Pontifícia Universidade Católica
do Rio de Janeiro

Paulo Renato Conceição Mendes

Spatio-temporal Localization of Actors in Video/360-Video and its Applications

Dissertação de Mestrado

Master thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Sérgio Colcher

Rio de Janeiro
August 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Paulo Renato Conceição Mendes

Spatio-temporal Localization of Actors in Video/360-Video and its Applications

Master thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Sérgio Colcher
Advisor
Departamento de Informática – PUC-Rio

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Dr. Roberto Gerson de Albuquerque Azevedo
Disney Research

Rio de Janeiro, August 16^{th}, 2021

Paulo Renato Conceição Mendes

Bachelor's degree in Computer Science at Federal University of Maranhão (UFMA) in 2019.

Bibliographic data

Mendes, Paulo Renato Conceição

Spatio-temporal Localization of Actors in Video/360-Video and its Applications / Paulo Renato Conceição Mendes; advisor: Sérgio Colcher. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2021.

v., 75 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Clusterização. 2. Reconhecimento Facial. 3. Recomendação de Vídeo. 4. Vídeo 360.
I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Thanks to my advisor Prof. Sérgio Colcher for his guidance and support in this journey. Thanks to my family for the endless support. Thanks to my friends from TeleMídia Lab, for their friendship and support. To all colleagues, faculty and staff of the PUC Rio Department of Informatics for the fellowship, learning and support. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Mendes, Paulo Renato Conceição; Colcher, Sérgio (Advisor). **Spatio-temporal Localization of Actors in Video/360-Video and its Applications**. Rio de Janeiro, 2021. 75p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The popularity of platforms for the storage and transmission of video content has created a substantial volume of video data. Given a set of actors present in a video, generating metadata with the temporal determination of the interval in which each actor is present, and their spatial 2D localization in each frame in these intervals can facilitate video retrieval and recommendation. In this work, we investigate *Video Face Clustering* for this spatio-temporal localization of actors in videos. We first describe our method for *Video Face Clustering* in which we take advantage of face detection, embeddings, and clustering methods to group similar faces of actors in different frames and provide the spatio-temporal localization of them. Then, we explore, propose, and investigate innovative applications of this spatio-temporal localization in three different tasks: (i) *Video Face Recognition*, (ii) *Educational Video Recommendation* and (iii) *Subtitles Positioning in 360-video*. For (i), we propose a cluster-matching-based method that is easily scalable and achieved a recall of 99.435% and precision of 99.131% in a small video set. For (ii), we propose an unsupervised method based on the presence of lecturers in different videos that does not require any additional information from the videos and achieved a mAP≈99%. For (iii), we propose a dynamic placement of subtitles based on the automatic localization of actors in 360-video.

Keywords

Clustering; Face Recognition; Video Recommendation; 360-Video.

Resumo

Mendes, Paulo Renato Conceição; Colcher, Sérgio. **Spatio-temporal Localization of Actors in Video/360-Video and its Applications**. Rio de Janeiro, 2021. 75p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A popularidade de plataformas para o armazenamento e compartilhamento de vídeo tem criado um volume massivo de horas de vídeo. Dado um conjunto de atores presentes em um vídeo, a geração de metadados com a determinação temporal dos intervalos em que cada um desses atores está presente, bem como a localização no espaço 2D dos quadros em cada um desses intervalos pode facilitar a recuperação de vídeo e a recomendação. Neste trabalho, nós investigamos a *Clusterização Facial em Vídeo* para a localização espaço-temporal de atores. Primeiro descrevemos nosso método de *Clusterização Facial em Vídeo* em que utilizamos métodos de detecção facial, geração de *embeddings* e clusterização para agrupar faces dos atores em diferentes quadros e fornecer a localização espaço-temporal destes atores. Então, nós exploramos, propomos, e investigamos aplicações inovadoras dessa localização espaço-temporal em três diferentes tarefas: (i) *Reconhecimento Facial em Vídeo*, (ii) *Recomendação de Vídeos Educacionais* e (iii) *Posicionamento de Legendas em Vídeos 360°*. Para a tarefa (i), propomos um método baseado na similaridade de clústeres que é facilmente escalável e obteve um recall de 99.435% e uma precisão de 99.131% em um conjunto de vídeos. Para a tarefa (ii), propomos um método não supervisionado baseado na presença de professores em diferentes vídeos. Tal método não requer nenhuma informação adicional sobre os vídeo e obteve um valor mAP≈99%. Para a tarefa (iii), propomos o posicionamento dinâmico de legendas baseado na localização de atores em vídeo 360°.

Palavras-chave

Clusterização; Reconhecimento Facial; Recomendação de Vídeo; Vídeo 360.

Table of contents

List of figures

List of tables

List of abreviations

BCL – Ball Cluster Learning

CNN – Convolutional Neural Network

FDDB – Face Detection Data set and Benchmark

FOV – Field of View

FRPC – Face Recognition Prize Challenge

GTP – Ground Truth Positives

HMD – Head-Mounted Display

HOG – Histogram of Oriented Gradients

HTML – HyperText Markup Language

HUD – Heads-up Display

IARPA – Intelligence Advanced Research Projects Activity

LFW – Labeled Faces in the Wild

MAP – Mean Average Precision MOOC – Massive Open Online Course

MTCNN – Multitask Cascaded Convolutional Networks

NAN – Neural Aggregation Network

NCL – Nested Context Language

NMS – Non-Maximum Suppression

PIP – Picture-in-Picture

SMIL – Synchronized Multimedia Integration Language

SRT – SubRip Subtitle Format

VR – Virtual Reality

VTE – Virtual Teaching Environment

XML – Extensible Markup Language

1
Introduction

In recent years, the popularity of platforms for the storage and transmission of video content has stimulated the production of a massive volume of video data, establishing new habits, and leveraging new applications with innovative forms of consumption of this kind of information. As an indication of this huge production (and consumption) of data, in 2019, for example, more than one billion hours of YouTube videos were watched per day.[1]

Generating metadata with (i) the identity information of actors present, (ii) the temporal determination of the intervals in which each of these actors are present, and (iii) their spatial localization in each of the frames along these intervals can facilitate video indexing, retrieval, recommendation and a series of other tasks which might enhance the way people interact and consume all this video data. Besides the identification, (ii) and (iii) together are what we call *Spatio-temporal Localization*.

In this dissertation, we investigate a method for the spatio-temporal localization of actors in videos. Our expected contribution is two-fold:

1. we propose a core process for the spatio-temporal localization in which we take advantage of face detection, embeddings, and clustering methods to group similar faces (presumably from the same actors) along with the frames, and

2. we further propose and explore the innovative application of this localization in three different practical and important tasks: *Video Face Recognition*, *Educational Video Recommendation*, and *Subtitles Positioning in 360-video*.

The common core of this dissertation relies on its first part, in which we describe a *Video Face Clustering* method for the spatio-temporal localization of actors. We describe its process composed of: *frame extraction*, *face detection*, *embeddings generation*, and *clustering*. Based on this *Video Face Clustering* method, we investigate three applications, all of them enabled by that common method of spatio-temporal localization and its benefits.

[1] https://kinsta.com/blog/youtube-stats/

The first application, *Video Face Recognition*, has been attracting the attention of researchers for more than two decades. Since the deep learning boom, face detection and recognition performance have greatly improved in terms of both speed and accuracy (4). Nowadays, face recognition systems are used in many areas such as video surveillance and security systems, video analytics systems, smart shopping, automatic face tagging in photo collections, investigative tools that search for identities in social networks based on face images, and in thousands of other applications in our daily lives.

In our work, we propose a *cluster-matching-based approach* for *Video Face Recognition* where clustering is used to group faces in both the face dataset and in the target video. This method was motivated (and made possible) mainly by the characteristics of our core *Video Face Clustering* method. Its main benefit is that classes (which represent different actors) do not have to be previously known, so the effort spent with annotations is significantly reduced — as it is done over clusters instead of single images. Consequently, face recognition becomes a task of comparing clusters from the dataset with the ones extracted from images or video sources. Therefore, our approach is easily scalable and can be used to automatically generate video metadata.

The second application, *Educational Video Recommendation*, may be considered as one of a more general class of applications referenced as *recommendation system applications* and was motivated by a shift of paradigms that we have been observing in recent years. The traditional paradigm of classroom courses, centered on the physical presence of a teacher, has been gradually giving space to online and hybrid courses, which enables the emergence of VTEs (Virtual Teaching Environment) and MOOCs (*Massive Open Online Courses*). If, on the one hand, the abundance of educational videos can contribute to and facilitate learning, on the other hand, it also makes it challenging to discover and access some specific content of interest (5). This issue is usually addressed by a proactive user search (using queries, for example), or by automatic recommendations made by specialized systems.

In general, current video recommendation methods are heavily dependent on textual information from the video, such as labels (*i.e.* keywords) (6, 7), or automatically generated captions (8) from the lecturer speech. These systems face problems such as errors introduced by manually inserted labels or by imprecise speech recognition. In this work, we propose an additional feature to enhance the recommendation of educational video content which is based on actors (in this specific case, lecturers) presence. To do that, again we take advantage of our core *Video Face Clustering* method. More precisely, we detect lecturers in a video taken as a reference and perform a clustering based on the

face of these lecturers in different videos. Given these clusters, we extract their *centroids* and perform another clustering step for creating a relationship between videos that share the presence of the same lecturers. Finally, we rank the recommended videos based on the amount of time the referenced lecturers were present. A particular benefit of this approach is that it can be done without supervision, allowing for new videos to be automatically analyzed.

Our third application (*Subtitles Positioning in 360-Video*) was motivated by the recent popularization of omnidirectional cameras and Head-Mounted-Displays (HMDs) that increased the amount of 360-video content available. Omnidirectional videos are spherical visual signals that allow the viewer to look around a full 360-degree view of a scene from a specific point.

Several people use subtitles when consuming audiovisual media, and these subtitles are important in contributing to the understanding of the video content (1). There are also some people who choose to consume videos without sound (2). Additionally, the work of Hayati and Mohmedi (9), as referenced in Hughes *et al.* (2), shows that consumers are more likely to watch videos entirely if they have subtitles presented with them. In traditional 2D videos, static subtitles are commonly used and they are usually placed at a fixed position, most commonly at the bottom-center of the screen (10). Different from traditional 2D videos, subtitles positioning in 360-videos is challenging because it involves both temporal and spatial domains (11), and there is no fixed “bottom-center" of the screen (1). Most current solutions rely on positioning subtitles either statically to the viewer or at a fixed position in the 360-degree environment. In this work, we adapt and apply our current solution for the spatio-temporal localization of actors to the 360-video domain. By doing that, we intend to use this localization for positioning subtitles close to the actors in the 360-video.

The remainder of this dissertation proposal is structured as follows. In Chapter 2, we discuss some works related to the core method of our dissertation and each of the applications we investigate. In Chapter 3, we define our approach for spatio-temporal localization of actors, which is our core method. The following three chapters contain the applications in which we investigate the applicability of this method. The first application, that of *Video Face Recognition*, is described in Chapter 4. In Chapter 5, we present our application of *Educational Video Recommendation*. Our last application, that of *Subtitles Positioning in 360-video*, is described in Chapter 6. Finally, in Chapter 7, we conclude this dissertation and point to possible future work.

2
Related Work

In this chapter, we make a brief review of the works related to ours. Section 2.1 contains work related to the core method of this work. The next three sections describe work related to the applications we propose based on our core method.

2.1
Spatio-temporal Localization of Actors

The task of detecting and tracking actors in videos has been the focus of much research. In early 2004, Küblbeck and Ernst (12) used an illumination invariant approach for face detection combined with a tracking mechanism performed by means of continuous detections. Kim *et al.* (13) addressed the problem of tracking faces in noisy videos using a tracker that adaptively builds a target model reflecting changes in appearance, typical of a video setting. This kind of approach does not perform well in the task of spatio-temporal localization of actors because they can mostly track them when they are continuously present on the video. Differently, the approach we use, which is based on clustering, does not require the actors to be continuously present on the video.

More similar to ours, recent works have investigated the use of clustering for grouping faces of actors in video and, consequently, providing the spatio-temporal localization of them. Tapaswi *et al.* (14) propose Ball Cluster Learning (BCL), a supervised approach to carve the embedding space into balls of equal size, one for each cluster. The radius of such ball is translated to a stopping criterion for iterative merging algorithms. Sharma *et al.* (15) propose a self-supervised Siamese network for video face clustering that can also be used in scenarios where tracks of actors are not available, such as image collections. The approach we use can also be applied to image collections, as further explained in Section 4, but it differs in the sense that we use pre-trained CNNs (Convolutional Neural Networks) and traditional clustering algorithms for performing this task.

It is worth mentioning that, in this dissertation, we do not intend to directly compare or propose a better method for the task of spatio-temporal

localization of actors than the existing ones. Instead, we intend to investigate to what extent our method opens up novel approaches for the three chosen applications and the benefits that can be achieved with our approach.

2.2 Video Face Recognition

Many methodologies have been proposed for *Video Face Recognition*, most commonly relying on comparing selected facial features of a given image with features of faces within a database. Using only one sample reference image of a person's face for the comparison may result in classification errors due to factors related to variations in lighting, image resolution, angle, etc. (16). To overcome this problem, some face recognition approaches use multiple face samples for comparison. However, this strategy does not scale well as the complexity is a function of the number of samples. Other approaches treat the face recognition task as a classification problem (17, 18), where a classifier model learns rules to assign faces to previously known classes within a dataset, where each class corresponds to one person. Nonetheless, this kind of approach does not deal well when new classes are incorporated because of the need to retrain the classification models. Moreover, when dealing with video, these kinds of methods have to be applied to each frame, again increasing their complexity.

Traditional deep learning models for face recognition such as DeepFace (19) and DeepID (20) use a CNN with fully connected layer output to produce a representation of high-level features (face embeddings) from an input image, followed by a softmax layer to indicate the identity of classes. Other approaches, such as FaceNet (21), can directly measure the similarity among faces using euclidean space. Inspired by DeepID, this model uses the *triplet loss* as the loss function to estimate similarity to one character's face to a collection of other faces. Triplet loss improves the accuracy of the CNN output by minimizing the euclidean distance between the anchor and the positive (face of the same identity) while maximizing the distance between the anchor and the negative (face of another identity). In this work, we evaluated different pre-trained CNN backbones on VGGFace2 dataset (22) to generate the face embeddings.

Proprietary systems for face recognition and matching are widely used by social network platforms. For instance, Facer (23) is Facebook's face detection and recognition framework. Given a photograph, it first detects all the faces. Then, it runs a deep model to determine the likelihood of that face belonging to one of the top-N user friends. This allows Facebook to suggest which friends

the user might want to tag within the uploaded photographs. FindFace[1] is an app that matches photos to profile pictures on VKontakte,[2] a Russian social networking website similar to Facebook. FindFace uses a deep model developed by NTech Lab that won the *2017 IARPA Face Recognition Prize Challenge* (FRPC) (24) in two nominations out of three ("Identification Speed" and "Verification Accuracy"). Similarly, our method can detect faces in videos and automatically recognize their identities by a clustering-based algorithm that uses a knowledge base with the faces pre-identified as a reference; however, a comparison with such methods was not possible due to access restrictions.

Some recent works are focused on video face recognition. Pena *et al.* (25) proposed a face recognition system to detect characters within videos, called *Globo Face Stream*. Their method uses a Histogram of Oriented Gradients (HOG) feature combined with a linear classifier to detect faces. Next, they use FaceNet to generate the embeddings, followed by the euclidean distance calculus to measure the similarity among faces. Yang *et al.* (26) proposed a deep network for video face recognition called NAN (Neural Aggregation Network). They use a CNN to generate the embeddings, followed by an aggregation module that consists of two attention blocks that adaptively aggregate the feature vectors to form a single feature inside the convex hull spanned by them. Rao *et al.* (27) proposed a method for video face recognition based on attention-aware deep reinforcement learning. They formulated the process of finding the attention of videos as a Markov decision process and training the attention model without using extra labels. Unlike existing attention models, their method takes information from both the image space and the feature space as the input to make use of face information that is discarded in the feature learning process. Sohn *et al.* (28) proposed an adaptative deep learning framework for image-based face recognition and video-based face recognition. Given an embedding generated by a CNN, their framework adaptation is achieved by distilling knowledge from the network to a video adaptation network through feature matching, performing feature restoration through synthetic data augmentation, and learning a domain-invariant feature through an adversarial domain discriminator.

