8th International Workshop on Mobile Video*. MoVid '16. New York, NY, USA: ACM, 2016, 5:1–5:6. ISBN: 978-1-4503-4357-2. DOI: 10.1145/2910018.2910658. URL: http://doi.acm.org/10.1145/2910018.2910658.

[118] Yue Wu, Tao Mei, Ying-Qing Xu, Nenghai Yu, and Shipeng Li. "Movieup: Automatic mobile video mashup." In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.12 (2015), pp. 1941–1954.

[119] Chun-Chuan Yang. "Design of the application-level protocol for synchronized multimedia sessions." In: *Communications, 2002. ICC 2002. IEEE International Conference on*. Vol. 4. IEEE. 2002, pp. 2518–2522.

[120] Hiba Yousef, Jean Le Feuvre, Giuseppe Valenzise, and Vedad Hulusic. "Video Quality Evaluation for Tile-Based Spatial Adaptation." In: *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE. 2018, pp. 1–6.

[121] Lourdes Beloqui Yuste, Fernando Boronat, Mario Montagud, and Hugh Melvin. "Understanding timelines within mpeg standards." In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 368–400.

[122] Alex Zambelli. "IIS smooth streaming technical overview." In: *Microsoft Corporation* 3 (2009), p. 40.

[123] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. "Internet of things for smart cities." In: *IEEE Internet of Things journal* 1.1 (2014), pp. 22–32.

A

APPENDIX

A.1 BUFFER-BASED ADAPTATION ALGORITHM

Throughout the manuscript, we are using references to HAS adaptation algorithms. Especially in Chapter 5, we run simulations using a buffer-based adaptation policy described in this section.

In HAS systems, the purpose of the adaptation algorithm is to detect the capabilities of the network connection with the server and identify the highest-bitrate representation that can support. The network evaluation typically happens by throughput estimation (throughput-based adaptation), or by observing the variations in the buffer contents (buffer-based adaptation) [60].

We implemented a conservative buffer-based adaptation policy that on top of supporting multiple representations, supports multiple sources/streams. Figure 49 shows an overview of the algorithm. We define two thresholds R_{max} which if reached the buffer is considered to be full, and R_{min} that if the buffer occupancy falls bellow it is considered to be almost empty. Whenever a buffer update occurs, if the client is in a high-bitrate representation and the buffer level is between R_{min} and R_{max}, it just requests the next segment of the same high-bitrate. Otherwise, if it is above R_{max}, it does nothing, or if it is bellow R_{min} it requests a low-bitrate segment.

If the update occurs, while in a low-bitrate representation and the buffer level is between R_{min} and R_{max}, if it just switched to this stream, tries to request a high-bitrate segment, or, if it was already on this stream, the client requests the next low-bitrate segment. If at the time of the update the buffer level is bellow R_{min}, it switches to a different stream (staying in low-bitrate representation). The reasoning behind this behaviour is that the buffer might be depleting due to the connection of the recorder to the server and not from the server to the client. If the buffer level at the update is above R_{max}, the client does nothing.

The algorithm defines the "just switched" behaviour by the number of segments requested in the current stream (Ns_{req}). The policy can be parametrized by setting different Ns_{req}, R_{max} and R_{min} values.

The proposed algorithm is loosely based on BBA [49], because it also uses the R_{max} and R_{min}. However, the main difference is that BBA has no provision for multiple sources/streams and that BBA is more aggressive because it always tries to switch to a high-bitrate representation when the buffer level is above R_{max}.

