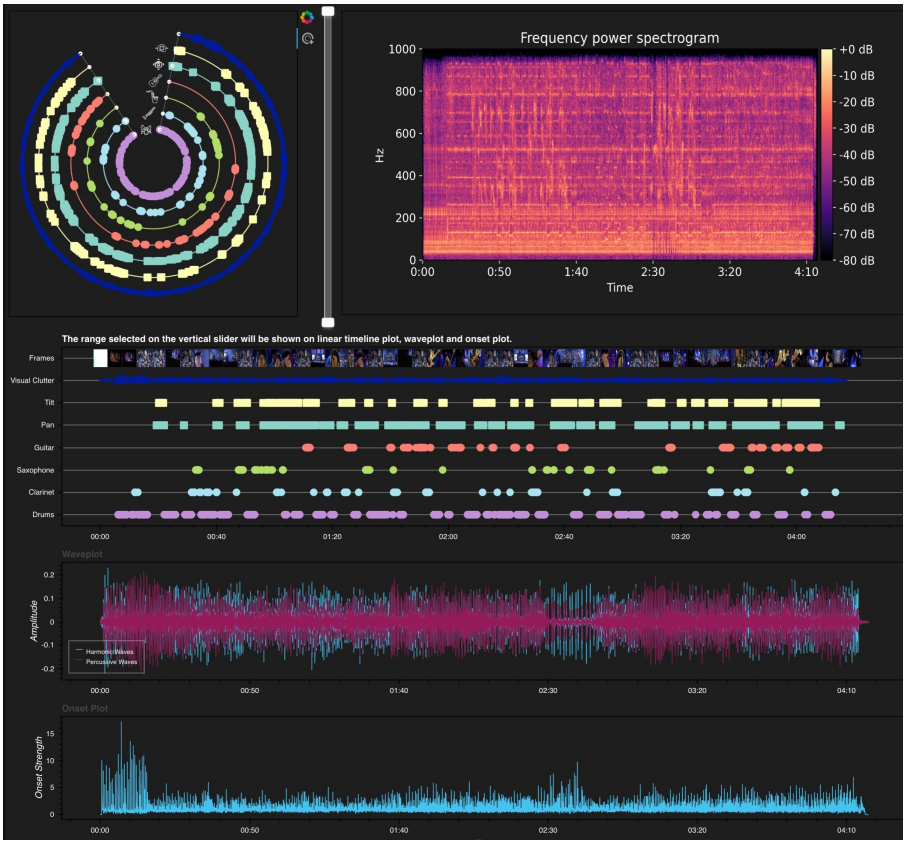
Master Thesis
Unveiling the Inner Structures of the Montreux Jazz Festival Concerts



August 25, 2022

by Melike Çiloğlu, 20-742-763

Supervisors:
Prof. Dr. Renato Pajarola
Dr. Alexandra Diehl

Visualization and MultiMedia Lab
Department of Informatics
University of Zurich

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

Acknowledgements

I would like to thank all members of the Visualization and MultiMedia Lab at UZH for supporting me throughout the thesis. Many thanks to Prof. Dr. Renato Pajarola for making it possible for me to work on this project and assisting me through it. My sincere thanks go to Dr. Alexandra Diehl for guiding the project and always being tender and accessible when needed. I am also indebted to Gaudenz Halter and Pascal Forny for patiently answering all my questions. I also thank Prof. Dr. Barbara Flückiger for making VIAN accessible to me.

Special thanks go to Alain Dufaux and Sylvain Cardin for their hospitality in Montreux and for sharing their wealth of knowledge in the field.

Finally, I am very grateful for my husband's endless patience and encouragement which led me to get a master's degree and all family members' unconditional support during this challenging time.

Abstract

The main goal of this work is to analyze the Montreux Jazz Festival video archive which was made accessible to the University of Zurich, Visualization and Multimedia Lab by "Narratives from the long tail: Transforming access to audiovisual archives" project partners. The project has been funded by the Sinergia Grant of the Swiss National Foundation. Montreux Jazz Festival video archive dates back to 1967, keeps recordings of live performances of Montreux Jazz Festival, and is listed as a Memory of the World register by UNESCO.

The problem that this thesis focuses on is the need for an online tool that will help music enthusiasts explore the archive and implement high-dimensional data analysis and visualization techniques to present different aspects of the existing recordings. The main proposed work is a web application with interactive visualizations to help the audience explore the Montreux Jazz Festival video archive. The web application is implemented by extending VIAN, a film analysis and visualization web application.

To provide a high-dimensional data visualization tool, possible video and audio analysis methods were explored and a requirement analysis was performed to determine the important features. The selected features were extracted using neural networks, signal processing techniques, and audio analysis tools.