Like (25, 26, 27, 28), our method uses a CNN to generate face embeddings from face images, with the difference that it uses an unsupervised cluster-based method to compare the similarity among face datasets and faces extracted from videos. Also, our approach can detect faces that do not have an identity registered in the face dataset with excellent performance.

[1]https://findface.br.aptoide.com/app
[2]https://vk.com/

2.3 Educational Video Recommendation

Recommendation mechanisms are usually based on two methods: *collaborative filtering* and *content-based filtering*. In collaborative filtering, the system groups users based on their common interest in items, using users' preferences, rates, purchases, or accesses to those items. With this approach, knowledge about the item's content is not needed; the recommendation is purely based on the relationship between users and items. The content-based filtering, differently, requires items' descriptions; similar items are the ones recommended to the user. Our approach fits in the latter category. In the remainder of this subsection, we describe some works devoted to the task of general *Video Recommendation*. Moreover, we give an especial focus on works that share our goal of investigating *Educational Video Recommendation*.

The way people watch and consume video content has been changing in the last years, moving from the traditional linear content transmission of televisions to streaming platforms. These platforms allow users to consume video content on-demand. Some examples of such platforms are YouTube,[3] Netflix,[4] Prime Video[5] and Globoplay.[6] In such platforms, users can retrieve video content through actively searching for their content of interest or can be presented with recommendations of such content, from which they may select one to watch. In this scenario, recommendations play a fundamental role in content promotion inside these platforms. It is common to use a collaborative filtering approach for recommending content to a specific user, and this kind of approach does not use any information about the content of the video. It is useful and shows good results when both the video content and user have a consumption history stored (29). However, with new titles being uploaded daily to these platforms associated with their expanding user base, collaborative filtering does not perform well with these new titles and users due to the lack of consumption history (30).

Taking into consideration the problem of video recommendation with recently added videos, Li *et al.* (31) propose a content-based video recommendation approach by taking advantage of CNNs to alleviate the cold-start problem. The authors represent video data with features from audio, images, and meta-data from the video content and use such content to recommend videos on a streaming platform. Lee and Abu-El-Haija (32) model recommendation as a video content-based similarity learning problem, and learn deep video em-

[3]https://youtube.com
[4]https://netflix.com
[5]https://primevideo.com
[6]https://globoplay.globo.com

beddings trained to predict video relationships identified by a co-watch-based system but using only visual and audio content. Han *et al.* (33) proposes to take advantage of the intrinsic motion information in dance videos to solve the video recommendation problem. The authors aim at recommending dance videos based on a mid-level action representation called Dancelets and use a random forest-based index to achieve fast matching of styles and to generate the final recommendation ranking of videos. Similar to (31) and (32), we take advantage of CNNs for extracting content from the videos and perform recommendations. Differently, our work focuses on recommending videos sharing the presence of the same actors. Similar to (33), our work also does not require any metadata from the video, it is solely based on the video content.

Regarding *Educational Video Recommendation*, most works perform analyses and comparisons using the video textual description or speech recognition performed on them. Omisore and Samuel (7) propose combining *fuzzy* techniques to recommend books with content suitable for students based on their reading histories in a digital library, while Mahajan *et al.* (6) propose, given a reference video, mining social media, and web for suggesting links for a student to visit. Moreover, Barrére *et al.* (8) use texts from speech recognition to create recommendations. These works are only based on textual characteristics (or content converted to it) for performing recommendations. Our work focuses on using a visual part of the video, more precisely the presence of actors.

2.4 Subtitles Positioning in 360-video

We searched for works that used strategies for subtitles positioning and extracted the strategies they presented, also described as subtitling behaviour in 360-degree videos (1). Then, we merged the similar strategies and divided them into three main categories: *screen-referenced subtitles*, *world-referenced subtitles*, and *dynamic subtitles*. Each of these categories are described in Subsections 2.4.1, 2.4.2, and 2.4.3 respectively. Table 2.1 contains a summary about the strategies in each category, their advantages and disadvantages.

2.4.1 Screen-Referenced Subtitles

In this category, the subtitles are positioned taking the screen as a reference, which can also be the viewport in an HMD. The subtitles follow the user's view and can be seen at any instant of time. We have identified two

strategies following this category: *static-follow* and *lag-follow*. Each of these strategies is described in the following paragraphs.

When defining the *static-follow* strategy, Brown *et al.* (1) argue that it is a common behavior for showing information in Virtual Reality (VR) experiences, as part of a "head-up display" (HUD). A HUD typically displays graphics that are fixed in front of the viewer at all times regardless of their posture and pose in a VR environment. Figure 2.1 shows this strategy, which uses the aforementioned HUD mechanic. In this strategy, the subtitles are shown to the viewer as if they were static relative to their head, by following the viewer as they look around the environment. The subtitles are placed 15° vertically below eye level. Brown *et al.* (1) mention that a possible caveat of this strategy is that some works have reported that overuse of HUD can cause issues with nausea (34, 35). The work of Meira *et al.* (36) uses this strategy for subtitles positioning. The authors mention that the subtitles are presented at the bottom of the user's viewport and follow their gaze, but they do not mention how many degrees below eye level are used. The work of Matos *et al.* (37) investigates the use of dynamic annotations in 360-degree video, with subtitles being one kind of such annotations. The authors mention the work of Brown *et al.* (1) and call the *static-follow* strategy by *persistent*, in which subtitles (annotations) are placed in front of the user's view. Rothe *et al.* (10) refer to this strategy as *static subtitles*, and say that, in a study they conducted, this was the preferred strategy among the ones proposed by Brown *et al.* (1). They also mention that the subtitles were positioned at 12.5° below eye level. The work of Hughes *et al.* (2) refers to this strategy as *fixed position in the display picture* and mentions that it is the most common way of using subtitles in 360-degree video. Finally, Montagud *et al.* (38) say that there is a follow-up on the work (1) that refers to this strategy as *folow head immediately*. In this follow-up work (a white paper), Brown *et al.* (39) evaluate the four strategies proposed in (1).

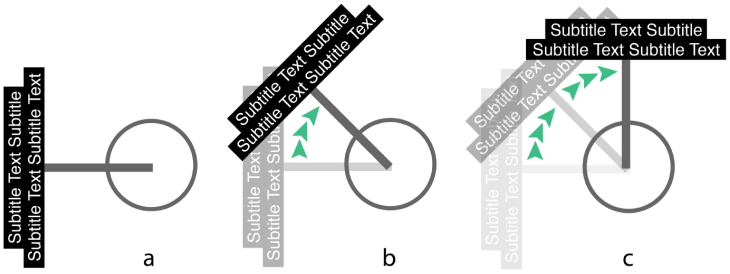


Figure 2.1: Static-Follow: The sequence a, b, c demonstrates how as the user turns their head, the subtitles stay fixed to the centre of their field of view. Extracted from Brown *et al.* (1).

The work of Brown *et al.* (1) defines the *lag-follow* (see Figure 2.2) strategy to address the sickness related to the *static-follow* strategy while still keeping the subtitles visible to the viewer. Similar to the *static-follow* strategy, the subtitles appear in front of the viewer. It remains in such position (relative to the environment) until the viewer's head rotates more than the 30° threshold. The subtitles then smoothly rotate to be in front of the viewer again. The main objective of this strategy is to provide freedom of movement to the viewer without an immediate reaction from subtitles. However, Brown *et al.* (1) say that this strategy may cause the viewer to reread the subtitles, which is not desirable. The work of Matos *et al.* (37) describes a strategy that is the same as this one. It is called *floating*, it starts in a position and floats into the viewer's field-of-view. Similar to what was described on the *static-follow* strategy, the work of Montagud *et al.* (38) refers to this strategy with a different name (*follow with lag*) but having the same definition.

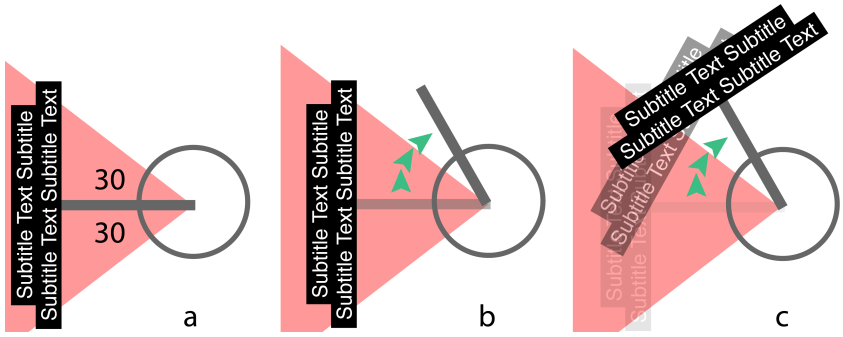


Figure 2.2: Lag-Follow: a. Small user head movements ($< 30°$) are ignored. b. But turning beyond this boundary c. The subtitles move smoothly to the center of the field-of-view. Extracted from Brown *et al.* (1).

2.4.2 World-Referenced Subtitles

In this category, the subtitles are positioned by taking the 360-degree environment as a reference. As referenced in Hughes *et al.* (2), this category of strategy leads to better results in comfort (40). Roth *et al.* (40), as referenced in Hughes *et al.* (2), also say that world-referenced subtitles conflict, in general, with the requirement that a user must always be able to read the subtitle text because it limits the user's freedom of exploring the scene. We have identified two strategies following this category: *repeated subtitles* and *appear subtitles*. These strategies are described in the following paragraphs.

In the *repeated subtitles* strategy, the subtitles are placed around the user. These subtitles stay fixed in the environment and do not follow the user's head motion. Figure 2.3 shows three repeated subtitles evenly spaced by angles of

120°. Such figure was extracted from the work of Brown *et al.* (1). The authors argue that one of the main advantages of this strategy for subtitles positioning is that the subtitles could be "burnt-in" to the video using a video editor. This strategy is referenced as *120-degree* in the work of Brown *et al.* (1), and as *evenly spaced* in the work of Montagud *et al.* (38). A caveat of this strategy, mentioned by Brown *et al.* (1), is that it may cover important content located in unfortunate positions. The work of Li *et al.* (41) uses this strategy to evaluate the impacts of subtitles in 360-degree video journalism. They do not evaluate the subtitles positioning itself, but the impact of the subtitles. The work of Chen *et al.* (42) uses this strategy for positioning subtitles while investigating film language in society news using 360-degree videos of The New York Times. During their study, some participants found the *repeated subtitles* strategy confusing as they thought, in some moments, that the subtitles in different positions had a different text.

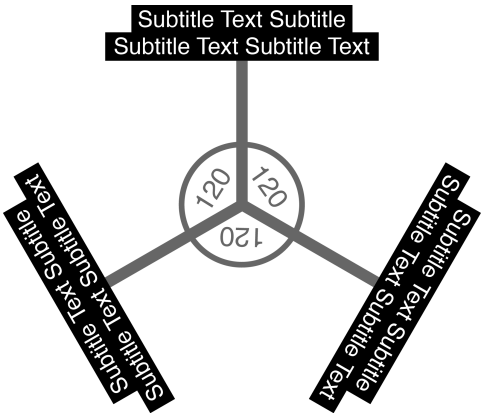


Figure 2.3: Three repeated subtitles (having the same text) are located in the environment at 120° angles around the viewer. Extracted from Brown *et al.* (1).

According to Brown *et al.* (1), the *appear subtitles* strategy was designed based on feedback from a user who was hard of hearing, they had the idea of creating a strategy in which the viewer can dismiss the subtitle after reading it. As it is depicted in Figure 2.4, the subtitles are placed at the center of the user's field of view horizontally, 15° below eye level. If the viewer moves their head, the subtitles remain static within the environment and do not follow their gaze. This strategy is also referenced as *appear in front, then fixed* in the work of Montagud *et al.* (38). A possible caveat of this strategy, mentioned by Brown *et al.* (1), is that the subtitles may be positioned in spurious locations if the viewer is quickly moving their head.

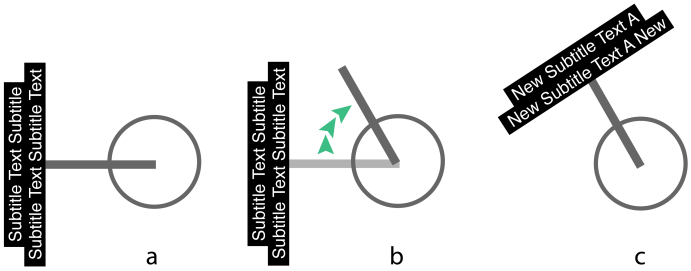


Figure 2.4: Appear: a. A subtitle appears at the centre of the user's view. b. If the user moves before the subtitle is due to change, it will remain static in the environment. c. When a new subtitle is shown, it will appear at the centre of the user's view again. Extracted from Brown *et al.* (1).

2.4.3 Dynamic Subtitles

In this category, the position of subtitles dynamically changes and depends on the scene (10). As referring to annotations (that could be subtitles), Matos *et al.* (37) say that there are cases where the point of interest is moving through the video, which requires a dynamic annotation that follows its movement. One strategy that we have identified in this category is the *speaker-following subtitles* strategy.

In the *speaker-following subtitles* strategy, the subtitles are placed close to the speaker (see Figure 2.5). Since the speakers may move during the video, this strategy fits in the *dynamic subtitles* category. This strategy also helps in the issue of *speaker identification*, as all persons in the room are visible in a 360-degree video (10). Rothe *et al.* (10) compared *speaker-following* subtitles with the *static-follow* strategy regarding task workload, simulator sickness and presence. For evaluating each of these dimensions, the authors used, respectively, the following questionnaires: NASA-TLX (43); Simulator Sickness Questionaire (44); and Presence Questionaire (45). When asking which strategy the participants preferred, the authors received balanced answers. However, *speaker-following* subtitles led to a higher score of presence, less sickness, and lower workload (10).

Figure 2.5: Speaker-following subtitles. Extracted from Hughes *et al.* (2).

Similar to the work of Rothe *et al.* (10), we intend to position subtitles close to the speakers in the 360-video. The main difference of our work, however, is that we automatically detect the actors present in a 360-video and use their position for placing the subtitles according to an authoring model we propose. In our authoring model, we can also determine the direction that the user is looking at. We use that to position the subtitles as *static-follow* when the actor speaking is not visible to the user. In this way, the disadvantage of *speaker-following* subtitles being not always visible is suppressed.

Table 2.1: Subtitles positioning strategies catalog for 360-degree video

Category	**Strategy**	**Advantages**	**Disadvantages**
Screen-Referenced	Static-Follow	easy to locate; freedom of movement; most common strategy;	issues with nausea;
	Lag-Follow	issues with nausea mitigated in comparison to static-follow;	may cause rereading;
World-Referenced	Repeated Subtitles	comfort; could be "burnt-in" the video;	may cover important content; may be confusing; not always visible;
	Appear	comfort; subtitles can be "dismissed";	may be positioned in spurious locations; not always visible;
Dynamic	Speaker-Following	help in speaker identification;	not always visible;

3
A Method For Video Face Clustering Method

This chapter describes the core of this dissertation, which is a method for *Video Face Clustering*. It consists of detecting and grouping faces from different video frames (ideally from the same actors) extracted from a video file. Figure 3.1 depicts this process, and each of its steps is described in the next paragraphs.

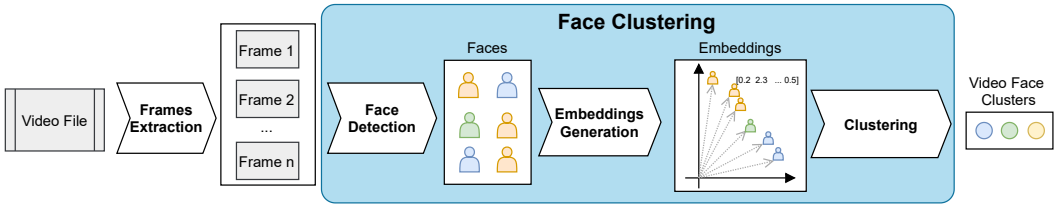


Figure 3.1: Video face clustering process.