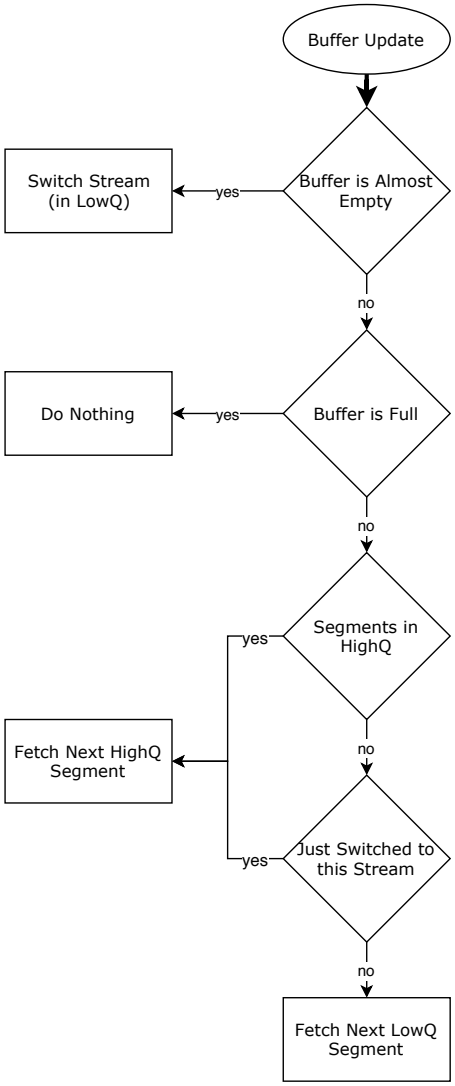


Figure 49: Overview of buffer-based quality adaptation algorithm

Bellow is a list of the specific policy values used at the simulations for Chapter 5:
Segment Length = 2s
R_{max} = 2 segments (4s)
R_{min} = ½ segment (1s)
Ns_{req} = 1 segment (2s)

FIELD-OF-VIEW (FoV) of a camera is the spatial area that is visually covered during the recording of a video. Because we use it for visualization and indexing, we are interested in its 2D projection.

To calculate the FoV, the location (P), direction ($\vec{d}$), visible distance (R) and angle of view (α) of the camera are required [10]. P and $\vec{d}$ are provided by the respective location and orientation sensors of the device. R is difficult to be evaluated for most cases, since it requires extensive prior knowledge about the surroundings (obstacles, weather etc.) but the value can be set according to the application. For example, in indoor recordings, an R value of 20 to 60 m should accommodate accurately enough FoV estimations. For outdoors, R can vary a lot and might reach up to 800 m; for such cases, it can be approximated using hotspot identification [46].

The angle of view can be calculated, using the sensor (film) size (d) and the effective focal length (f) of the camera, with the following formula:

$$\alpha = 2 * arctan\frac{d}{2f}$$

Because d and f are not always available, when they are missing we are doing a conservative FoV estimation using $\alpha \approx 65°$. This value is selected because it corresponds to α for (wide) lenses between 24 and 35mm [1], which are the cameras that are typical in modern smartphones [2] [3]. For comparison, other platforms use also $\alpha = 65°$ as default value (e.g. when calculating the camera panning speed) [31], while GeoUGV was using $\alpha = 51°$ when the other parameters were not available [66].

REGION-OF-INTEREST (RoI) is the spatial area that a specific event is taking place during the recording (e.g. a scene at a music concert). Unless it is predefined, it can be automatically identified via consensus on the active views, which is based on the assumption that a significant amount of "good" quality cameras will record it.

For our work we recommend using a sensor-based system [30]. The methodology of the system is fist to define a viewing angle (e.g. for music concerts 90° is recommended [31]). Then, use the orientation recordings history (it can be short-term for fast bootstrap and/or moving RoIs) to identify the RoI. The recording history can also be used to refine the viewing angle over time.

Figure 50 shows (a) the visualization of seven active recorders and (b) the histogram of the recorded compass values used to calculate the viewing angle and RoI. We can observe the importance of the

1 https://en.wikipedia.org/wiki/Angle_of_view#Common_lens_angles_of_view
2 https://www.gsmarena.com/
3 https://medium.com/@andrewkemendo/smartphone-camera-focal-length-reference-for-computer-vision-and-ar-applications-cd4c932046b1