A similarity metric was defined and implemented to compare different performance videos. Using this measure it is possible to see the outstanding songs which are significantly different than the others or to detect song clusters.

To present the extracted data, attractive interactive visualizations were designed and implemented as a tool to analyze the visual and audio patterns of videos. Both an overview visualization and detailed performance analysis views are provided. The overview visualization presents the data clusters and similarities between different recordings and the detail view visualizes the extracted visual and audio features of a video.

The final product is presented as multiple scripts to perform the feature extraction and the similarity analyses and a web application that includes interactive visualizations to help music enthusiasts explore the Montreux Jazz Festival archive.

Contents

1 Introduction

Montreux Jazz Festival video archive was started as a visionary movement by Claude Nobs who foresaw the significance of such a heritage back in 1967 and decided to keep recordings of live performances of Montreux Jazz Festival. In 2013, UNESCO listed the archive as a Memory of the World register [1]. The goal of this work is to analyze the Montreux Jazz Festival video archive which was made accessible to the University of Zurich, Visualization and Multimedia Lab by "Narratives from the long tail: Transforming access to audiovisual archives" project partners. The project has been funded by the Sinergia Grant of the Swiss National Foundation.

The Montreux Jazz Digital Project aims to use the archive both educationally and for scientific research and also to present it to the public for entertainment purposes. The archive is being explored by multiple research labs interdisciplinary across fields of audio processing, image and video processing, virtual reality research, and social sciences such as heritage studies and culture. There is a need for an online tool that will help explore the archive and present different aspects of the existing recordings to provide a common ground for people coming from different backgrounds.

Music visualization uses Music Information Research (MIR) to extract meaningful features to visualize, which is an ongoing field that still has important challenges to solve. Literature provides multiple music visualization tools that use audio analysis. However, the existing tools lack using video related features that would provide more information about the song and the mood. The existing visualization tools focus on studio recorded songs and overlook the effects of a live performance ambiance on the songs. No work analyzing live concert videos both in audio and video aspects and providing an interactive visualization tool comes to our notice.

A literature review reveals multiple visualization tools that are designed to communicate similarity based music clusters. In addition to that, multiple other tools visualize music based on waveform or notes that are played. However, there were no examples of tools that both provide a similarity overview of a clustering layout and provide a detailed analysis view for each song.

Previous music visualization studies used a limited amount of features to analyze the songs. Most studies have relied on the notes and chords, some also analyzed musical features such as tempo, genre, mood, etc. Nonetheless, the literature lacks a visualization tool that analyzes musical performance videos in detail using an extensive amount of features, including both video and audio related features to analyze musical instruments, frequency spectrum, or visual clutter.

To provide a high-dimensional data visualization tool, possible video and audio analysis methods were explored and a primary list of extractable features was created. After performing a requirement analysis with the help of an area expert, the important features that are needed for the analysis of live jazz song performances were determined. The selected features were extracted using neural networks, signal processing techniques, and audio analysis tools.

From the set of extracted features for live performance videos, a subset was selected as crucial features to consider while comparing different songs. Using this subset, a similarity metric was defined and implemented. The defined metric returns a distance scalar for each recording that measures how different the relevant recording is compared to all other videos. Using this measure it is possible to see the outstanding songs which are significantly different than the others or to detect song clusters.

To communicate the extracted data to the general audience in an engaging and attractive way, a set of interactive visualizations was designed. The visual design was implemented as a tool that the

users can interact with and analyze the visual and audio patterns of videos. The end product provides a web application and interactive visualizations to help music enthusiasts explore the Montreux Jazz Festival archive. Both an overview visualization and a detailed song analysis view are provided. The overview visualization communicates clusters and similarities between different recordings. The detail view presents extracted visual and audio features of a song.

This thesis comprises eight main chapters. Chapter 1 is to provide the reader with a motivating introduction to the topic and a brief description of the work done.

In Chapter 2, an analysis of the existing related work is carried out. Research papers on similarity based music collection visualization, music visualization, and video visualization are addressed. The related work is put into perspective and the differences between this thesis and existing work is highlighted.

Chapter 3 presents the problem statement, provides information on input data, and explains the requirements of this thesis in detail.

In Chapter 4 the solution that this work provides is explained. Each step taken to approach the problem has been presented.

In Chapter 5 the implementation details are explained. The technological decisions and the structure of the code are described.

Chapter 6 presents example use cases of the end product.

Chapter 7 presents the limitations of the implemented work to provide a complete overview.

Chapter 8 discusses the results of the implementation.

Finally, Chapter 9 concludes this thesis by presenting motivation for possible future work on the provided visual analysis tool.