First, we perform *Frames Extraction* by receiving a video file as input and extracting its frames according to a given frame rate. These frames are used as a set of images for the next step.

The *Face Detection* step uses an object detection model for detecting faces in each of its images. In general, any object detection model can identify which, among a known set of objects, are present in the image, and provides information about their positions. In our case, objects are faces and, therefore, the face detection model is responsible for returning the bounding boxes of the faces present in the image, specified by the x and y axes coordinates of the upper-left corner and of the lower-right corner of the rectangle that establishes the visual limits that encapsulate each face. With these bounding boxes, we can isolate and extract the bounded images, obtaining a dataset composed of images of faces only.

The objective of the *Embeddings Generation* step is to represent each face image as a vector in $\mathbb{R}^n$. To achieve that, it processes each of the faces generated in the previous step through a CNN, generating embeddings. An embedding is a representation of the input in a lower-dimensional space. Ideally, an embedding captures some semantics of the input, e.g. by placing semantically similar inputs close together in the embedding space. At the end of this step, we have all faces represented as embeddings.

In the *Clustering* step, we group embeddings and, consequently, faces that are close in the embedding space using a clustering algorithm. Clustering is the task of dividing a set of data points, embeddings in this case, into groups (clusters) such that data points in a given group are similar to other data points in the same group and dissimilar to the data points in other groups. The clustering process should produce a partition of the dataset, i.e., each generated cluster represents a specific person, and the union of all clusters covers the whole dataset. It is common for clustering algorithms to require the number of clusters in advance. However, in the context of *Video Face Clustering*, we do not know this number in advance, that is the number of actors in the video. For that reason, we have designed a strategy for automatically choosing an adequate number of clusters. Section 3.1 describes this strategy.

3.1 Iterative Strategy for Unknown Number of Clusters

In this Section, we define a strategy based on the *Silhouette Score* (s) (46) to choose an adequate clustering configuration. The *Silhouette Score* (46) corresponds to the mean of the *Silhouette Coefficient* of all samples. This coefficient (S) for each sample is

$$S = \frac{b - a}{max(a, b)} \tag{3-1}$$

where a is the mean distance from a sample to all other samples in the same cluster, and b is the mean distance from a sample to all other samples in the closest cluster to that sample. In this way, the best value is 1 and the worst is -1. Values close to 0 indicate overlapping clusters, whereas negative values usually indicate that a sample has been assigned to the wrong cluster since a different cluster is more similar.

With this strategy (see Algorithm 1), we increase the number of clusters until the maximum *Silhouette Score* decreases no more than t times in a row or until it reaches the maximum number of clusters (lines 5-18), which is the number of embeddings ($|e|$). The `Clustering` procedure (line 7) can be substituted by any clustering algorithm that requires the number of clusters in advance. When the iteration stops, we return the clustering configuration with the highest *Silhouette Score*. Since this score requires at least two clusters, it would not be possible to compute it for a clustering configuration with only one cluster (there are only faces of a single actor). To overcome this problem, we start with 2 clusters consecutively increasing it as described above. Then, if the returned clustering configuration has a *Silhouette Score* smaller than a

threshold ω, that probably indicates overlapping, we say that all faces belong to one single cluster (lines 19-20).

Algorithm 1 Iteratively finding the best clustering configuration for unknown number of clusters.

```
1:  procedure BlindClustering(e, t, ω)
2:      n_K ← 1
3:      s_max ← −1
4:      t_cur ← 0
5:      while t_cur ≤ t & n_K < |e| do
6:          n_K ← n_K + 1
7:          K_cur ← Clustering(e, n_K)
8:          s ← SilhouetteScore(K_cur)
9:          if s < s_max then
10:             t_cur ← t_cur + 1
11:         else
12:             K ← K_cur
13:             t_cur ← 0
14:             if s > s_max then
15:                 s_max ← s
16:             end if
17:         end if
18:     end while
19:     if s_max < ω then
20:         K ← OneCluster(e)
21:     end if
22:     return K
23: end procedure
```

By the end of this process, we expect to have the spatio-temporal localization of the actors present in a video file. Figure 3.2 shows an example of the results achieved by our *Video Face Clustering* method. It contains a timeline of a video with lines coloring the segments that each actor appears in.

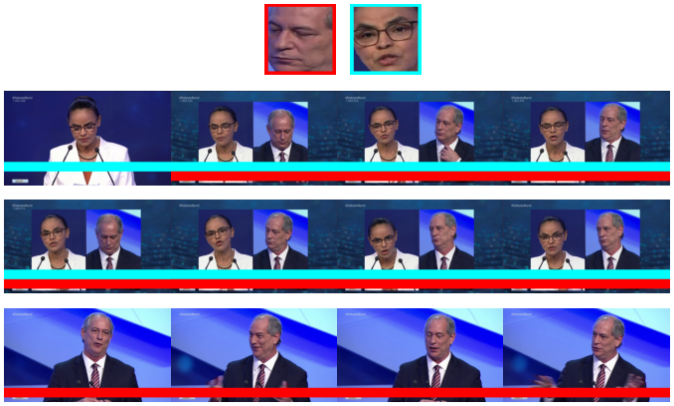


Figure 3.2: Example of spatio-temporal localization.

The following three chapters describe the applications we investigate in this dissertation. These three applications propose novel approaches for tasks in video using spatio-temporal localization of actors through *Video Face Clustering*.

4
Cluster-Matching-Based Method For Video Face Recognition

In this chapter, we describe the first application we investigated using *Video Face Clustering*. We propose a cluster-matching-based approach for video face recognition where *face clustering* is used to group faces in both the face dataset and in the target video (*video face clustering*). Consequently, classes do not have to be previously known, and the effort spent with annotations is significantly reduced — as it is done over clusters instead of single images. Face recognition becomes a task of comparing clusters from the dataset with the ones extracted from images or video sources. Therefore, our approach is easily scalable.

This chapter is structured as follows. In Section 4.1, we describe the dataset we produced to perform experiments. In Section 4.2, we detail our proposed method, followed by two sections with experiments: Section 4.3, regarding the clustering methods we use, and Section 4.4, with the experiments with our matching heuristic. Section 4.5 is devoted to the overall evaluation of our method and, finally, in Section 4.6, we conclude this chapter by discussing our results.

4.1
Dataset

Our dataset was collected using the information provided by the Brazilian Chamber of Deputies[1] for the 55th legislature, which was in session from February 1st, 2015 through January 31st, 2019. In total, 623 different deputies participated during some period in the 55th legislature, but we collected only the initial 513 deputies.

For each of the 513 deputies, we collected images in which he/she was present using Google images. The resulting dataset has a total of 9,003 images, with a mean of 17.55 images per deputy and a standard deviation of 6.91. The maximum and minimum number of images per deputy are respectively 32 and 2. We have randomly selected one image of each deputy for validation and the rest for training. Thus, our dataset is divided into two: the training set, with 8,490 images, and the validation set, with 513 images.

[1] https://www.camara.leg.br/

Besides, we created another set containing images of people who are not present in the deputies set. For that, we randomly selected 513 images from the *Labeled Faces in the Wild* (LFW) (47) dataset, that contains more than 13,000 images of faces collected from the web, and defined this subset of LFW as our *Non-registered people* set.

4.2 Proposed Method

Our method intends to recognize people in video using *Video Face Clustering* and a matching heuristic. Figure 4.1 shows our proposed approach.

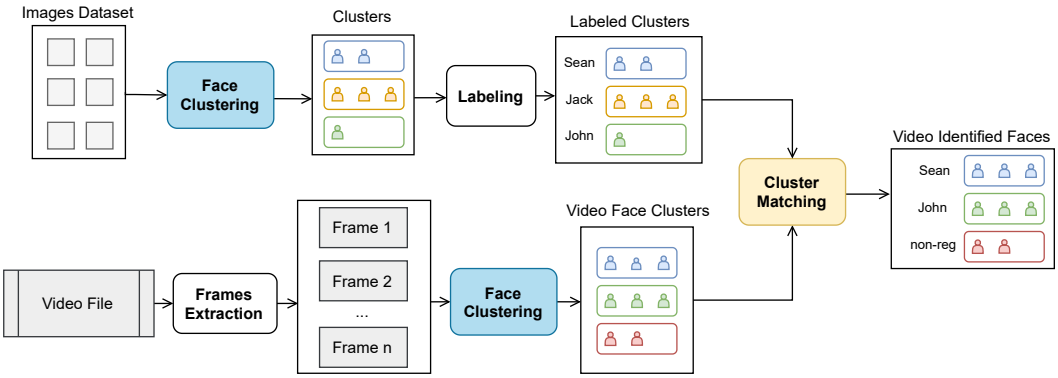


Figure 4.1: Cluster-Matching based Method for Video Face Recognition.

In our approach, we use *Face Clustering* in an images dataset and in the referenced video. Then, the clusters of the images dataset are labeled. In the *Labeling* step, we assign labels (identities) to represent the clusters. A label can be anything that represents the faces present in the cluster, e.g. a name or an id number. Using this pipeline, instead of having to label every single face for constructing a labeled dataset, it is only necessary to label each generated cluster. Consequently, all the faces present in a cluster are assigned to the same label. Hence, the complexity of labeling becomes a function dependent on the number of clusters, which is at most as great as the number of individuals. At the end of this step, we have a dataset of *Labeled Clusters*.

Next, we perform *Cluster Matching* where the clusters from the image dataset and from the video are matched using a heuristic based on clusters distance. The *Cluster Matching* step receives the set of clusters from the video and the set of labeled clusters, which is used as a reference for recognizing the clusters (and consequently the faces) in the video. We designed a method based on cluster distance for performing this recognition. We compare each candidate cluster in the video with each of the labeled clusters in the reference dataset, using what we call a *cluster embedding*. This *cluster embedding* is computed for both the candidate cluster in the video and the reference cluster in the

labeled dataset. The *cluster embedding* maps each cluster to a vector in the embedding space of the face embeddings. In Section 4.4, we evaluate two ways of obtaining a *cluster embedding*.

Let q denote the *cluster embedding* of the query cluster, where $q \in \mathbb{R}^n$, in which n is the dimension of the embedding space. Let K denote the set of labeled clusters. For each $k \in K$, let c_k denote the *cluster embedding* of k. We compute a similarity function of q and each c_k for $k \in K$. This similarity function is based on the Root Mean Square Error (RMSE) function, which is largely used for computing embeddings distances in machine learning techniques.[2] We decided to use the inverse of the RMSE since we want to return larger values for clusters that are closer to q, to be able to use these values to compute a probabilistic distribution:

$$s_{q,k} = \frac{1}{\sqrt{\frac{1}{n}\sum_{i=0}^{n}(q_i - c_{k,i})^2}} \tag{4-1}$$

Since we are not interested in clusters that are too distant from q, we define $\overline{s}_{q,k}$ that considers only the α largest s_q values. The α value is a parameter that we further evaluate in this chapter. Thus, our similarity function becomes

$$\overline{s}_{q,k} = \begin{cases} s_{q,k} & \text{if } s_{q,k} \in max_{\alpha(s_q)} \\ 0 & \text{otherwise} \end{cases} \tag{4-2}$$

Given the similarity $\overline{s}_{q,k}$, we compute the probability of q being a match with each labeled cluster using the *softmax* function. This function takes as input a vector of real numbers and normalizes it into a probability distribution consisting of probabilities proportional to the exponential of the input numbers. It is defined as

$$p_{q,k} = \frac{e^{\overline{s}_{q,k}}}{\sum_{j \in K} e^{\overline{s}_{q,j}}} \tag{4-3}$$

Finally, we define the function σ that, given a *cluster embedding* query q, returns the cluster in the labeled clusters whose *cluster embedding* is more likely to have a match with the query if it has a probability greater than 0.5. Otherwise, the query is assigned as being a match with none of the clusters (ø is given as result). The σ function is defined as follows.

$$\sigma(q) = \begin{cases} argmax(p_q) & if \quad p_{q,argmax(p_q)} > 0.5 \\ \text{ø} & otherwise \end{cases} \tag{4-4}$$

The $argmax$ function applied over p_q returns the cluster k whose probability of q being a match with it is the greatest. We observed that when a

[2]https://www.sciencedirect.com/topics/engineering/root-mean-square-error

cluster has faces that belong to a person present in the labeled clusters, the probability tends to be higher for one single labeled cluster. However, when the person is not in any of the labeled clusters, the probability tends to be more distributed among different labeled clusters. For that reason, when none of the labeled clusters has a probability greater than 0.5, we say that the query cluster does not match any of the labeled clusters. Hence, the person in the video represented by the query cluster is not present in the labeled clusters.

4.3 Faces Clustering Validation

In this section, we evaluate the quality of the clustering generated in the macro-step *Face Clustering*. In the *Face Detection* step, we use MTCNN (48) (Multitask Cascaded Convolutional Networks) which is widely used for the face detection task (49, 50). We used the *mtcnn* Python library for its implementation[3] and resized each face detected to 224x224. We tested three different CNNs for the *Embeddings Generation* step in association with three different clustering algorithms for the *Clustering* step. For that, we evaluated each pair of *CNN x ClusteringAlg* through the quality of the clusters they generated. To experiment, we used the training set with 8490 deputy images.

In what follows, we describe the algorithms used for the experiments, the metrics used to evaluate, and the results we obtained.

4.3.1 Embeddings Generation

For this step, we have three candidate CNNs that were previously trained on the VGGFace2 dataset (22). The VGGFace2 dataset contains 3.31 million images of 9131 subjects and has large variations in pose, age, illumination, ethnicity, and profession. The three candidate CNNs used are VGG-16 (51), ResNet-50 (52) and SE-ResNet-50 (53) (SeNet-50 for short). VGG-16 generates embeddings in the $\mathbb{R}^{512}$ feature space, while ResNet-50 and SeNet-50 generate embeddings in the $\mathbb{R}^{2048}$ feature space.

4.3.2 Clustering Algorithms

For this step, we selected the following clustering algorithms as candidates: K-Means (KM) (54), Affinity Propagation (AP) (55), and Agglomerative Clustering (AC) (56). Each of these algorithms is briefly explained next.

[3]https://pypi.org/project/mtcnn/

K-Means (54) is one of the most widely used unsupervised machine learning algorithms. To process the data to be clustered, the K-Means algorithm begins with a randomly selected group of K centroids, which are updated iteratively to optimize the distance of the data points to the closest centroid. The algorithm stops when either the centroids have stabilized or the maximum number of iterations has been reached.

The Affinity Propagation algorithm (55), in contrast to other clustering methods, does not require the number of clusters to be previously specified. In this algorithm, each data point sends messages to the other points informing them of their relative attractiveness to the sender. Those targets then reply to the senders informing their availability to associate with them, considering the attractiveness of all the messages they received. The senders then reply to the targets informing the target's updated relative attractiveness. This process continues until a consensus is established. When the sender data point is associated with one of its targets, that target becomes the point's exemplar. The points with the same exemplar are assigned to the same cluster.

The Agglomerative Clustering algorithm recursively merges the pair of clusters that minimally increase a given linkage distance. The linkage criteria specify the distance to use between two sets of data points. The Agglomerative Clustering algorithm (56) merges pairs of clusters that minimize the linkage criteria. In this work, we chose the *Ward* criteria (56), which minimizes the variance of the clusters being merged. By using this method, at each step, the algorithm finds the pair of clusters that leads to a minimum increase in total within-cluster variance after merging. This increase is measured by a weighted squared distance between cluster centers. In the first step, each data point is a cluster. The clusters are merged following the criteria until the number of clusters K is reached.

Different from the Affinity Propagation algorithm, K-Means and Agglomerative Clustering require the number of clusters in advance. For this case, we used 513 (number of deputies in the *train set*). However, when the number of clusters is not known, we use the strategy defined in Section 3.1.