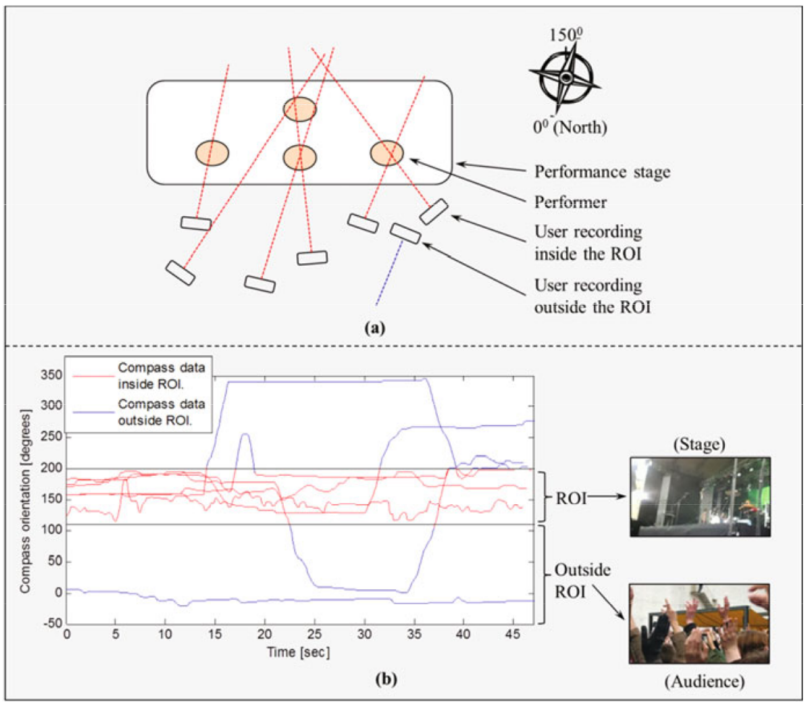


Figure 50: RoI identification (a) visualization of recorders (b) histogram of cameras orientation (from [30])

consensus system in Fig. 50 (b) that there is a user constantly filming an area outside the RoI and would have been included if a simple averaging algorithm was used instead. The illustrated example assumes that all the recorders are on the same side of the scene, thus the RoI is perceived as a plane, but similar sensor-based approaches exist for convex hull RoIs [116].

A.3 USER STUDY FEEDBACK

Following are the participant IDs with the comments they made during or after the user study.

ID: 0 - test run

ID: 1 - test run

ID: 2 - test run

ID: 3 - test run

ID: 4
"Over time it becomes hard, because I do not have in mind the impressions from previous videos"
"The frequency of the scene cuts varies within the same video"
"The evaluation was a difficult task"

ID: 5
"The cuts between views were not smooth"
"The view switching should be spatially consistent"
"The camera in some views moves very fast"

ID: 6
(Commenting specifically on S5) "I would like a fixed camera that just shows the whole orchestra"
(Commenting specifically on S5) "In a classical music concert the question does not make much sense"
"It would be interesting to have a camera that focuses on the players only"

ID: 7
"A lot of view obstructions"
"Shakiness is very annoying"
"The worst part of the videos were when the camera was filming the crew or random people"
"Resolution was not that much of a problem, comparing to the other issues (like shakiness)"
"Clean close-ups were missing"

ID: 8
"All of the videos were very bad overall"
"I prefer bad image quality than shakiness or not filming the orchestra"

ID: 9
"The questions put me in a negative disposition"
"It was very disturbing when the camera was not showing the orchestra"
"I prefer cameras with less detail (i.e. lower image quality) and less shakiness, than the opposite"
"View obstruction from people or objects was annoying. The effect was especially annoying when the person filming was passing behind an obstacle (person or object), because if a person passes in front of the camera is expected to happen at some point"

ID: 10
"I personally can tolerate bad image quality"
"At points the camera was far away and in low-quality and couldn't see the details"
"I paid too much attention in the beginning, so it became tiring after a while. Maybe if there was some sort of interaction with the user it would feel more interesting."
"It would be interesting the same test to be conducted in a mobile phone (because of different expectations, affects experience etc.)"

ID: 11
"I felt more focused in the beginning of the study"
"I think that if I watched the videos twice, I would change my mind"
"The study was too long"
"Maybe we should evaluate the good training video, to set a baseline."
"The videos were too many (and too long). Maybe they should be shortened down to the disputable parts"

ID: 12
"Cameras were of bad quality. Feels like the videos were predefined extracts that were shuffled and stitched" (when asked what was the most striking issue with the videos) "shakiness of the camera and filming floors, walls etc."
"The scene showing when the video ends shapes my opinion" (i.e. if a video finishes with a bad view, I have a bad opinion for the video)

ID: 13
"If the videos were longer it might have been tiring watching them" (commented after replying S4)
"I have watched UGC videos with the audio from each camera. In this case that the audio comes from one source it is more difficult to judge the bad videos"
"The image quality when the video starts affects positively my perception (if it starts with a good view I might overlook further degra-

dations)