2 Related Work

This chapter presents similar works to present one, as revealed by the literature review. To our knowledge, the literature lacks examples of visualization tools designed for live musical performance videos. Hence as related work, both music visualization tools and video visualization tools were considered. The structure of this chapter is as follows:

- Section 2.1 summarizes music collection visualizations based on a similarity metric.
- Section 2.2 lists related works that focus on describing single songs based on feature extraction.
- Section 2.3 lists example works in the intersection of the Section 2.1 and Section 2.2. Works summarized here are to provide both similarity based collection visualization and song detail visualization.
- Section 2.4 lists related video visualization works.

The literature review revealed that there are no examples of live concert video visualizations that target a general audience which provides both an overview to compare multiple videos and a detailed view to present the video features.

2.1 Similarity Based Music Collection Visualization

Multiple works define and implement similarity metrics to compare songs and implement visualization tools to describe the distance between songs. Related work to music collection visualization using a similarity metric is listed here.

2.1.1 AudioRadar: A Metaphorical Visualization for the Navigation of Large Music Collections

The work by O. Hilliges et al. proposes an exploratory interactive visualization tool for music collections. It uses radar as a metaphor [2]. As shown in Figure 2.1, the selected song is added to the center point of a circular visualization and similar songs are placed around it. The comparison is based on four main dimensions, whether the song is melodic or rhythmic, slow or fast, clean or rough, calm or turbulent. The speed of the song refers to the beats per minute, cleanliness refers to the noisiness and calmness refers to the change of noise levels throughout the song.

Audio is automatically analyzed and assigned scalar values for each dimension. Each song represents a point in a four-dimensional space. Later the four-dimensional space is projected into a two-dimensional one to communicate the song's similarities to the user. The projection is done by simply ignoring two dimensions according to the user input. The distance between songs in two-dimensional visualization space maps the song similarities.

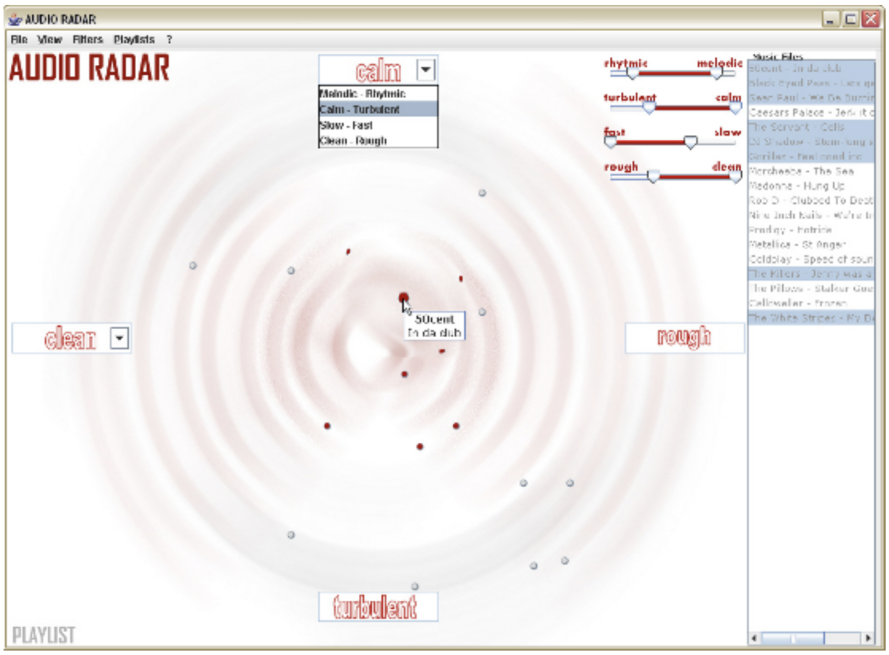


Figure 2.1: AudioRadar [2]

2.1.2 MuVis: An Application for Interactive Exploration of Large Music Collections

MuVis is a visualization tool that utilizes treemaps for browsing and filtering songs to create playlists [3]. There exists a feature extraction module. The extracted features are tags, track title, duration, genre, song release date, and fluctuation patterns. The tool provides similarity based positioning according to the properties of songs which helps users add similar songs to a playlist.

It is an interactive tool and users can filter the features and re-structure the treemap to explore more. It is possible to write queries to navigate around the treemap easily.

The work does a user study and assigns users four tasks including searching a specific song, creating a playlist, and filtering the playlist. The time required to achieve these tasks was compared to achieving the same tasks on Windows Media Player. It was observed that with MuVis average time required for each task is significantly smaller.