4.3.3 Metrics

We evaluate the models using the V-Measure (57), which is an entropy-based measure that computes how successfully the criteria of homogeneity and completeness have been satisfied. This metric is extensively used for comparing clustering solutions and has been used in different domain fields such as biology (58), computational linguistics (59), and document engineering (60).

The V-Measure is a harmonic mean of homogeneity and completeness scores, similar to how precision and recall are frequently combined into F-measure (61). It assumes a dataset comprising N data points, and a set of classes, $C = \{c_i | i = 1, ..., n\}$ and a set of clusters, $K = \{k_i | i = 1, ..., m\}$. They also assume A as a matrix produced by the clustering algorithm representing the clustering solution, such that $A = \{a_{ij}\}$ where a_{ij} is the number of data points that are members of class c_i and elements of cluster k_j.

The homogeneity is perfect when a clustering algorithm assigns only those data points that are members of a single class to a single cluster so that the entropy is zero in each cluster. In this way, the homogeneity score determines how close a given clustering is to this ideal by examining the conditional entropy of the class distribution given the proposed clustering. Therefore, when the clustering is perfectly homogeneous, such a value, $H(C|K) = 0$.

On the other hand, when the clustering is not perfect according to this criterion, this value is proportional to the size of the dataset and the classes. For this reason, the authors normalize this value by the maximum reduction in entropy the clustering information could provide, specifically, $H(C)$. $H(C|K)$ has its maximal value when it is equal to $H(C)$ and provides no new information.

Finally, to address to the convention of 1 being desirable and 0 undesirable, the homogeneity is defined as:

$$h = \begin{cases} 1 & \text{if } H(C|K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{otherwise} \end{cases} \tag{4-5}$$

where

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \tag{4-6}$$

$$H(C) = -\sum_{c=1}^{|C|}\frac{\sum_{k=1}^{|K|} a_{ck}}{n} log \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \tag{4-7}$$

Completeness is symmetrical with respect to homogeneity, and it is perfect when a clustering assigns all data points that are members of the same class to a single cluster. For computing such a score, the authors examine the distribution of cluster assignments within each class. Completeness is defined as

$$c = \begin{cases} 1 & \text{if } H(K|C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{otherwise} \end{cases} \tag{4-8}$$

where

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{ck}}{N}log\frac{a_{ck}}{\sum_{k=1}^{|K|}a_{ck}} \tag{4-9}$$

$$H(K) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|}a_{ck}}{n}log\frac{\sum_{c=1}^{|C|}a_{ck}}{n} \tag{4-10}$$

Finally, by using Equation 4-5 and Equation 4-8, V-measure is defined as

$$V_\beta = (1+\beta)\frac{h \cdot c}{\beta \cdot h + c} \tag{4-11}$$

The parameter β is used to calibrate the relative importance of homogeneity and completeness when computing the V-measure. If β is greater than 1, the completeness is more important for the V-measure, whereas if β is less than 1, homogeneity is more important for the V-measure. In this work, we use $\beta = 1$, so that completeness and homogeneity contribute equally to the V-measure.

4.3.4 Results

Table 4.1 shows the homogeneity, completeness, and V-Measure for each combination of CNN and clustering algorithm.

From Table 4.1, one can conclude that the best combination of CNN and clustering algorithm was SeNet-50 with Agglomerative Clustering. For this reason, we decided to use SeNet-50 for the *Embeddings Generation* step and the Agglomerative Clustering algorithm for the *Clustering* step. As a result, each face on the dataset is represented as an embedding in the $\mathbb{R}^{2048}$ produced by SeNet-50.

One can observe that the Affinity Propagation algorithm had a V-measure of 0 when used with ResNet-50 and VGG-16. This happens because, in both cases, the algorithm assigned all the data points (face embeddings in this context) to one single cluster. Consequently, those combinations had a homogeneity score of 0 because there was no information gain after clustering– and completeness of 1. After all, data points of the same class were not scattered in different clusters. With such a combination, the V-measure, which is a harmonic mean of homogeneity and completeness, has a value of 0.

Table 4.1: Results of the evaluation of the clusters created by each combination of CNN and clustering algorithms.

CNN	**Clustering**	h	c	V_1
ResNet-50	KM	0.9665	0.9675	0.9670
	AP	0.0000	1.0000	0.0000
	AC	0.9821	0.9798	0.9810
SeNet-50	KM	0.9725	0.9726	0.9725
	AP	0.9859	0.9558	0.9706
	AC	**0.9862**	**0.9833**	**0.9847**
VGG-16	KM	0.8340	0.8415	0.8378
	AP	0.0000	1.0000	0.0000
	AC	0.8899	0.8929	0.8914

4.4 Cluster-Matching Validation

Based on the results of the experiment described in Section 4.3, we developed another experiment in order to choose the best α value in Equation 4-2 and the best method for computing the *cluster embedding* for the clusters.

As the best combination for clustering was SeNet-50 with the Agglomerative Clustering algorithm, this experiment uses the embeddings and labeled clusters generated by it. Hence, the labeled clusters are the set of faces from the 8490 deputy images with their respective embeddings and labels. Each cluster was labeled with the name of the deputy whose face is more frequent in it.

For testing, we used the *Validation set* (V) consisting of 513 deputy images and the *Non-registered people set* (U) also consisting of 513 images. We used these images as if they were the clusters from a video file, each cluster having a single sample. In this way, we have a more heterogeneous set for performing this experiment as if we had used a small set of video files.

We evaluate two methods for computing a *cluster embedding* for the labeled clusters: *cluster centroid* and *sample with best silhouette coefficient*. A cluster centroid denotes the mean of the elements in a cluster. It is calculated the mean of all elements in a cluster for each axis in the embedding space. The second method uses the embedding of the sample whose silhouette coefficient is the highest. This coefficient is calculated using the mean intra-cluster distance and the mean nearest-cluster distance for each sample (detailed in Subsection 3.1).

4.4.1 Metrics

We evaluate the two methods for computing the *cluster embedding* by assessing how well the cluster matching recognizes registered people and correctly tells when a person is not registered. A non-registered person should not have a match with any of the labeled clusters.

Let $\lambda(q)$ be the most frequent label of a cluster q, $|V|$ the size of the *Validation set* and $|U|$ the size of the *Non-registered people set*. Although λ represents the label of a cluster in general, we also define Λ as being the set of all the face labels present in a cluster. Let us also assume true as 1 and false as 0. To perform the evaluation, we compute the following four metrics:

$$m_1 = \sum_{v \in V} \frac{\sigma(v) \neq \emptyset}{|V|} \quad (4\text{-}12) \qquad m_2 = \sum_{v \in V} \frac{\lambda(v) = \lambda(\sigma(v))}{|V|} \quad (4\text{-}13)$$

$$m_3 = \sum_{v \in V} \frac{\lambda(v) \in \Lambda(\sigma(v))}{|V|} \quad (4\text{-}14) \qquad m_4 = \sum_{u \in U} \frac{\sigma(u) = \emptyset}{|U|} \quad (4\text{-}15)$$

Where m_1 denotes the percentage of clusters from V matched with any cluster, $m2$ the percentage of clusters from V matched with a cluster with the same label, $m3$ the percentage of clusters from V matched with a cluster in which the label is present, and $m4$ that denotes the percentage of clusters from U that have not matched with any cluster.

4.4.2 Results

In this subsection, we show the results for each of the methods for generating the *cluster embeddings* in the *cluster matching* step. For each of the methods, we tested different α values in Equation 4-2.

Table 4.2 shows the results obtained using the cluster centroids as *cluster embeddings*. It can be seen that the higher the α value is, the lower is the percentage of registered faces assigned to the correct labeled cluster. On the other hand, the percentage of non-registered faces assigned to none of the labeled clusters increases with the α value. One can see that with $\alpha = 5$, we have a balance between assigning people to the correct cluster and being able to tell when a person is not in the labeled clusters.

Table 4.3 shows the results obtained using the sample with the highest silhouette coefficient as the *cluster embedding* for each of the labeled clusters. One can observe that the α value correlates with the percentage values similar to the one observed in Table 4.2. By using the silhouette coefficient for choosing

Table 4.2: Results obtained using cluster centroids as *cluster embeddings*

α	**m1**	**m2**	**m3**	**m4**
2	100.000%	94.932%	98.246%	0.000%
3	97.076%	94.152%	96.881%	94.542%
4	95.517%	93.177%	95.322%	98.635%
5	**94.542**%	**92.398**%	**94.347**%	**99.220**%
6	93.567%	91.813%	93.372%	99.610%
7	92.788%	91.228%	92.593%	99.805%
8	91.813%	90.643%	91.618%	99.805%
9	90.448%	89.279%	90.253%	99.805%
10	89.474%	88.304%	89.279%	99.805%

the *cluster centroids*, we have an algorithm more capable of determining when a person is not registered, achieving a percentage of 100% when $\alpha > 5$. However, when using this approach, the algorithm fails to correctly assign registered people to the correct labeled cluster in comparison to when it uses the cluster centroids.

Table 4.3: Results obtained using samples with the highest silhouette coefficient as *cluster embeddings*

α	**m1**	**m2**	**m3**	**m4**
2	100.000%	93.177%	94.347%	0.000%
3	86.355%	85.575%	86.160%	98.246%
4	76.998%	76.608%	76.998%	99.415%
5	72.904%	72.515%	72.904%	99.805%
6	68.811%	68.421%	68.811%	100.000%
7	64.912%	64.522%	64.912%	100.000%
8	61.209%	60.819%	61.209%	100.000%
9	60.039%	59.649%	60.039%	100.000%
10	58.285%	57.895%	58.285%	100.000%

4.5 Video Face Recognition Evaluation

To evaluate our complete pipeline, we selected videos that contain only registered people (videos a to d), videos with both registered and non-registered people (videos e to i) and videos with only non-registered people (videos j to m). The labeled clusters are the same used in Section 4.4, which were generated using MTCNN (48) for detecting faces, SeNet-50 for extracting its embeddings, and the Agglomerative Clustering algorithm to cluster them. The label of the clusters corresponds to the name of the deputy whose face is more frequent in it.

For performing the face identification on a video file, we first extract its frames using a frame rate of 1fps. For each of the frames, we extract the faces

present on it using MTCNN (48). Then, for each face identified, we resize it to 224x244, extract its embedding using SeNet-50, and cluster these faces using the Agglomerative Clustering algorithm. Since we do not know the number of people present in the video, we use the strategy described in Subsection 3.1 with the maximum time of sequential increases $t = 5$, which was empirically determined to be a good stop point.

Next, we perform the *Cluster Matching* with the labeled clusters and the video face clusters. We used cluster centroids as *cluster embeddings* and a $\alpha = 5$ as the results of Section 4.4 show that this configuration can correctly match the clusters while preserving the capacity of distinguishing non-registered faces. At the end of this process, each face present on the video is labeled either with the name of a registered person or as non-registered.

We evaluate our method by the Precision (Prec), Recall (Rec), and F1-Score for the faces in the video. As usual, Precision is defined as the percentage of detected faces that our method correctly labels, the Recall gives the percentage of faces that our method correctly labels among all faces in the video, and the F1-score represents an overall performance metric based on the harmonic mean of the precision and recall. We also count the number of exact frames match (#EM). It corresponds to the number of frames (#F) for which our method correctly labeled all the faces that appear. Table 4.4 shows the results we obtained, the number of different people in each video ($\#P$), and the number of people who are present in the labeled clusters ($\#R$).

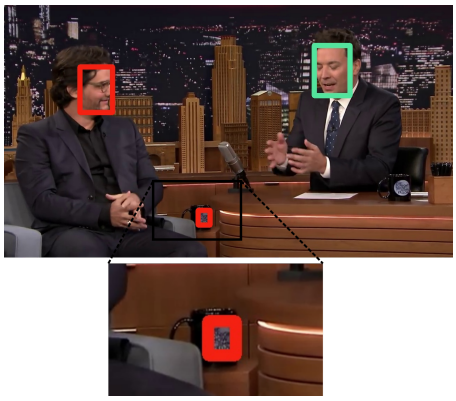
Table 4.4: Results using the proposed approach in videos.

Video	**#P**	**#R**	**#F**	**#EM**	**Rec.**	**Prec.**	**F1**
videos with only registered people							
a	1	1	105	105	100.000%	100.000%	100.000%
b	1	1	80	79	100.000%	98.765%	99.379%
c	1	1	60	59	100.000%	98.113%	99.048%
d	2	2	226	226	100.000%	100.000%	100.000%
videos with both registered and non-registered people							
e	2	1	101	99	100.000%	98.113%	99.048%
f	2	1	650	650	100.000%	100.000%	100.000%
g	8	1	201	190	96.471%	96.850%	96.660%
h	4	1	231	215	98.097%	98.514%	98.305%
i	2	1	88	88	100.000%	100.000%	100.000%
videos with only non-registered people							
j	6	0	625	617	99.551%	99.551%	99.551%
k	2	0	231	228	100.000%	98.872%	99.433%
l	2	0	131	121	99.482%	95.522%	97.462%
m	1	0	225	222	99.556%	99.115%	99.335%

One can observe that our method can correctly classify the faces of people

that are present in the labeled clusters while also being able to tell when a person is not registered in the labeled clusters. However, Table 4.4 shows that the precision was smaller for some videos. Analyzing these videos, we observed that this result was mostly due to some non-face objects that were detected as if they were faces. In *Video l*, for instance, in some frames, a mug was detected as if it was a face (Figure 4.2b). On the other hand, in *Video g*, which had the lowest recall, one face was not detected in some frames because it was partially covered by a text (Figure 4.2a).

(a) Example where a face is not detected because it is covered



(b) Example where a mug is detected as if it was a face.

Figure 4.2: Cases where our approach was incorrect.

Similar to the work of Pena *et. al* (25), our method can be used to generate metadata in video files indicating the people that appear in it. Figure 4.3 shows the first 12 seconds of *Video d* (detailed in Table 4.4) where two identified Brazilian politicians are shown in each frame with its respective colored cluster.

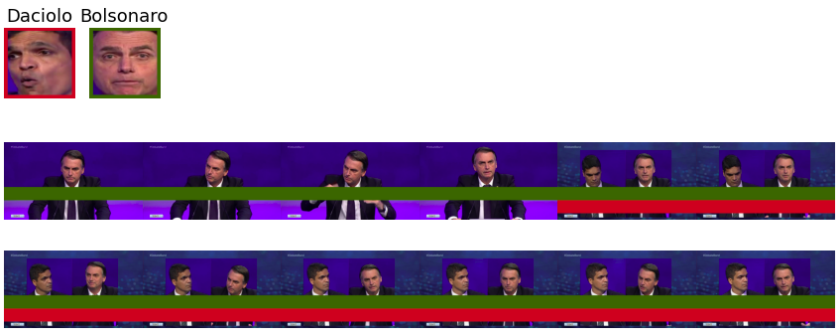


Figure 4.3: Timeline with tagged frames by their clusters of registered people

Besides being able to recognize people in video files, by using face embeddings and clustering, we can detect the frames where the same person appears without even knowing who the person is or if he/she is in the labeled clusters. Figure 4.4 shows the first 24 seconds of *Video j* (detailed in Table 4.4)

with the frames tagged with the clusters identified in each frame, where each color represents a cluster.

Figure 4.4: Timeline with tagged frames by their clusters of non-registered people

4.6 Discussion

In this chapter, we proposed a cluster-matching-based method for video face recognition. This method uses *Video Face Clustering* and a heuristic for cluster matching to recognize people in videos. Our method has achieved a recall of 99.435% and precision of 99.131% when considering all faces present in the frames extracted from a set of 13 video files.

As a consequence of face clustering, our work also demonstrates that this technique can be useful for labeling datasets in a less time-consuming way, where clusters are labeled instead of individual data points. Although this work focuses on faces, the method proposed can be applied to other domain fields. The only requirement is that the designer has to conceive a technique in which objects of the same class are placed closed to each other in the corresponding vector space.