ID: 14
"Shakiness is really annoying"
"Even if it is showing the action and the image is clear, if it is shaky I would have switched off after a few seconds"
"...especially when the subject is close to the camera, the shakiness effect is horrible"
"Image Quality is the least important"
"Even just one close-up but steady camera would be better (and switching to a steady camera gives a sense of “immersiveness“)"
"Cello and Bass are not very visible (and when they are, it’s shaky)"
"As soon as the camera is stable, it is fine no matter what is the quality"
"Sometimes the camera switches too late (a bit after it is out of FoV)"

ID: 15
"Some videos shake a lot"
"The more videos I am watching, the more confused I am"

ID: 16
"Questions would have been better if the were either positive or negative, not mixed"
"The problem with the cameras is not that they change too often, it is that they change abruptly"
(wrt S5) "I personally find interesting seeing the rest of the room, not just the orchestra. But I do not like it when it is filming the floor etc."
"The abrupt changes influence my overall opinion on the view changes"
"I would like to have a camera with a close-up to some player (at least at some point)"
"When an instrument keeps playing, with the others stopped, I would expect to have that specific instrument on my view, not the idle players"

ID: 17
"It was confusing that the statements were mixed positive/negative"
"All of the videos in terms of overall quality were close"
"I prefer the steady cameras (without shakiness)"

ID: 18
"Comparing to blurriness, shakiness is more unbearable"
"Fast move and shakiness made me feel dizziness and a bit sick"
(wrt S4) "For the last few videos I felt more tired (and I appreciated more when there was a good view)"
"Shooting out of RoI made me lose focus"

ID: 19
(wrt S5) "Answering was problematic because all the views were similar"
"When the last scene was bad, it was influencing negatively my answer"
"Shakiness and camera moving fast were very annoying and tiring"

ID: 20
"Some times it was showing me stuff I was not interested about"
"It was a difficult evaluation"
"Too shaky was the worst"
"I'd prefer a video that was less shaky and of worse quality"

ID: 21
"I feel I have seen the same views with and without image stabilization and with and without image degradation"
"It was difficult to tell artifacts from encoding from artifacts from the content"
"The most important problem was moving around and lack of stabilization (shakiness), followed by bad image quality and bad FoV"
"I would rather have a stable, clean in FoV view, even if people are pasing in front of the camera"
"Away views made me more resilient to shakiness than close-ups"
"Quality switches were more tolerable when happening in a transition, not in the middle of a scene"

ID: 22
(after watching the training videos) "'Best' video depends on the goal - e.g. editing a video for impression in different than editing for casually watching"
"A couple of videos start really bad and end really good, in those cases it is difficult to decide"
"A sudden up-to-down move that occurred in some videos was very disturbing"

ID: 23
"I think all the videos are very bad"
"Some begin ok and at the end are really bad. There is a consistency problem"

ID: 24
"Many views would have been better with a fixed camera (instead of the user moving around)"
"The view changes were usually ok, but for me the most important was the framing"
"Sometimes it is tiring because it does not follow well the current

frame [meaning shot / camera framing]"
"Some of the transitions were nice, but others can be improved"
"Out of FoV shots sometimes can be interesting (e.g. to peak "behind the scenes" or the audience)"
"Shakiness of the camera was the most annoying factor"
"Sometimes a nice smooth move of the camera was disrupted from a switch"

ID: 25
"It was not easy to separate the whole experience for the individual questions"
"Moving and Shakiness were the worst factors - it tired me. If the image is just not clean it is normal"

Assuming a system generating consecutive non-overlapping frames of duration T_f each, with varying delays. At a given time t, the client-side buffer contains 6 frames, as illustrated in Figure 51. The received frames (with $T\_arr \leqslant t$) are displayed in gray, while the delayed ($T\_arr > t$) are whitened out.

The buffer size (B_{size}) should be equal to the number of *consecutive* frames contained, starting from the first frame to be scheduled for playback, in that example the $B_{size} = 3$, with first frame FRN = n. The buffer length is equal to the buffer size multiplied by the frame duration (i.e. $B_{len} = B_{size} * T_f$). The *fragmented* buffer size (B_{size_frag}) is equal to the *total* number of frames in the buffer, in that example $B_{size\_frag} = 6$. Respectively, the fragmented buffer length is equal to the fragmented buffer size, multiplied to the frame duration (i.e. $B_{len\_frag} = B_{size\_frag} * T_f$).