2.1.3 A Visual Exploration of Melodic Relationships within Traditional Music Collections

The work by C.Walshaw proposes a visualization technique to explore relationships between folk and traditional songs using an existing melodic similarity measure [4]. The main goal of this work is to compare different music collections.

In this work datasets with multiple corpora are considered. For each pair of songs in the dataset, the similarity metric between them is calculated. Then using these similarity scores, corpus graphs, shown in Figure 2.2, were constructed using the node-link representation. Nodes represent songs, links represent thresholded similarities and different corpora are color coded. With the help of color coding, it is possible to compare multiple collections and explore similarities between them. Unlike previously mentioned tools the implemented visualization is not interactive.

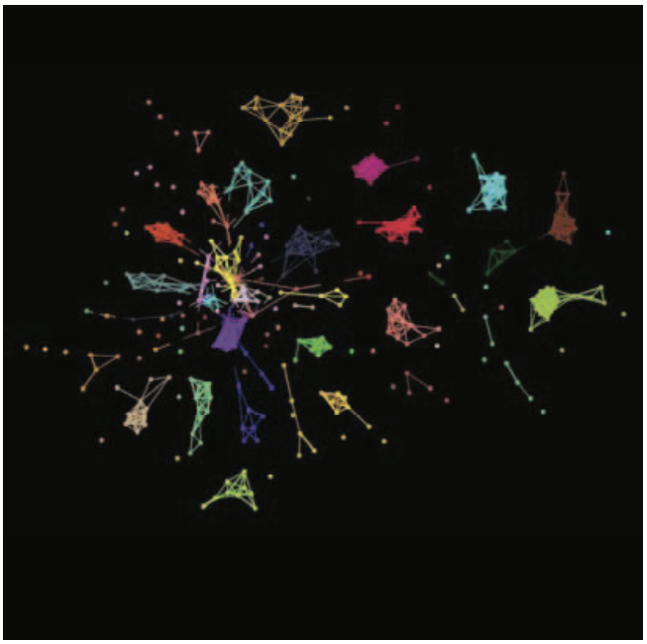


Figure 2.2: Corpus similarity graphs [4]

2.1.4 Similarity Graph: Visual Exploration of Song Collections

This work proposes a new visualization that considers similarities that can occur between small parts of songs [5]. It does not compare songs as a whole but also investigates similarities that can take place in some sections.

The visualization is shown in Figure 2.3 and consists of an overview and detailed view named global similarity graph and local similarity graph respectively. The local similarity graph is presented in a circle in the figure. The local similarity graph uses Bézier curves to connect similar song segments. The global similarity graph uses t-SNE to map the songs into two-dimensional space and connects songs according to overall song similarities using Hermite curves. The visualization is interactive and allows users to explore the collection freely. Color coding was used to highlight different artists' covers of the same songs.

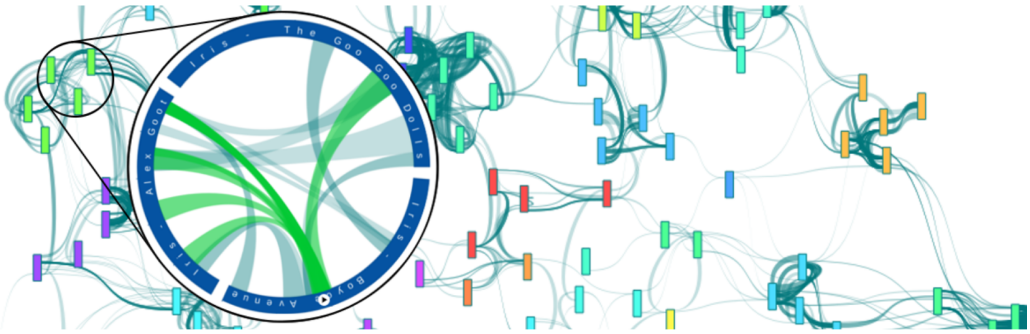


Figure 2.3: Similarity Graph [5]

2.2 Music Visualization

This thesis proposes a visualization tool that can help compare multiple songs based on similarity and also can present each song's features in detail. The previously mentioned related work focuses more on

visualizing song similarities than describing each song individually using its features. Related works that focus on describing single songs based on feature extraction are listed here.

2.2.1 Visualizing the Semantics of Music

SongVis is a visualization tool designed to present the semantic descriptors of music using icons [6]. First, the work conducts an online questionnaire to determine semantic descriptors of music. According to the user input danceability, tempo, mood, genre, and dominant musical instrument were selected as the semantic features to describe the song.