One of the limitations of this work is related to the size of the video set used for the overall evaluation of our method. This is due to the difficulty of finding videos where it is possible to manually identify in each frame whether each person present is registered or not in our labeled clusters. Another limitation is that the proposed method was not able to detect partially covered faces. This is due to a limitation of the MTCNN (48), which is currently not robust enough to deal with this condition. The results of this application were published at relevant multimedia conference (62).

5
Face-Clustering-Based Method for Educational Video Recommendation

It is a common practice among educational content creators to make collaborative videos, that is, videos in which more than one lecturer is presenting the lecture content. Such collaborations create a network of lecturers teaching a given subject. Therefore, a method that identifies these collaborations may help students find their content of interest more easily. In this chapter, we describe the second application we investigated using our *Video Face Clustering* method. We propose a recommender method based on the actor's presence for educational videos. In this case, the actors are lecturers (or teachers, professors, etc.) that are presenting educational content on video. For instance, if a student watches a video containing lecturers A and B, our method aims at recommending other videos that contain at least one of these lecturers. This method provides an additional aid for educational recommender systems, allowing them to use the presence of lecturers as a feature for composing their recommendations.

The remainder of this chapter is structured as follows. Section 5.1 presents the dataset we used. We present our method in Section 5.2, followed by Section 5.3, that shows the experiments to evaluate our recommendation ranking mechanisms. Finally, in Section 5.4, we conclude this chapter by discussing our results.

5.1
Dataset

The experiments were conducted using a dataset created in the context of this application. It is composed of 98 educational videos publicly available on YouTube.[1,2] Each video contains at least one lecturer; moreover, some videos could have some special participation or collaboration. Therefore, the number of lecturers in each video varies from 1 to 5. In total, 16 different lecturers are present in the dataset. Each lecturer is present, on average, in 6.67% of the

[1]https://www.youtube.com/channel/UCT0JugAtGmqiYkwxFZOwAtg
[2]https://www.youtube.com/user/deboraaladim

videos. We also annotated the name of each lecturer present in the videos, so that we can verify if correct videos are being recommended.

The videos duration vary from 00m:30s to 1h:49m:01s. The average duration of the videos is 23m:34s, with a standard deviation of 23m:05s. The high value of the standard deviation for the time estimates indicates that the videos are not in the same time range, and therefore have a wide duration variety.

5.2 Proposed Method

Our method intends to recommend educational videos based on the lecturers that appear in each video, so that, when a person watches a video, other videos containing the same lecturers are recommended. To do that, we first represent each video in the dataset of educational videos by the clusters of lecturers present using *Video Face Clustering*. Once we have all videos represented, we perform the pipeline depicted in Figure 5.1. Each of the steps of this pipeline is described in the remainder of this section.

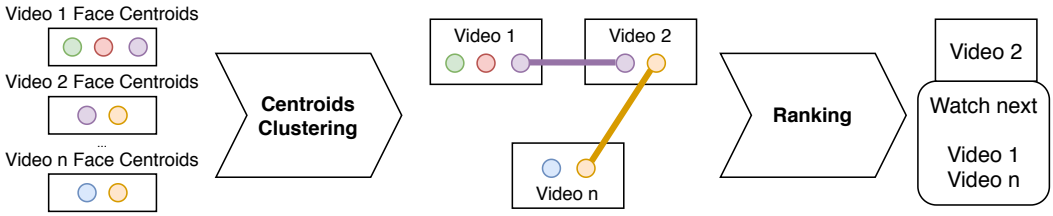


Figure 5.1: Video Recommendation based on Lecturers Centroids Clustering

First, we gather the centroids from the videos of the dataset as one single set and perform the *Centroids Clustering*. To perform this clustering, we also use the strategy for an unknown number of clusters described in Section 3.1. By doing that, we group centroids from the same lecturer that are in different videos. For instance, in Figure 5.1, one can see that the *purple lecturer* is present in both Videos 1 and 2, while the *orange lecturer* is present in both Videos 2 and n. By the end of this step, we have the group L of lecturers present in the dataset of videos V. We also denote L_v as the group of lecturers present in video v.

Next, based on these relationships among different videos, we perform *Ranking*, by recommending videos in which lecturers of the current video are present. For doing that, we compute a similarity score using the presence of the lecturers in the current video and the presence of these same lecturers in the other video. Let $p_{l,v}$ denote the percentage of frames in which the lecturer

$l \in L_v$ is present in video $v \in V$. For each video $v \in V$ and $u \in V - v$ we compute a score of similarity $S_{v,u}$.

$$S_{v,u} = \sum_{l \in L_v} p_{l,v} \cdot p_{l,u} \tag{5-1}$$

Finally, using this score, for each video v we compute a ranking R_v where $R_{v,i}$ denotes the *i-greatest* S_v and $R_{v,i} \geq R_{v,i+1}$ for all $i \in 1...n_v$, where n_v is the number of videos u in which $S_{v,u} > 0$. In this way, the more lecturers a video has in common with the reference video, and the more time these lecturers are present in both videos, the higher the video is positioned in the ranking of the reference video.

By the end of this phase, we have a ranking of recommended videos for each video in the dataset. It is important to notice that our method is unsupervised and does not require the information of the lecturers in advance. Consequently, we do not have to store any information regarding the identity of the lecturers, respecting their privacy.

5.3 Ranking Evaluation

This section describes an experiment we conducted for evaluating how well our recommendation approach performed in terms of recommending relevant videos. We define that a recommendation is relevant if the reference and the recommended video have at least one lecturer in common.

First, we compute the clusters that represent each lecturer in each video using *Video Face Clustering*. We start by performing *Frames Extraction* for each video file using a frame rate of 1 frame per second (fps). Next, in the *Face Detection* step, we use MTCNN (48). Once we have detected the faces of lecturers in the video frames, we perform *Embeddings Generation* using SeNet-50 (53) that generates embeddings on the $\mathbb{R}^{2048}$ feature space. We used the architecture and weights pre-trained on the VGGFace2 dataset (22). Finally, in the *Clustering* step, since we do not know the number of clusters in advance, we used the strategy described in Section 3.1 with $t = 5$, $\omega = 0.2$, that empirically demonstrated good results. For the `Clustering` procedure, we used Agglomerative Clustering (56). The choice of using SeNet-50 and Agglomerative Clustering comes from the fact that this combination achieved the best results in the experiment described in Section 4.3.

For performing the video recommendation task, we gather the centroids (that represent each lecturer in the video) from all videos in the dataset. Next, we perform the process described in Section 5.2. For *Centroids Clustering*, we also use Algorithm 1 with the same parameters $t = 5$, $\omega = 0.2$, and

the Aglommerative Clustering as `Clustering` procedure. Finally, based on the clusters generated, we perform the *Ranking* step

To evaluate our ranking, for each video we compute the Average Precision (AP), which evaluates how well a ranking of recommendations is based on each element's relevancy. This metric penalizes more a ranking if a non-relevant element is recommended in the first positions than if it was in the last ones. Let P_k be the precision of the first k elements of a ranking, which is the percentage of videos that are relevant in the sub-ranking that starts at position 1 and ends at position k. Let α_k denote the relevancy of the video in position k, where $\alpha_k = 1$ if the video is relevant, and 0 otherwise. The AP of a given ranking is defined as follows

$$AP = \frac{1}{GTP} \sum_{k=1}^{n} P_k \cdot \alpha_k \tag{5-2}$$

where GTP refers to the total number of ground truth positives in the ranking, which is the total number of videos that are considered relevant in a ranking. Figure 5.2 shows an example of how the AP is computed for a given ranking. In this case, the $GTP = 3$ because the total number of relevant videos in the ranking is 3 (videos A, B and D).

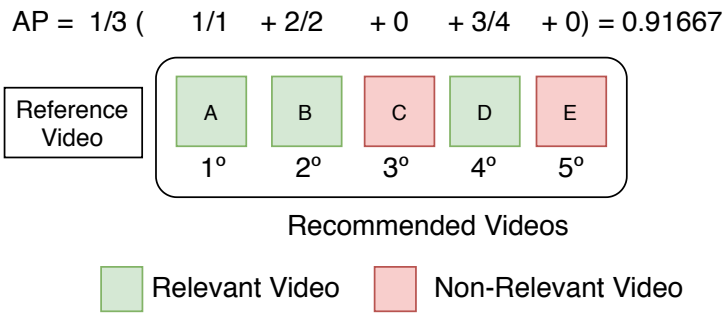


Figure 5.2: Example of how the Average Precision (AP) is computed for a reference video and its recommended videos.

In order to prevent outliers from having much influence in the recommendation (e.g. a person that is not a lecturer – and not relevant to the video – and appears for a short amount of time), we experimented different thresholds of presence intervals in a video for a person to be considered as "present" when computing the score for the ranking. In this way, a $p_{l,v}$ lesser than the threshold is considered as 0. Besides the Average Precision, we also compute the mean and minimum values of the recall (MeanR), precision (MeanP), and F1-Score (MeanF1) for the recommendation generated for each of the videos, without considering the positioning of these videos in the rankings. The recall refers to the percentage of relevant videos that are present in the ranking. The F1-score represents an overall performance metric based on the harmonic mean

of the precision and recall. Table 5.1 shows the thresholds used, the values of recall, precision, F1-score, and the Mean Average Precision (mAP).

One can observe from Table 5.1 that the precision increased with the use of the threshold. Different from the precision, the recall decreased with the increase of the threshold. It means that with a greater threshold more videos that should be recommended were not chosen by our method. It is important to notice that these two metrics (precision and recall) do not consider the ordering of the recommendations. Different from them, the Mean Average Precision (mAP) has high values for all thresholds, especially because the score for computing the ranking takes into consideration the percentage of time that a person appears in the reference and recommended videos. Then, we can conclude that our proposed approach for ordering the recommended videos tends to recommend more suitable videos first with a high mAP≈ 0.99.

Table 5.1: Results obtained with our approach with different thresholds of time presence for a lecturer to be considered as present in a video.

Thershold	**MeanR**	**MeanP**	**MeanF1**	**mAP**
0%	0,88851	0,64681	0,70971	0,98641
1%	0,88851	0,64681	0,70971	0,98641
2%	0,88851	0,64885	0,71171	0,98641
3%	0,88851	0,67368	0,73086	0,98642
4%	0,88851	0,67368	0,73086	0,98642
5%	0,88851	0,69923	0,74930	0,98642
6%	0,88851	0,73615	0,77648	0,98642
7%	0,88742	0,77849	0,80768	0,98642
8%	0,88742	0,78408	0,81111	0,98643
9%	0,88742	0,80171	0,82410	0,98643
10%	0,88742	0,83165	0,84306	0,98643
11%	0,88742	0,85306	0,85693	0,98643
12%	0,88616	0,85956	0,86018	0,98662
13%	0,88490	0,88216	0,87305	0,98688
14%	0,88289	0,90265	0,88450	0,98884
15%	0,88289	0,90265	0,88450	0,98884
16%	0,88163	0,91327	0,88980	0,98908
17%	0,87197	0,91538	0,88580	0,98912
18%	0,87197	0,91538	0,88580	0,98912
19%	0,87086	0,93165	0,89476	0,98946
20%	0,86130	0,93645	0,89218	0,99000
21%	0,86130	0,93645	0,89218	0,99000
22%	0,86130	0,95054	0,89886	0,99000
23%	0,86130	0,95054	0,89886	0,99000
24%	0,85805	0,95718	0,90046	0,99165
25%	0,85805	0,95718	0,90046	0,99165

5.4 Discussion

In this chapter, we present a method for educational video recommendation using *Video Face Clustering*. More precisely, we use an unsupervised clustering-based method and a heuristic for ranking. We extract the video face clusters centroids to perform another clustering step that creates a relationship of videos that share the presence of the same lecturers. Finally, we rank the recommended videos based on the amount of time each lecturer is present. It is worth mentioning that our method is completely automatic and does not require any information from the video files in advance. Moreover, our approach does not need to know or store the identity of the lecturers for performing recommendations, preserving their privacy.

A collateral contribution of this application, that is a result of *Video Face Clustering*, is video segmentation by lecturers. As illustrated in Figure 5.3, we can create a timeline based on lecturers' presence, which can be used to help students in finding moments where specific lecturers are present. With this segmentation, we could recommend specific parts of the video to the student.

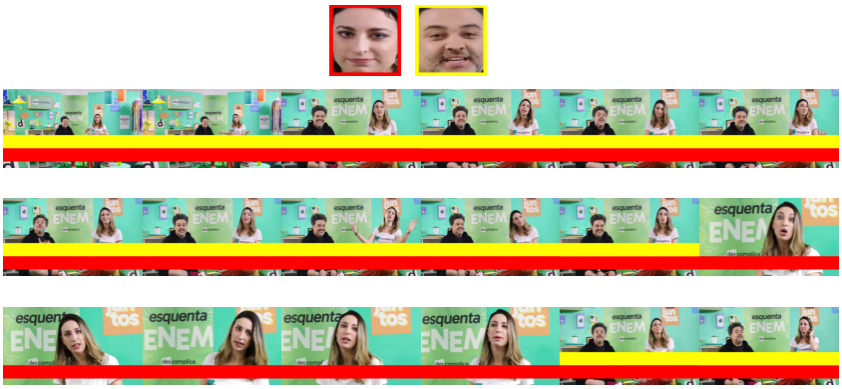


Figure 5.3: Educational video timeline tagged by the lecturers presence. Notice that the frames with the lecturer on the left are tagged in red, while the frames with the lecturer on the right are tagged in yellow.

The main limitation of our method is that we can only recommend videos in which the lecturers are visually present. However, this method can be used in a hybrid recommendation approach, that combines both textual and audiovisual information from the video to create clusters. The results of this application were published at relevant multimedia conference (63).

6
Video Face Clustering for Subtitles Positioning in 360-Videos

In (64), we proposed an authoring model for interactive 360-video. In such a model, we can define interactive 360-videos that are presented together with additional information attached to it, such as images, text, subtitles, 2D traditional videos, and spatial audio. The positioning of such information is defined by their polar coordinates, start time, and duration. For instance, we can define that a text moves with the user's head motion and is always visible or that such text is placed at a fixed position if it is in the user's field of view. In this chapter, we describe how *video face clustering* can be used together with this authoring model for automatic subtitles positioning in 360-video. Section 6.1 describes how we adapt *Video Face Clustering* for the context of 360-videos. In Section 6.2, we describe the authoring model we proposed that together with *Video Face Clustering* can be used for automatic positioning subtitles in 360-videos.

6.1
Video Face Clustering in 360-Videos

Nowadays, the most common way for representing and transmitting 360-video is using equirectangular projection (65). With the equirectangular projection, each sphere point is defined by two angles (66): *latitude* $\theta \in [-90°, +90°]$ and *longitude* $\phi \in [-180°, +180°]$. This kind of projection creates challenges for image processing and computer vision algorithms, especially to the convolution-based ones because of the severe distortions in areas vertically distant from the center of the image (see Figure 6.1). Due to these distortions, we adapted the *Face Detection* step of our method which uses a traditional CNN. Section 6.1.1 describes how we adapted the *Face Detection* step to the 360-video context.

6.1.1
Face Detection in Equirectangular 360°

In order to mitigate the problem of severe distortions present in the equirectangular projection, we have used an approach based on viewports

(a) Outdoor equirectangular image. (b) Indoor equirectangular image.

Figure 6.1: Examples of 360-images represented through equirectangular projection.

extraction. Figure 6.2 shows this process and each of its steps is explained in the following paragraphs.

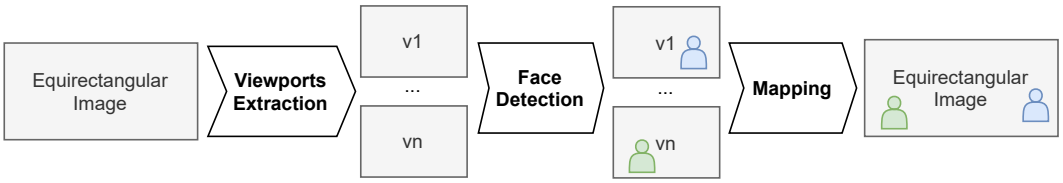


Figure 6.2: 360-degree face detection process.