Figure 51: Buffer visualization with received (gray) and missing (white) frames

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Amélioration des Applications Vidéo Grâce aux Métadonnées Temporelles

Mots clés : Streaming, Multimedia, UX, UGC, Synchronization

Résumé : Les dispositifs d'enregistrement vidéo sont souvent équipés de capteurs (smartphones par exemple, avec récepteur GPS, gyroscope, etc.) ou utilisés dans des systèmes où des capteurs sont présents (par exemple, caméras de surveillance, zones avec capteurs de température et/ou d'humidité). Par conséquent, de nombreux systèmes traitent et distribuent la vidéo avec des flux de métadonnées temporels, souvent sous forme de contenu généré par l'utilisateur (UGC). La diffusion vidéo a fait l'objet d'études approfondies, mais les flux de métadonnées ont des caractéristiques et des formes différentes, et il n'existe en pratique pas de méthode cohérente et efficace pour les traiter conjointement avec les flux vidéo. Dans cette thèse, nous étudions les moyens d'améliorer les applications vidéo grâce aux métadonnées temporelles. Nous définissons comme métadonnées temporelles toutes les données non audiovisuelles enregistrées ou produites, qui sont pertinentes à un moment précis sur la ligne de temps du média.
"L'amélioration" des applications vidéo a une double signification, et ce travail se compose de deux parties respectives. Premièrement, utiliser les métadonnées temporelles pour étendre les capacités des applications multimédias, en introduisant de nouvelles fonctionnalités. Deuxièmement, utiliser les métadonnées chronométrées pour améliorer la distribution de contenu pour de telles applications.
Pour l'extension d'applications multimédias, nous avons adopté une approche exploratoire et nous présentons deux cas d'utilisation avec des exemples d'application. Dans le premier cas, les métadonnées temporelles sont utilisées comme données d'entrée pour générer du contenu, et dans le second, elles sont utilisées pour étendre les capacités de navigation pour le contenu multimédia sous-jacent. En concevant et en mettant en œuvre deux scénarios d'application différents, nous avons pu identifier le potentiel et les limites des systèmes vidéo avec métadonnées temporelles.
Nous utilisons les résultats de la première partie afin d'améliorer les applications vidéo, en utilisant les métadonnées temporelles pour optimiser la diffusion du contenu. Plus précisément, nous étudions l'utilisation de métadonnées temporelles pour l'adaptation multi-variables dans la diffusion vidéo multi-vues et nous testons nos propositions sur une des plateformes développées précédemment. Notre dernière contribution est un système de buffering pour la lecture synchrone et à faible latence dans les systèmes de streaming en direct.

Title : Enhancing Video Applications Through Timed Metadata

Keywords : Streaming, Multimedia, UX, UGC, Synchronization

Abstract : Video recording devices are often equipped with sensors (smartphones for example, with GPS receiver, gyroscope etc.), or used in settings where sensors are present (e.g. monitor cameras, in areas with temperature and/or humidity sensors). As a result, many systems process and distribute video together with timed metadata streams, often sourced as User-Generated Content. Video delivery has been thoroughly studied, however timed metadata streams have varying characteristics and forms, thus a consistent and effective way to handle them in conjunction with the video streams does not exist.
In this Thesis we study ways to enhance video applications through timed metadata. We define as timed metadata all the non-audiovisual data recorded or produced, that are relevant to a specific time on the media timeline.
"Enhancing" video applications has a double meaning, and this work consists of two respective parts. First, using the timed metadata to extend the capabilities of multimedia applications, by introducing novel functionalities. Second, using the timed metadata to improve the content delivery for such applications.
To extend multimedia applications, we have taken an exploratory approach, and we demonstrate two use cases with application examples. In the first case, timed metadata is used as input for generating content, and in the second, it is used to extend the navigational capabilities for the underlying multimedia content. By designing and implementing two different application scenarios we were able to identify the potential and limitations of video systems with timed metadata.
We use the findings from the first part, to work from the perspective of enhancing video applications, by using the timed metadata to improve delivery of the content. More specifically, we study the use of timed metadata for multi-variable adaptation in multi-view video delivery - and we test our proposals on one of the platforms developed previously. Our final contribution is a buffering scheme for synchronous and low-latency playback in live streaming systems.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France