Songs are analyzed to extract the semantic features using music information retrieval techniques and for each feature, icons were assigned according to the value. For example according to the speed of the tempo whether a rabbit icon or a turtle icon is assigned to the song. An example song visualization is shown in Figure 2.4.

To give more insight into the flow of the song depending on the time it uses colored bars where the colors are based on the properties of the waveform. The resulting visualizations for songs are easy to interpret even for the general audience, engaging thanks to the icons and attractive. However, it does not provide an explicit way to compare multiple songs according to their similarities.

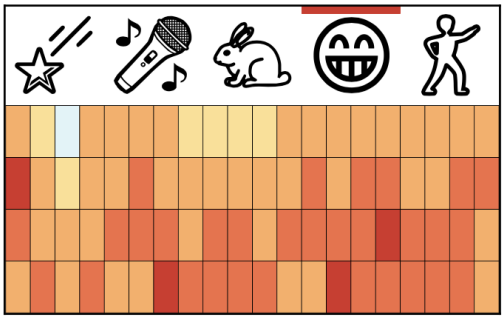


Figure 2.4: SongVis [6]

2.2.2 Misual: Music Visualization based on Acoustic Data

Misual is a music visualization tool that uses the waveform of audio as the main source of information [7]. The wave strength over time is smoothed through a moving average function and visualized. An example Misual of a song is shown in Figure 2.5. The visualization uses lighting and shading techniques to make it appear 3D. Hence the volume of the shape encodes the power of the audio signal. Repetitions through the song are detected using the mel-frequency analysis and they were color coded. It is easy to identify the chorus via the coloring.

The main goal of the visualization is to help people categorize music. The design is intuitive and easy to understand. However, the features considered are limited and the work does not help analyze musical data.

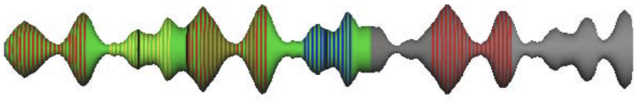


Figure 2.5: Misual [7]

2.2.3 Isochords: Visualizing structure in music

Isochords is an animated visualization that aims for music classification based on structure [8]. It uses MIDI files to extract the musical events and visualizes them in a Tonnetz grid. An example Tonnetz grid is shown in Figure 2.6. It encodes major and minor chords using the arrow orientation.

Isochords is designed as an animation to be used as an ear training aid. It allows users to see the musical events while they are also listening to them happening. Also, the paper provides examples that present how the visual representation of the music of different genres differs clearly.

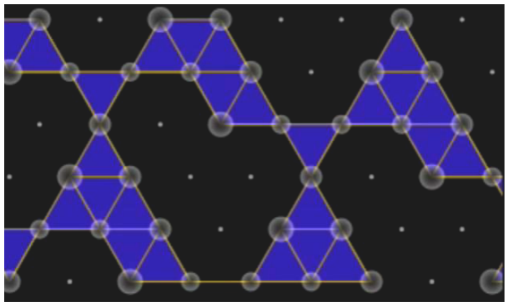


Figure 2.6: Isochords [8]

2.2.4 ImproViz: Visual Explorations of Jazz Improvisations

ImproViz is an important related work to consider as the focus of it is Jazz music visualization. ImproViz is a visual design to analyze jazz improvisations to explore the harmonic styles of different musicians [9]. The visualization was implemented for the song "All Blues". Manual analysis is needed to prepare the visualization. In this study, different musicians' harmonic palettes were explored. Even though just like this work the main focus is jazz music visualization, ImproViz only visualizes one song and considers only the notes while doing so.