Given an equirectangular image, which could also be a frame from a 360-video, we first perform *Viewports Extraction*. A viewport represents a portion of the 360-degree scene (see Figure 6.3), it is similar to when a person takes a picture in the real world. The picture represents the world from a point of view towards a specific direction. A viewport is defined by its center, in polar coordinates (lat, long), its field of view (FoV), and the width or height of the resulting image.

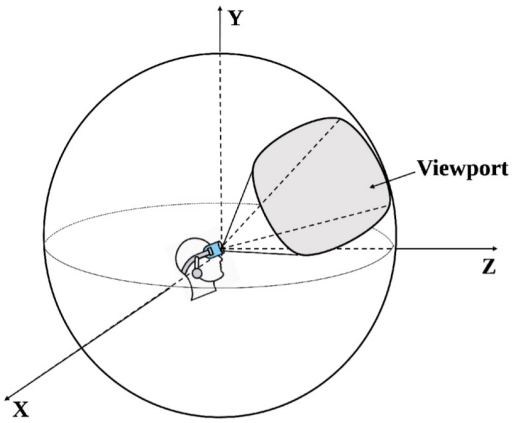


Figure 6.3: Example of a viewport from a 360-video. Extracted from Nguyen *et al.* (3).

Because we want the viewports to cover the greatest possible area from the 360-degree image, our method receives two parameters: the viewports'

density (d) and *FoV*. The *density* is equal to the number of viewports in the latitude range. Proportionally, the number of viewports in the longitude range is equal to double the *density*. Figure 6.4 shows viewports extracted from Figure 6.1a with *density*=3 and FoV=60°. As the density was equal to three, the number of viewports is three in the latitude range and six in the longitude range. In total, 18 viewports were extracted in the example below.

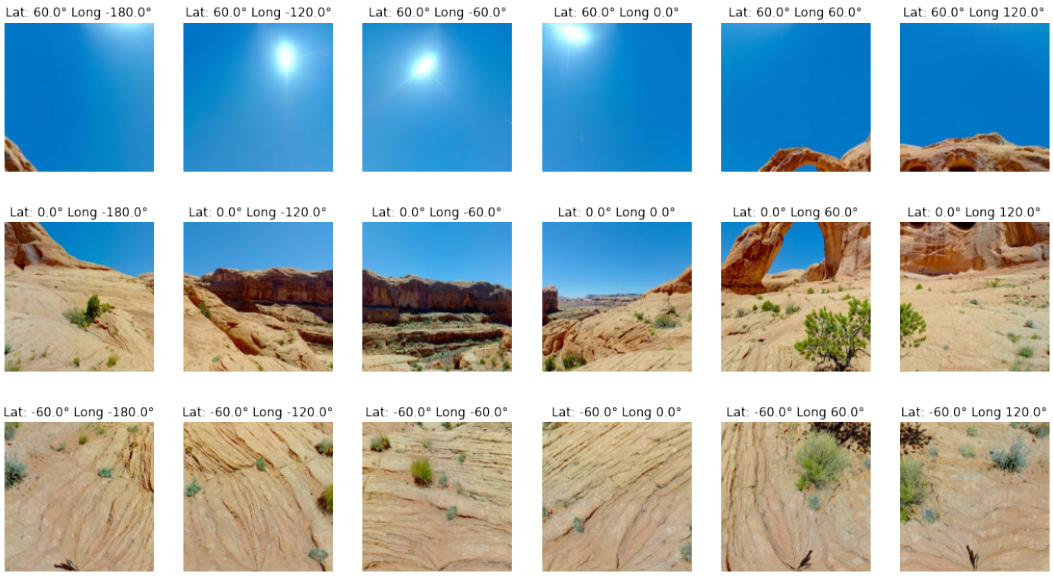


Figure 6.4: Viewports extracted from Figure 6.1a with *density*=3 and FoV=60°.

With these viewports, we aim at reducing the distortions since we are representing the equirectangular image with a series of standard images, similar to the ones used by traditional CNNs. Next, for each of these viewports, we perform *Face Detection* using a traditional CNN. Then, for each viewport we have a group of faces detected.

In the last step of our approach, called *Mapping*, we map each face detected back to the equirectangular image. One can notice that some parts of the equirectangular image are present in more than one viewport. Notice that the sun is present in three viewports in the first line of Figure 6.4. To avoid repeated detections in the equirectangular image, we use the Non-Maximum Supression (NMS) algorithm. This algorithm eliminates overlapping bounding boxes within a given threshold and is widely used in object detection algorithms (67, 68, 69, 70, 71).

We evaluated this approach in comparison to applying a detection model directly to the equirectangular image. We searched for datasets for face detection in equirectangular 360-images. However, by the time of our search, we have not found any. For that reason, we decided to create a synthetic dataset to perform this evaluation. Similar to Fu *et al.* (72), we created a synthetic

dataset based on the Face Detection Data set and Benchmark (FDDB) (73). Fu *et al.* (72) create a synthetic dataset for face detection in 360-degree fisheye images. Ours, differently, aims at equirectangular images, projecting standard images to the equirectangular projection.

The FDDB dataset (73) is a popular benchmark for face detection evaluation containing 2845 images and 5171 faces. We collected 19 indoor and outdoor equirectangular images from Google Images,[1] ESO,[2] and PxHere[3] to use as background. For each image in the FDDB dataset, we randomly chose a latitude, longitude, and equirectangular background image to project it following the projection to equirectangular described in the work of Su *et al.* (74). For each face present on the FDDB image, we projected its bounding box to the new equirectangular synthetic image. Figure 6.5 shows an example of an image from FDDB projected in the polar coordinates $lat = -60°$ $long = 0°$ to an equirectangular image.

(a) FDDB image example. (b) Projection example of FDDB image.

Figure 6.5: FDDB image projection to Equirectangular image.

Depending on the position that the traditional FDDB image is projected, the distortions are more or less severe. The distortions increase as the projection vertically distances from the center of the equirectangular image. Figure 6.6 shows different resulting equirectangular images depending on the projection position. Notice that the projections vertically close to the border of the image have more distortions than the ones close to the center of the image.

We evaluated our approach in the synthetic dataset using MTCNN(48) as the model for face detection. We compare the usage of MTCNN alone, directly in the equirectangular images, against MTCNN with viewports. For the MTCNN with viewports, we used extracted viewports with *density*=3 and FoV=60°. We evaluated both methods using the Mean Average Precision (mAP) that is widely used for the object detection task (75, 76, 77, 78).

[1]https://www.google.com/imghp
[2]https://www.eso.org/public
[3]https://pxhere.com

Figure 6.6: Different resulting synthetic equirectangular images depending on the position that the rectlinear image was projected.

From Figure 6.6, one can notice that a greater module of the latitude ($|lat|$) implies more severe distortions, as the projection is more vertically distant from the center of the equirectangular image. Figure 6.7 shows the results we obtained for MTCNN and MTCNN with viewports. We vary the x-axis of the chart so that we consider only the synthetic equirectangular images whose projection's $|lat|$ is greater than or equal to each x. In this way, a greater x means a more difficult subset of images from our synthetic dataset. The mAP value for MTCNN drops drastically as we start considering only more and more difficult images. In contrast, with the use of viewports, the mAP value still decreases but at a slower pace. Besides achieving better results, one of the main advantages of this technique is that it can be used with a model that was trained for detecting faces in traditional images.

With this adaptation, we can use *Video Face Clustering* for 360-video. To do that, in the *face detection* step we use the viewports strategy with MTCNN.

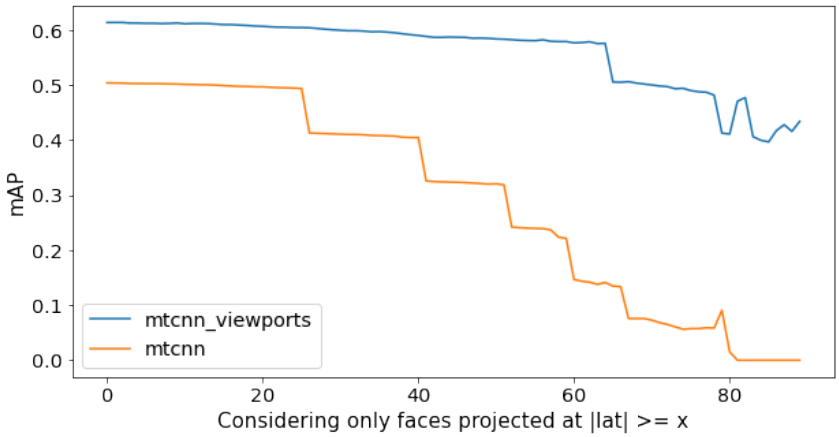


Figure 6.7: Face detection results using MTCNN and MTCNN with viewports.

6.2 An Authoring Model for Interactive 360-Videos

In this section, we describe an authoring model for interactive 360-videos. Although the model is not entirely used for subtitles positioning in 360-videos, we describe it completely because the development of such an authoring model was part of the researches conducted during our master research.

Despite the limited "look around" interactivity supported by traditional 360-videos and HMDs, *interactive 360-videos* (79, 80) support additional interactive elements, such as overlaid 2D/3D information, hyperlinks, and additional input elements. Such features allow an enhanced user experience, supporting applications such as remote operations and telepresence, museums, immersive interactive narratives, and improve educational content. Most of the current 360-video streaming services, however, still do not support such interactive features for 360-videos. Indeed, when compared to its 2D-only video counterpart, although most of the popular streaming services (e.g., Youtube and Facebook) offer tools to create interactive 2D videos (*e.g.*, to insert additional information, subtitles, and hyperlinks), they lack similar tools to create interactive 360-videos.

We propose a declarative authoring model that allows authors to design and create interactive 360-videos. First, we analyze different scenarios of immersive multimedia applications based on 360-videos intending to extract the main requirements for such an authoring model. Then, based on the gathered requirements, we propose an XML-based declarative model that allows authors to design and create 360 interactive videos.

6.2.1 Scenarios and requirements

The following target scenarios are used to gather the requirements for our authoring model for interactive 360-videos.

360 hypervideo. This scenario is characterized by navigation among 360-videos, and can be useful, for instance, in entertainment or educational context (see Figure 6.8). The user can navigate from the current to another 360-video, while the current one proceeds in a "preview mode". Besides the support for navigation between 360-degree videos, it should be possible to add overlaid information to it (e.g., image, video, text) to enhance its viewing experience. An example of such a scenario is a virtual tour in a museum, where the user may navigate through different rooms. Each room could have additional information about the presented piece of art. Another example is a virtual learning environment, in which the user can navigate among different topics and learn more about each of them with additional information.

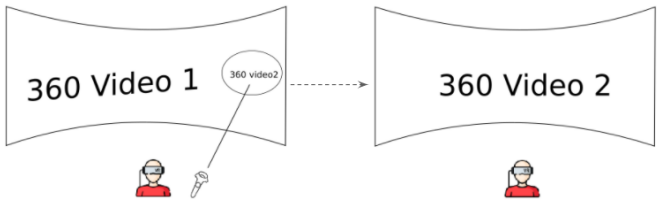


Figure 6.8: 360 hypervideo scenario.

Accessible 360-video. In this scenario, a 360-video is presented together with its translation in either sign language or subtitles (see Figure 6.9). For the sign language case, both a 2D regular video using Picture-in-Picture (PiP) or a 3D human model could be displayed. Such a scenario intends to provide an immersive experience for people with hearing disabilities. An example is a voice-based virtual tutorial for managing a machine in a factory that can support sign languages or subtitles to allow the inclusion of people with hearing disabilities.

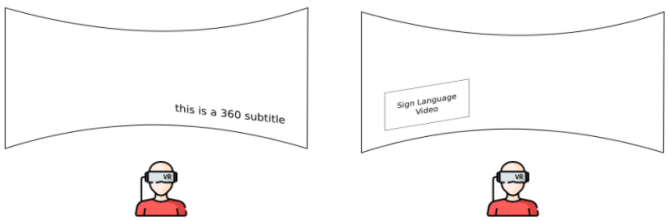


Figure 6.9: Acessible 360-video scenario.

360-video with guided attention. In this scenario, a 360-video has a recommended region to look at. Whenever the user is not looking at the recommended region, additional guiders (e.g., arrows) can be rendered informing the user to where he should be looking in the 360-video (see Figure 6.10). Also, a live view of it can be presented in picture-in-picture (PiP) in part of the field of view; thus, providing the user with key content, whether he/she is looking at it or not. Besides visual cues, spatial audio cues are also important in virtual environments and can help guide users' attention. Thus, the audio of the 360-degree video may come from the recommended position, in a way that it sounds different depending on the position the user is looking at. To support such a scenario, it should be possible to detect if the user is looking in a specific direction of the scene and to render another segment of the video using PiP. The region of interest of a 360-degree video is content-dependent, thus the producer should be able to customize it towards providing the best possible user experience. An example of such a scenario is a lecture, in which the recommended position to look at is where the lecturer may be located, e.g. the stage.

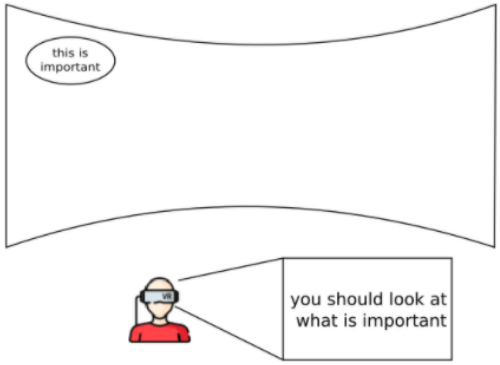


Figure 6.10: 360-video with guided attention scenario.

Based on the above target scenarios, we can extract requirements addressing presentation (RP) and interaction (RI) that should be supported by an authoring model for 360 interactive videos, which are detailed in what follows.

- **RP.1 Overlaid information**. Additional media objects (video, image, sound, and text) might be positioned in the 360-degree environment.
- **RP.2 Styling**. Different media elements may share the same position and presentation characteristics. Thus, the author should be able to reuse the same style in different elements.

- **RP.3 3D Audio**. A 3D audio manipulates the sound delivered by stereo speakers or headphones, creating the idea that the audio is coming from a specific direction.
- **RP.4 Subtitles**. Support subtitles positioning in 360-video.
- **RP.5 360 Spatial Layout**. The media elements should use a 360-based coordinate system, whereas the viewer camera is the center.
- **RP.6 Presentation Timing**. The media elements should be presented synchronized with the 360-video of the scene.
- **RI.1 Navigation**. This requirement intent to enable users to navigate between 360-videos. This also includes showing them a preview of possibles videos to visit.
- **RI.2 Hotspot**. This requirement defines a position in the 360-degree scene in which different actions can be performed when the user looks at it or is not looking at it.
- **RI.3 Viewport preview**. This requirement refers to the use of a picture-in-picture live view at the user's current viewport, rendering an important section of the 360-degree scene defined by a *hotspot*.

6.2.2 Proposed model

To fulfill the aforementioned requirements, we propose an authoring model that allows authors to design and create 360 interactive videos. Table 6.1 presents the entities of the model.

<presentation360> is the root element of our model. It has two children elements: *<head>* and *<body>*. Similar to other declarative multimedia languages (*e.g.*, HTML, SMIL (81), and NCL (82)), the reusable elements inside the *<head>* and the structured content inside the *<body>*.

In the *<head>*, reusable objects can be grouped by the <style> element, which can be seen as a macro in which any of the media attributes may be defined (**RP.2**). Then, any element may reuse these attributes assigning their *style* attribute to the *id* of a previously defined *<style>*.