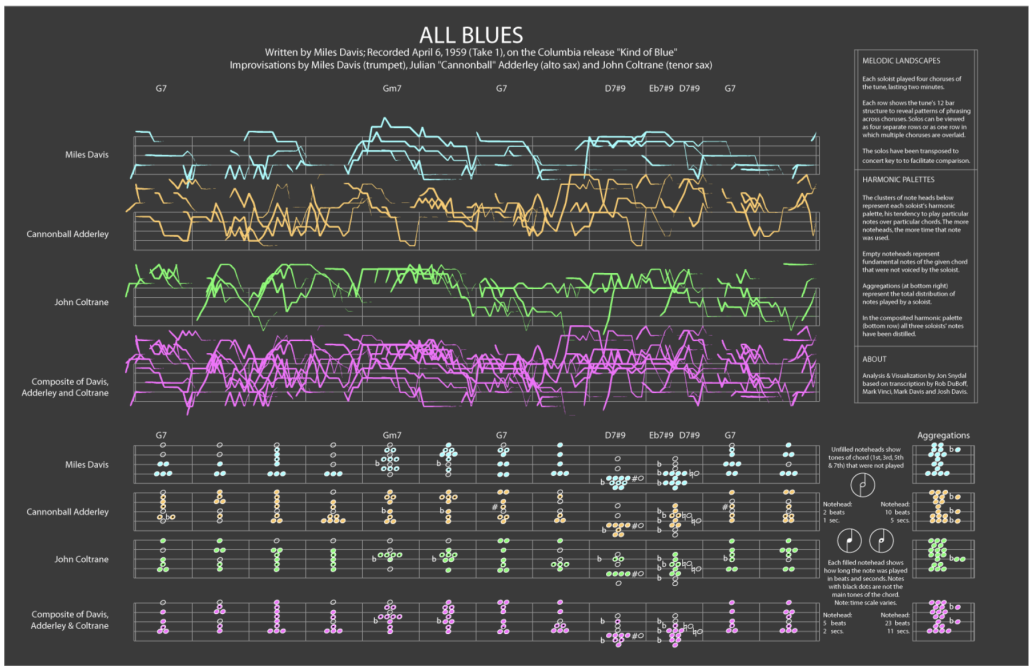


Figure 2.7: ImproViz [9]

2.3 Music and Similarity Based Music Collection Visualization

All tools mentioned above, either focus on music collection visualization, and it is impossible to visualize single songs, or focus on single song visualization and ignore similarities between songs. This section includes an example that relies on the intersection of the previous two sections.

2.3.1 Visualization of music collections based on structural content similarity

The work of A. Soriano et al. provides both similarity based collection visualization and song detail visualization. It constructs unique icons for each song depending on the MIDI file format because MIDI files encode properties of individual notes played during the song [10]. The resulting icons are vertical bars that are composed of multiple bars of different colors and sizes according to the underlying structure of the song. An example icon created for a song is presented in Figure 2.8. It is possible to see an overview of all songs on a scatter plot where they are placed according to similarity and also it is possible to see the icons of each song closely to get more information about the song. The clustering graph can be seen in Figure 2.9. However, even though using MIDI files is a great way to create a similarity metric, the icons created based on them are not very intuitive or easy to understand for inexperienced people.

Figure 2.8: An example icon created for a song [10]

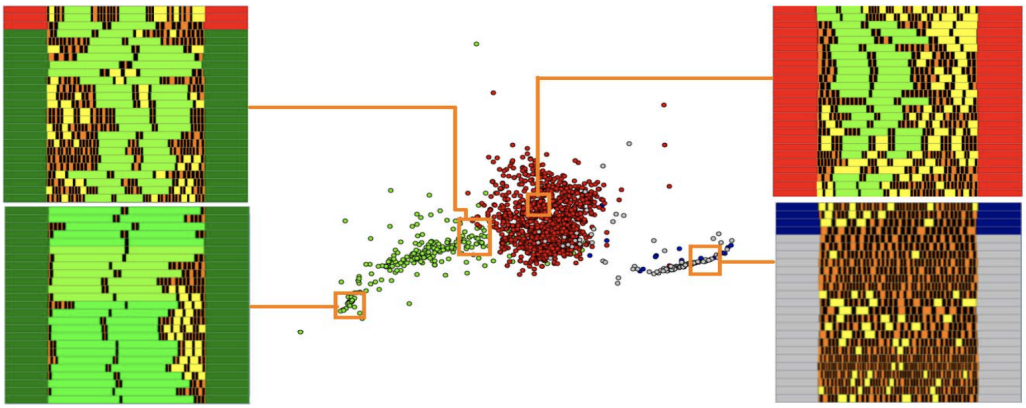


Figure 2.9: Clustering graph [10]

2.4 Video Visualization

All of the considered related work so far was about music visualization rather than concert video visualization which is this thesis' focus. The conducted literature review did not reveal a work specifically focusing on live performance video visualization. Hence, here, a list of video visualizations is provided as the related work including a music video visualization tool.

2.4.1 Affective Visualization and Retrieval for Music Video

i.MV is a music video visualization tool that is the closest work to this one in terms of the data analyzed [11]. *i*.MV extracts both visual features like lighting, and saturation and also audio features like tempo and beat strength. It applies affective analysis in which it classifies music videos according to the feelings they arouse and color codes according to the evoked feelings. The visualization is shown in Figure 2.10. It is a tool to classify and compare music videos but does not visualize the extracted features and does not provide a detailed view of each music video.