The *<body>* element is composed of different <scene360> elements. Its *entry* attribute indicates the identifier of *<scene360>* that starts with the application. The *<scene360>* element is defined by its *id*, *src* (source of the 360-degree video) and *volume* (varying from 0 to 1). Each scene is composed of a set of media objects and the temporal behavior of those objects (**RP.1**). In its current version, we support text, image, video, audio, and subtitle

element	children	attributes
presentation360	*head*, body	
head	*style*	
style		*
body	*scene360+*	entry
scene360	*text**, *image**, *video**, *subtitle**, *preview**, *hotspot**, *mirror**	src, volume?
text*, *image		id, src, r?, phi?, theta?, tpos?, style?, begin?, dur?, followCamera?, onselect?
audio*, *video*, *subtitle*, *preview		id, src, r?, phi?, theta?, tpos?, style?, begin?, dur?, followCamera?, onselect?, clipBegin?, clipEnd?
mirror		id, src, r?, phi?, theta?, tpos?, style?, begin?, dur?, followCamera?
hotspot		id, src, r?, phi?, theta?, tpos?, style?, begin?, dur?, onLookAt?, duringNotLookingAt?, onSelect?

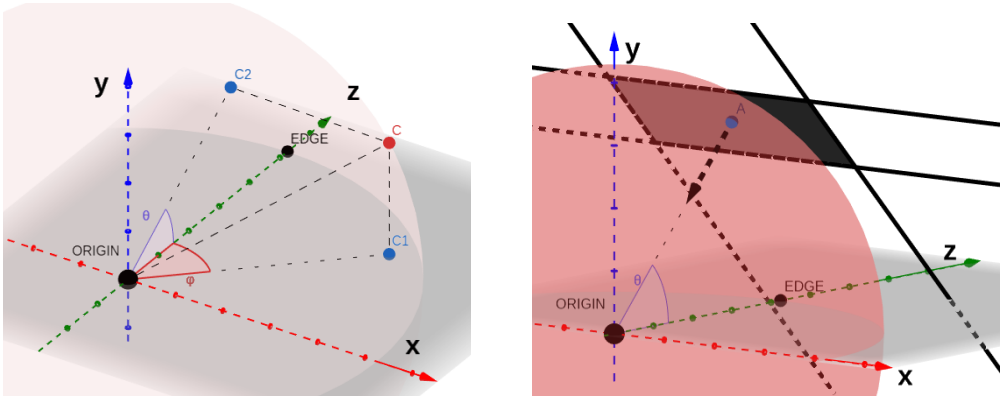
Table 6.1: Elements BNF. Conventionally: a *plus sign* (+) indicates that the element can occur one or more times. The *asterisk* (*) indicates that the element occurs zero or more times. Optional attributes (may not exist or have an occurrence) appear with a *question mark* (?) unlike the others that are mandatory.

media objects, each one with its respective XML element (see Table 6.1). The following attributes are shared by all media elements:

- *id*: the identifier by which the element is referenced by other elements.
- *r, phi, theta*: define the element initial 360 spatial position.
- *tpos*: it determines how the element moves through the 360-video's duration by pointing to a file that specifies 360 positions for predetermined moments in time. Such positions are relative to the initial *r, phi, theta.*
- *begin*: the time, in seconds, in which the element is started relatively to the 360-video it is a child of.
- *dur*: the duration, in seconds, of the element.
- *clipBegin, clipEnd*: define that will be presented only a portion from sourced media.
- *followCamera*: a Boolean attribute that, if *true*, makes the element moves with the camera. Such movement creates the impression that the element is fixed for the user.
- *onselect*: it refers to the *id* of a 360-degree interactive video in a way that when the element is selected (using controllers of an HMD), the user is transported to the referred 360-degree interactive video.

Besides the above attributes, individual media objects can also have additional attributes. For instance, *<image>*, *<video>* and *<audio>* have the attribute *src*, that refers to the file source of the media object; *<audio>* and *<video>* have the attribute *volume*, that defines the volume of such media objects, varying from 0 to 1. *<audio>* also has a 3D behavior (**RP.3**), in a way that the user perceives the position in which it is placed. 3D Audio, also referred as spatial audio, is considered itself a type of immersive content (2). *<subtitle>*, *<preview>* and *<mirror>* are non-traditional objects. The *<subtitle>* element has also a *src* attribute that refers to an SRT (SubRip Subtitle Format) file (**RP.4**). *<preview>* and *<mirror>* refer to other elements from the scene and have specific behaviors.

In our 360 spatial model (**RP.5**), the media objects are positioned using a polar coordinate system. Figure 6.11a shows the definition of the polar coordinates of a point C, which represents the center of a media object. $C1$ is the projection of C on the xz plane, while $C2$ is the projection on the yz plane. Each r attribute value specifies the radius of an imaginary sphere where the media objects are positioned. All of those spheres are concentric. The angles are defined based on the segment that goes from the origin to the edge. The attribute *phi* (ϕ) specifies the horizontal angle (or longitude, in degrees) from that segment to the one that goes from $C1$ to the origin. Similarly, the attribute *theta* (θ) defines the vertical angle (or latitude), but using $C2$. The camera is positioned at the center of that coordinate system (0, 0, 0) and has a field of view of 60 degrees. The 360-video is rendered as a background so that it is always behind anything else in the scene.



(a) Polar coordinates example. (b) Element rotation example.

Figure 6.11: Polar coordinates system

The rotation of a media object is set in a way that the normal vector on the center of its surface points to the origin. By doing so, in any position it is placed, the media object is always facing the viewer. Figure 6.11b shows this

rotation, where A is the center of the media object.

In the proposed model, each *<scene360>* has always a main 360-degree interactive video. The initial scene in which the application starts is defined by an attribute *entry* in the element <body>. This attribute refers to the *id* attribute of the *<scene360>* element pointing to the main 360-degree video. Media objects inside a 360 scene can be synchronized to the main video (**RP.6**). For this, we use SMIL *begin* and *dur* attributes in a media object. These, respectively, define the start time and duration for the presentation of the specified media object.

The *onselect* attribute is used to support user navigation through 360 scenes (**RI.1**). This attribute can be defined in any media element, by referencing the *id* of the target scene in its value. Once defined in an element, it transforms it into a navigation element for the referenced scene. Moreover, the *<preview>* defines a viewport of a *<scene360>* using its *id* in the *src*, which can be viewed as a 2D video inside another scene. The temporal segment of the target scene displayed in the preview is defined by the attributes *clipBegin* and *clipEnd*.

The 360-video might have a particular region of interest to the viewer, which is represented by the *<hotspot>* (**RI.2**) element. Meixner (83) defines a hotspot as an interactive area in a video that invokes an action. The author may create interactions when the user is either looking at it or not. As in the other elements, its position is defined by the *r, phi, theta* attributes. This element has also the attributes *onLookAt* and *duringNotLookingAt*. The attribute *onLookAt* specifies another element to be started once the user looks at the *<hotspot>*. Similarly, the attribute *duringNotLookingAt* specifies another element to be started when the user is not looking at the *<hotspot>* and is stopped once the user looks at it.

The *<mirror>* element refers to picture-in-picture live view of a viewport of the *<scene360>* (**RI.3**). This viewport is defined using the *src* attribute and assigning it to the *id* of a *<hotspot>* element. Using these elements, authors can define that when the user is staring at a viewport different from the one defined as a *<hotspot>*, the *<mirror>* element will be triggered showing the PiP view at the current user's viewport, attracting his attention to the important section of the 360-degree video. Since the *<mirror>* element refers to a viewport of the current scene, it does not support *onselect* because it would not be intuitive to have an element referring to a scene and navigating to another when selected.

6.3 Dynamic Subtitles Positioning in 360-video

Using the adaptations described in Section 6.1, we are able to apply *video face clustering* in 360-videos. Figure 6.12 shows an example of this usage. It contains the two actors present in the 360-video and part of the video's timeline tagged with colors representing the presence of these actors.

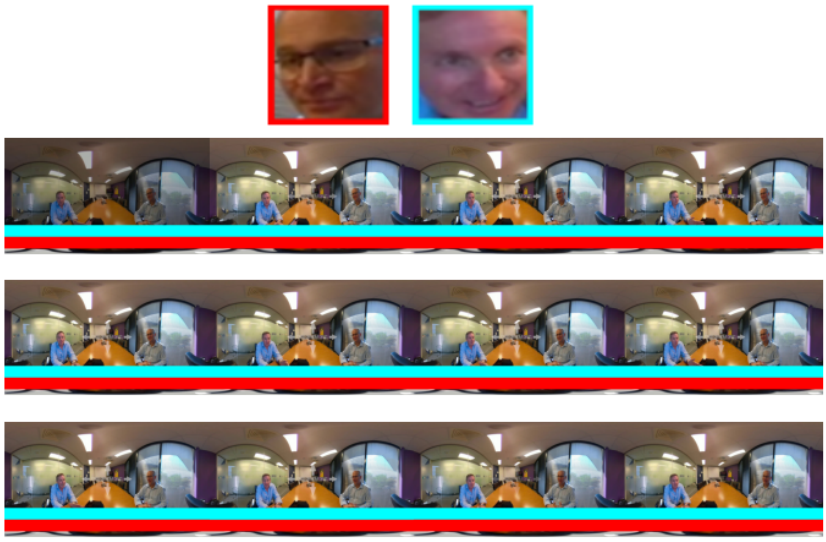


Figure 6.12: 360-video timeline tagged by the actors presence. In this case, both actors were present in all frames.

With this clustering, for each actor detected, we generate a file containing the positions in polar coordinates of him/her for each time. By doing so, we use these files for defining the subtitles positioning using our authoring model. Listing 6.1 shows the XML representation of an interactive 360-video with subtitles. In lines 6 and 7, we define subtitles elements, one for each actor. Using the *tpos* attribute, we define that these subtitles follow the actors' positions in the video. As the positions defined in the *tpos* files are relative to the initial position of the subtitle element, we use *theta*=5, so that the subtitles are placed 5 degrees below the center of the actors' faces.

```
1  <presentation360>
2    <head>
3    </head>
4    <body entry = "boardroom">
5      <scene360 id="boardroom" src="boardroom.mp4" volume="1">
6        <subtitle src="actor1.srt" r = "10" theta="5" tpos="actor1-positions.csv"/>
7        <subtitle src="actor2.srt" r = "10" theta="5" tpos="actor2-positions.csv"/>
8      </scene360>
9    </body>
10 </presentation360>
```

Listing 6.1: XML representation for 360-video with subtitles following the actors positions.

We can also define a more interactive setting for subtitles in 360-video with our authoring model and *Video Face Clustering*. In this new setting, the subtitles follow the actors when they are in the user's viewport, which means that the user can see them. When they are not visible, subtitles are presented at the bottom center of the user's viewport. Listing 6.2 contains the definition of this setting.

In the *head*, we define three *styles*. These *styles* contain attributes that can be reused by the elements in the body. The first two *styles* (lines 3 and 4) define attributes regarding the actor positions in the 360-video with their respective times. These styles are used by the *subtitle* elements on lines 9 and 10 for defining that such subtitles follow the actors' positions. We also define two *hotspot* elements that follow the actor positions (lines 11 and 12). They are used like triggers with the attribute *duringNotLookingAt* to start the other *subtitles* when the *hotspot* elements (the actors) are not visible to the user. The other *substile* elements reuse the style defined on line 5. This *style* contains the attribute *followCamera* set to *true*, so that it follows the users head motion. In this way, the *subtitles* defined on lines 13 and 14 follow the user's head motion.

```
1  <presentation360>
2    <head>
3      <style id ="sty-actor1" tpos="actor1-positions.csv" begin="0s" r = "10"/>
4      <style id ="sty-actor2" tpos="actor2-positions.csv" begin="0s" r = "10"/>
5      <style id ="sty-follow" begin ="0s" theta="20" followCamera="true" r="10"/>
6    </head>
7    <body entry = "boardroom">
8      <scene360 id="boardroom" src="boardroom.mp4" volume="1">
9        <subtitle style="sty-actor1" src="actor1.srt" theta="5"/>
10       <subtitle style="sty-actor2" src="actor2.srt" theta="5"/>
11       <hotspot style ="sty-actor1" duringNotLookingAt="sub-follow-actor1"/>
12       <hotspot style ="sty-actor2" duringNotLookingAt="sub-follow-actor2"/>
13       <subtitle style ="sty-follow" id="sub-follow-actor1" src="actor1.srt"/>
14       <subtitle style ="sty-follow" id="sub-follow-actor2" src="actor2.srt"/>
15     </scene360>
16   </body>
17 </presentation360>
```

Listing 6.2: XML representation for 360-video with subtitles following the actors positions.

Figure 6.13 shows how this setting works. In Figure 6.13a, the subtitles are placed close to the actor that is visible in the current user's viewport. When the actor is not visible, as shown in Figure 6.13b, the subtitles are placed at the bottom of the user's viewport. In this way, the user does not lose the content present in the subtitles.

It is worth noticing that, with this setting, the disadvantage presented in Table 2.1 that the speaker-following subtitles are not always visible to the user is mitigated.

(a) Subtitles close to actors' faces when they are in the user's viewport.

(b) Subtitles positioning in the bottom of user's viewport otherwise.

Figure 6.13: Dynamic subtitles positioning using video face clustering and our authoring model.

6.4 Discussion

This chapter proposed an automatic way for positioning subtitles in 360-video. This is done with the dynamic placement of subtitles based on the automatic localization of actors. For doing that we use an authoring model for composing and presenting these dynamic subtitles in 360-videos.

Different from traditional 2d-videos, 360-videos are commonly represented using the equirectangular projection. This projection introduces severe distortions that are not easily handled by traditional CNNs. Then, we used an approach based on viewports extraction for the face detection step of *Video Face Clustering*. We first extract viewports from the equirectangular image (or 360-video frame, in this case) and use MTCNN pre-trained with standard images for face detection in each viewport. Finally, these detections are mapped back to the equirectangular image. We evaluated this approach against MTCNN without viewports in a synthetic equirectangular dataset based on the FDDB benchmark. The usage of viewports achieved better results, especially when considering images with more severe distortions.

In this chapter, we also proposed an authoring model that allows the design and creation of interactive 360-videos. The proposed model is the result of the analysis of different scenarios of immersive 360 multimedia applications. Besides supporting dynamic subtitles positioning in 360-videos, it also supports navigation among 360-videos, additional media, interactive behaviors such as selection and guided attention. For other case studies not closely related to this dissertation, please check our paper that presents this authoring model (64). This model is supported by a player based on the Unity Game Engine.[4]

[4]https://github.com/TeleMidia/VR360Authoring

7
Conclusions

In this dissertation, we present a method for spatio-temporal localization of actors based on *Video Face Clustering*. As part of this method, we also define an algorithm for finding an adequate number of clusters based on the silhouette score. We investigate to what extent this method can be used as the core to leverage and enhance some innovative applications, especially in three different practical and important tasks: *Video Face Recognition*, *Educational Video Recommendation*, and *Subtitles Positioning in 360-Video*.

For the *Video Face Recognition*, we propose a cluster-matching-based approach, derived mainly by the characteristics of our core *Video Face Clustering* method, which is very scalable since the effort spent with annotations is significantly reduced — as it is done over clusters instead of single images. This method uses *Video Face Clustering* and a heuristic for cluster matching to recognize people in videos. It has achieved a recall of 99.435% and precision of 99.131% when considering faces extracted from a set of 13 video files. As another consequence of face clustering, our technique can be useful for creating and labeling datasets in a less time-consuming way by labeling clusters instead of individual images. One of the limitations of this application is related to the size of the video set used for the overall evaluation of our method. This is due to the difficulty of finding videos where it is possible to manually identify in each frame whether each person present is registered or not in our labeled clusters.

For the *Educational Video Recommendation* task, we investigate a new feature that can be used for such recommendation: the presence of specific lecturers, which again is a direct result of the application of our core method. After performing *Video Face Clustering* on each video, we extract their centroids to perform another clustering step that creates a relationship of videos that share the presence of the same lecturers. Finally, we rank the recommended videos based on the amount of time that each lecturer is present. Our method uses only the video files for performing recommendations, no other information about these videos nor the identity of the lecturers is necessary. It is worth mentioning that we do not intend to substitute other video recommendation methods, rather our application shows that, if the presence

of lecturers is a relevant feature for educational video recommendation, it can be used for this purpose with a mAP value of 0.99. The main limitation of this application is that we can only recommend videos in which the lecturers are visually present. As future work, we intend to investigate a hybrid recommendation approach, that combines both textual and audiovisual information from the video to create clusters.