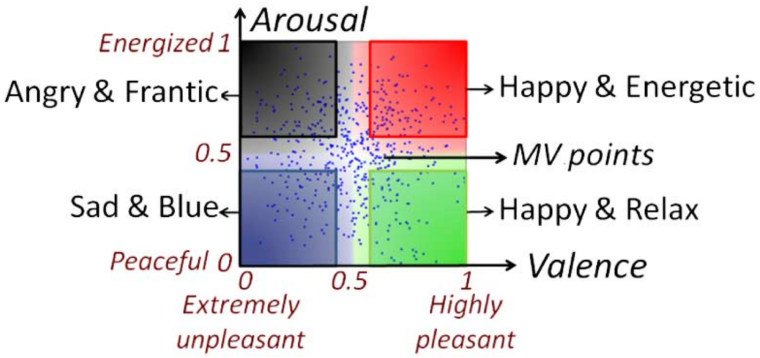


Figure 2.10: *i*.MV [11]

2.4.2 Stained-Glass Visualization for Highly Condensed Video Summaries

This work proposes a technique for creating visual video summaries [12]. It analyzes the videos to extract the sections where high activity and motion occur. These sections determine the keyframes. Later from keyframes, the areas of interest and context are extracted. The irregular shapes of areas of interest are glued together to create a visual summary having a stained-glass look as shown in Figure 2.11.

Figure 2.11: Stained-Glass Visualization [12]

Even though the visualization provides a great understanding of the theme of the video, the keyframes are not being presented in chronological order and the temporal data is completely lost.

2.4.3 ColorsInMotion: Interactive Visualization and Exploration of Video Spaces

ColorsInMotion is an interactive visualization of videos considering cultural aspects, color dominance, and movement [13]. Motion aspects can be represented by average scene loops that blur the pixels in which motion occurs. Colors can be shown using average color and dominant color loops. It is possible to search videos according to the colors.

The work implements a video analyzer module and visualization module inputs processed videos by the analyzer module.

3 Problem Statement

This chapter presents the problem that this thesis solves, and introduces the assumptions on analyzed data and the requirements for a possible solution.

The problem tackled in this work is the lack of high-dimensional data analysis and visualization tools to explore live concert videos.

The data used is the Montreux Jazz Festival video archive of Claude Nobs Foundation. The videos are in mp4 format and each video includes one song performed live by artists during Montreux Jazz Festival. Since the videos were recorded during live performances the audio includes cheering crowds or noise, unlike studio recorded songs. Videos date back to 1967, hence there are some recordings that are lacking in video and audio quality relative to today's technological standards.

The requirements for this thesis can be listed as follows.

Requirement 1: Feature extraction

Through a requirements analysis, essential features to consider during the analysis of performance videos should be listed. These features can include visual components that depend on video frames or might depend on audio components. After determining essential features in the context of live recording analysis, possible implementation techniques should be searched. Finally, feature extraction should be implemented in Python. A script that will obtain features from song performance videos in mp4 format should be presented.

Requirement 2: Similarity Metric Definition and Implementation

A similarity metric to compare different video recordings should be defined. Essential features to consider while comparing different videos should be selected. Then a distance function should be constructed to characterize how far apart the features of two different videos are. The outcome is the implementation of the defined metric in Python which returns a scalar, when presented with two videos, stating how similar the videos are.

Requirement 3: Data Model Design

A data model should be designed. The metadata and the extracted features should have been integrated into a data model. A PostgreSQL database structure should be prepared and an automatic way to insert the extracted features into the database should be implemented.

Requirement 4: Visualization Interface Implementation

An attractive and engaging visualization should be designed to communicate the extracted features to the general audience. The visualization should be presented as a web-based application implemented using front-end technologies such as NodeJS, BokehJS, or Vue.js. On the backend, Python should run to receive the extracted features from the PostgreSQL database.

The visualization should allow users interactively analyze the video and features over time. It should be simple and easy to understand as it targets a general audience but also provide insights into the visual and audio components of the videos. Also, it should be possible to explore videos by similarity and compare them.

4 Technical Solution

This chapter introduces the solution this thesis proposes for the problem explained previously. The outline of this chapter is as follows:

- Section 4.1 introduces the feature extraction stage and explains the selected features.
- Section 4.2 introduces the designed data model.
- Section 4.3 explains the proposed similarity metric and how it is calculated.
- Section 4.4 presents the prepared visual designs.
- Section 4.5 introduces the designed and implemented web application and its functionalities.

4.1 Feature Extraction

The main goal of this work is to explore the Montreux Jazz Video Archive and implement high-dimensional data analysis and visualization techniques. The first step of this data analysis was to determine the relevant features and search existing feature extraction methods. To gain more insight into the videos and provide a comprehensive analysis, it was decided to extract both visual and audio features. The overview of the extracted features is provided in Table 4.1.

This section introduces the extracted features in detail where Section 4.1.1 presents visual features and Section 4.1.2 presents audio features.