For the *Subtitles Positioning in 360-Video* task, our main contribution is the proposal of dynamic placement of subtitles based on the automatic localization of actors. To achieve this goal, we adapted our spatio-temporal localization method to the 360-video setting and created an authoring model for interactive 360-videos. Because of the severe distortions present in equirectangular 360-videos, we used an approach based on viewports extraction for the face detection step of *Video Face Clustering*. To evaluate this approach against the sole use of a traditional CNN, we created a synthetic dataset by projecting images from the FDDB benchmark to equirectangular backgrounds. Our approach, which consists of using viewports with a traditional CNN, performed better than using the traditional CNN directly to the equirectangular images. Moreover, we proposed an authoring model that allows authors to design and create interactive 360 videos. The proposed model is the result of the analysis of different scenarios of immersive 360 multimedia applications. Besides supporting subtitles positioning in 360-videos, it also supports navigation among 360-videos, additional media, etc.

In summary, we highlight the following contributions of this dissertation:

1. A method for video face recognition that is easily scalable and helps in labeling images dataset;

2. A method for educational video recommendation based on the presence of lecturers;

3. A method for face detection in equirectangular 360-images that uses models pre-trained with traditional images;

4. A synthetic dataset for face detection in equirectangular 360-images;

5. An authoring model for interactive 360-videos;

6. A player for interactive 360-videos;

7. An approach for automatic positioning subtitles in 360-videos based on the actors' positions;

The choice for our three proposed applications was due to observations of opportunities to apply *Video Face Clustering* in different contexts. We started with *Video Face Recognition* task. We observed that the use of only the *Video Face Clustering* part of the method could generate the time segments that each actor is present without having to identify them. Then, we hypothesized that this particular feature could also be used to clustering actors in different videos. From this observation, we decided to investigate a method for *Educational Video Recommendation* using this clustering of actors in different videos. The idea for *Subtitles Positioning in 360-video* came from the authoring model we developed. We observed that we could automatically identify the actors using *Video Face Clustering* so that the subtitles could follow the speakers in the 360-video. Thus, we adapted *Video Face Clustering* for the 360-video setting. It is worth noticing that with this adaptation, the other two applications (*Video Face Recognition* and *Educational Video Recommendation*) could also be applied to the context of 360-videos.

This work has opened lots of branches for future research. For the particular case of subtitles positioning in 360-video, there is still much to explore. It is not yet clear what is the most adequate way to place subtitles relative to the actors' positions. For instance: is it better to place them above or below the actors' faces? Should the subtitles follow the actors immediately or it would be better to have a delay with a smooth transition? A path towards answering these questions is by experimenting and evaluating these and other possible settings with users.

Other possible future work comes from the combination of the applications we explored in this dissertation. For instance, the identification of lecturers using a lecturer's image dataset with *Video Face Recognition* could improve the task of *Educational Video Recommendation* by inserting additional relationships among lecturers (e.g. lecturers that teach the same subject). Another possibility is to use *Video Face Recognition* to identify and insert the name of the actor present in 360-videos together with the subtitles so that the users know who is talking when the actor is not visible to them.

Although this dissertation has focused on faces, the methods and techniques here investigated could also be applied in other contexts. For instance, we could recommend videos based on the presence of the same *actions* in different videos, so that a reference video with many actions related to culinary, for example, such as pouring and chopping should have videos with similar *actions* as recommended. For doing that, we should be able to represent each action identified as an embedding (vector). Then, the remainder of our method for *Educational Video Recommendation* could be used. In the context of 360-

videos, if we could generate art embeddings from pieces of art, we could apply the process of *Video Face Clustering* (or *Video Art Clustering* in this case) to detect and determine the video segments that each piece of art is present. In this case, instead of using *CNNs* trained for detecting and generating embeddings for faces, we would use them for pieces of art. By doing that, we would be able to use our authoring model to attach additional media such as text and audio to pieces of art in an interactive 360-video. This application could be useful in the context of virtual tours in museums and historical sites.

7.1 Publications

As a result of this research, three papers have already been published at relevant multimedia conferences (62, 63, 64). In (62), we have evaluated video face clustering together with a cluster-matching method for video face recognition. In (63), we have used video face clustering and the presence of actors in different videos as a means for recommending educational videos. In (64), we have developed an authoring model and a player for interactive 360-video.

Bibliography

[1] A. Brown, J. Turner, J. Patterson, A. Schmitz, M. Armstrong, and M. Glancy, "Subtitles in 360-degree Video," in *Adjunct Publication of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video*, ser. TVX '17 Adjunct. New York, NY, USA: Association for Computing Machinery, Jun. 2017, pp. 3–8. [Online]. Available: https://doi.org/10.1145/3084289.3089915

[2] C. Hughes, M. Montagud Climent, and P. tho Pesch, "Disruptive Approaches for Subtitling in Immersive Environments," in *Proceedings of the 2019 ACM International Conference on Interactive Experiences for TV and Online Video*, ser. TVX '19. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 216–229. [Online]. Available: https://doi.org/10.1145/3317697.3325123

[3] D. V. Nguyen, H. T. Tran, and T. C. Thang, "An evaluation of tile selection methods for viewport-adaptive streaming of 360-degree video," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 1, pp. 1–24, 2020.

[4] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, "Deep face recognition: A survey," in *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. IEEE, 2018, pp. 471–478.

[5] L. L. Dias, J. S. Barbosa, E. Barrére, and J. F. de Souza, "An approach to identify similarity among educational resources using external knowledge bases," *Brazilian Journal of Computers in Education*, vol. 25, no. 02, p. 18, 2017.

[6] R. Mahajan, J. Sodhi, and V. Mahajan, "Optimising web usage mining for building adaptive e-learning site: a case study," *International Journal of Innovation and Learning*, vol. 18, no. 4, pp. 471–486, 2015.

[7] M. Omisore and O. Samuel, "Personalized recommender system for digital libraries," *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, vol. 9, no. 1, pp. 18–32, 2014.

[8] E. Barrére, J. F. de Souza, M. A. Vitor, and M. A. de Almeida, "Utilização de enriquecimento semântico para a recomendação automática de videoaulas no moodle," *Revista Brasileira de Informática na Educação*, vol. 28, p. 319, 2020.

[9] A. Hayati and F. Mohmedi, "The effect of films with and without subtitles on listening comprehension of efl learners," *British Journal of Educational Technology*, vol. 42, no. 1, pp. 181–192, 2011.

[10] S. Rothe, K. Tran, and H. Hussmann, "Dynamic Subtitles in Cinematic Virtual Reality," in *Proceedings of the 2018 ACM International Conference on Interactive Experiences for TV and Online Video*, ser. TVX '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 209–214. [Online]. Available: https://doi.org/10.1145/3210825.3213556

[11] B. Agulló, M. Montagud, and I. Fraile, "Making interaction with virtual reality accessible: rendering and guiding methods for subtitles," *AI EDAM*, vol. 33, no. 4, pp. 416–428, 2019.

[12] C. Küblbeck and A. Ernst, "Face detection and tracking in video sequences using the modifiedcensus transformation," *Image and Vision Computing*, vol. 24, no. 6, pp. 564–572, 2006.

[13] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[14] M. Tapaswi, M. T. Law, and S. Fidler, "Video face clustering with unknown number of clusters," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5027–5036.

[15] V. Sharma, M. Tapaswi, M. S. Sarfraz, and R. Stiefelhagen, "Self-supervised learning of face representations for video face clustering," in *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. IEEE, 2019, pp. 1–8.

[16] Y. Adini, Y. Moses, and S. Ullman, "Face recognition: the problem of compensating for changes in illumination direction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 721–732, 1997.

[17] H. S. Dadi and G. M. Pillutla, "Improved face recognition rate using hog features and svm classifier," *IOSR Journal of Electronics and Communication Engineering*, vol. 11, no. 04, pp. 34–44, 2016.

[18] V. Ghosal, P. Tikmani, and P. Gupta, "Face classification using gabor wavelets and random forest," in *2009 Canadian Conference on Computer and Robot Vision*, 2009, pp. 68–73.

[19] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[20] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1891–1898.

[21] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[22] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 67–74.

[23] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro *et al.*, "Applied machine learning at facebook: A datacenter infrastructure perspective," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2018, pp. 620–629.

[24] P. Grother, P. Grother, M. Ngan, K. Hanaoka, C. Boehnen, and L. Ericson, *The 2017 IARPA Face Recognition Prize Challenge (FRPC)*. US Department of Commerce, National Institute of Standards and Technology, 2017.

[25] R. Pena, F. Ferreira, F. Caroli, L. Schirmer, and H. Lopes, "Globo face stream: A system for video meta-data generation in an entertainment industry setting," *23 International Conference on Enterprise Information Systems (ICEIS)*, 2020.

[26] J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua, "Neural aggregation network for video face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4362–4371.

[27] Y. Rao, J. Lu, and J. Zhou, "Attention-aware deep reinforcement learning for video face recognition," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3931–3940.

[28] K. Sohn, S. Liu, G. Zhong, X. Yu, M.-H. Yang, and M. Chandraker, "Unsupervised domain adaptation for face recognition in unlabeled videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3210–3218.

[29] F. Ferreira, D. R. Souza, I. Moura, M. Barbieri, and H. CV Lopes, "Investigating multimodal features for video recommendations at globoplay," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 571–572.

[30] S. Suvash, S. Scott, B. Darius, X. Lexing, and C. Jordan, "Social collaborative filtering for cold-start recommendations," in *Proceedings of the ACM Conference on Recommender Systems, RecSys*, vol. 14, 2014.

[31] Y. Li, H. Wang, H. Liu, and B. Chen, "A study on content-based video recommendation," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 4581–4585.

[32] J. Lee and S. Abu-El-Haija, "Large-scale content-only video recommendation," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 987–995.

[33] T. Han, H. Yao, C. Xu, X. Sun, Y. Zhang, and J. J. Corso, "Dancelets mining for video recommendation based on dance styles," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 712–724, 2016.

[34] J. J. LaViola Jr, "A discussion of cybersickness in virtual environments," *ACM Sigchi Bulletin*, vol. 32, no. 1, pp. 47–56, 2000.

[35] S. Sharples, S. Cobb, A. Moody, and J. R. Wilson, "Virtual reality induced symptoms and effects (vrise): Comparison of head mounted display (hmd), desktop and projection display systems," *Displays*, vol. 29, no. 2, pp. 58–69, 2008.

[36] J. Meira, J. Marques, J. Jacob, R. Nóbrega, R. Rodrigues, A. Coelho, and A. A. de Sousa, "Video annotation for immersive journalism using masking techniques," in *2016 23rd Portuguese Meeting on Computer Graphics and Interaction (EPCGI)*, Nov. 2016, pp. 1–7.

[37] T. Matos, R. Nóbrega, R. Rodrigues, and M. Pinheiro, "Dynamic annotations on an interactive web-based 360º video player," in *Proceedings of the 23rd International ACM Conference on 3D Web Technology*, ser. Web3D '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1145/3208806.3208818

[38] M. Montagud, P. Orero, and A. Matamala, "Culture 4 all: accessibility-enabled cultural experiences through immersive VR360 content," *Personal and Ubiquitous Computing*, Jan. 2020. [Online]. Available: https://doi.org/10.1007/s00779-019-01357-3

[39] A. Brown, J. Turner, J. Patterson, A. Schmitz, M. Armstrong, and M. Glancy, "Exploring subtitle behaviour for 360 video.[white paper]," *British Broadcasting Company*, 2018.

[40] S. Rothe, K. Tran, and H. Hussmann, "Positioning of subtitles in cinematic virtual reality." in *ICAT-EGVE*, 2018, pp. 1–8.

[41] K. Li, D. Yang, S. Ji, and L. Liu, "The Impacts of Subtitles on 360-Degree Video Journalism Watching," in *2018 International Joint Conference on Information, Media and Engineering (ICIME)*, Dec. 2018, pp. 130–134.

[42] T.-Y. Chen, H.-C. Liu, and C.-Y. Hsu, "Film language analysis in society news-A case study of The New York Times," in *2017 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*, Nov. 2017, pp. 63–68.

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

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

[45] B. G. Witmer and M. J. Singer, "Measuring presence in virtual environments: A presence questionnaire," *Presence*, vol. 7, no. 3, pp. 225–240, 1998.

[46] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[47] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[48] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, Oct 2016.

[49] E. Jose, M. Greeshma, M. H. TP, and M. Supriya, "Face recognition based surveillance system using facenet and mtcnn on jetson tx2," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*. IEEE, 2019, pp. 608–613.

[50] G. Bezerra and R. Gomes, "Recognition of occluded and lateral faces using mtcnn, dlib and homographies," 2018.

[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[53] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[54] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[55] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *science*, vol. 315, no. 5814, pp. 972–976, 2007.

[56] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[57] J. B. Hirschberg and A. Rosenberg, "V-measure: a conditional entropy-based external cluster evaluation," *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[58] E. M. Ross and F. Markowetz, "Onconem: inferring tumor evolution from single-cell sequencing data," *Genome biology*, vol. 17, no. 1, pp. 1–14, 2016.

[59] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski, "Linear algebraic structure of word senses, with applications to polysemy," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 483–495, 2018.

[60] R. Layton, P. Watters, and R. Dazeley, "Automated unsupervised authorship analysis using evidence accumulation clustering," *Natural Language Engineering*, vol. 19, no. 1, pp. 95–120, 2013.

[61] C. Van Rijsbergen, "Information retrieval 2nd edition butterworths," *London available on internet*, 1979.

[62] P. Mendes, A. Busson, S. Colcher, D. Schwabe, A. Guedes, and C. Laufer, "A cluster-matching-based method for video face recognition," in *Proceedings of the Brazilian Symposium on Multimedia and the Web*, 2020, pp. 97–104.

[63] P. Mendes, E. Vieira, A. Guedes, A. Busson, and S. Colcher, "A clustering-based method for automatic educational video recommendation using deep face-features of lecturers," in *2020 IEEE International Symposium on Multimedia (ISM)*, 2020, pp. 158–161.

[64] P. Mendes, A. Guedes, D. Moraes, R. Azevedo, and S. Colcher, "An authoring model for interactive 360 videos," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2020, pp. 1–6.

[65] W. Yang, Y. Qian, J.-K. Kämäräinen, F. Cricri, and L. Fan, "Object detection in equirectangular panorama," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 2190–2195.

[66] J. P. Snyder, *Map projections–A working manual*. US Government Printing Office, 1987, vol. 1395.

[67] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[68] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[69] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[70] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[71] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[72] J. Fu, S. R. Alvar, I. Bajic, and R. Vaughan, "Fddb-360: Face detection in 360-degree fisheye images," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 15–19.

[73] V. Jain and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," UMass Amherst technical report, Tech. Rep., 2010.

[74] Y.-C. Su and K. Grauman, "Learning spherical convolution for fast features from 360 imagery," *Advances in Neural Information Processing Systems*, vol. 30, pp. 529–539, 2017.

[75] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[76] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[77] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[78] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[79] T. Chambel, M. N. Chhaganlal, and L. A. R. Neng, "Towards immersive interactive video through 360° hypervideo," in *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, ser. ACE '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2071423.2071518

[80] M. Berning, T. Yonezawa, T. Riedel, J. Nakazawa, M. Beigl, and H. Tokuda, "parnorama: 360 degree interactive video for augmented reality prototyping," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, 2013, pp. 1471–1474.

[81] Y. Ayers, A. Cohen, D. Bulterman *et al.*, "Synchronized multimedia integration language (smil) 2.0," *W3C Recommendations*, 2001.

[82] L. F. G. Soares and R. F. Rodrigues, "Nested context language 3.0 part 8–ncl digital tv profiles," *Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio*, vol. 1200, no. 35, p. 06, 2006.

[83] B. Meixner, "Hypervideos and interactive multimedia presentations," *ACM Comput. Surv.*, vol. 50, no. 1, Mar. 2017. [Online]. Available: https://doi.org/10.1145/3038925