4.1.1 Visual Features

The following is the list of visual features that were extracted from the live concert video frames using image processing techniques.

- Camera Motion is explained in Section 4.1.1.1.
- Visual Clutter is introduced in Section 4.1.1.2.
- Musical Instrument detection is explained in Section 4.1.1.3.

4.1.1.1 Camera Motion

Using various camera motions to control the audience's attention is a strong and widely used method applied by directors. It is our point of interest to know how often the panning, tilting, and zooming were used throughout the video and whether it reveals a pattern. This is a global feature which could be applied to a variety of videos and is also interesting to explore in the context of a concert video. For example, it could be possible to see a pattern where a musician is performing a solo, the camera zooms in towards the artist or the musical instrument.

Using the optical flow between consecutive frames it is possible to calculate the dominant flow angle and magnitude. Knowing this vector and comparing it through the scenes makes it possible to predict the camera motion.

Visual Features			
Feature	**Method**	**Temporality**	**Output Type**
Camera Motion	Optical Flow	Time Dependent	Motion name (if detected)
Visual Clutter	Feature Congestion	Time Dependent	Clutter Scalar
Musical Instrument Detection	Convolutional Neural Network	Time Dependent	Probability values per each instrument
Audio Features			
Feature	**Method**	**Temporality**	**Output Type**
Tempo	Audio Processing	Not Time Dependent	Beats Per Minute
Tempogram	Onset Detection	Time Dependent	Onset Strength and Time
Power Spectrogram	Signal Processing	Time Dependent	Signal Strength Per Frequency
Waveform	Signal Processing	Time Dependent	Signal Amplitude Over Time
Dominant Musical Instrument and Instrument Family	Neural Network	Not Time Dependent	Probability values per each instrument and instrument family
Dominant Mood and Genre	Neural Network	Not Time Dependent	Probability values per each mood and genre

Table 4.1: Overview of extracted features.

4.1.1.2 Visual Clutter

The cognitive load on an audience that watches a video is related to visual clutter in the frames of the video. Clutter is again a context free feature that can reveal interesting patterns in different types of movies. To measure visual clutter feature congestion method was used. According to the feature congestion technique, the clutter on an image is correlated to the variance of luminance contrast, color, and orientation of the image. Presuming that if these properties change significantly through an image, the image is cluttered.

4.1.1.3 Musical Instrument Detection From Video

While analyzing a video, one of the most important tasks is to determine the remarkable points of scenes. The simplest way to discover the focus of a scene is through applying object detection to the frames to extract the visible objects. In the context of a live concert video recording, it is valuable to extract the musical instruments on the scene. For this purpose, a pre-trained convolutional neural network for musical instrument detection was used. The model was run on every frame of the concert videos and for each frame, it returns 30 confidence values representing the probability of 30 instruments being visible on the given frame. Then the confidence values are thresholded so that only the ones with high confidence values are considered.

4.1.2 Audio Features

After analyzing the video frames, the audio was extracted and analyzed. The following audio features were considered to be crucial while studying concert recordings and these features were calculated using audio and signal processing techniques.

- Tempo is explained in Section 4.1.2.1.
- Tempogram is introduced in Section 4.1.2.2.
- Power Spectrogram is introduced in Section 4.1.2.3.
- Waveform is presented in Section 4.1.2.4.
- Dominant Musical Instruments and Instrument Families are introduced in Section 4.1.2.5.
- Dominant Mood and Genre detection is presented in Section 4.1.2.6.

4.1.2.1 Tempo

The tempo of a song is a powerful tool for composers to convey a feeling to the audience. In the context of analysis knowing the tempo might help to understand the mood during the concert, audience reaction, and even the genre of the song. Even though there is a possibility of tempo changing through a live performance of one song, here one single averaged tempo was calculated in the unit of BPM.

4.1.2.2 Tempogram

The tempo of a song is an important metric to consider but it might not be enough to provide temporal insight into beats. To have a better understanding, it would be useful and compelling to see whether the song speeds up or slows down during a live performance. On studio recordings, it is very unlikely to observe varying tempo in a jazz, rock, or pop song. However, in the case of live performances for exploratory data analysis, it was decided to look more into tempo in a time-dependent manner. One way to analyze this is to look at the initiation point of musical events. These are called onset and they are directly related to the tempo of music at a given time.

4.1.2.3 Power Spectrogram

Power Spectrogram shows various frequencies' signal strengths over time. It provides a quick overview of the audio signal and helps people grasp the dominant frequencies through the song. The spectrogram is extensively used and is crucial for music analysis. Even though it is mostly used by area experts, it is easy to understand for the general audience.

4.1.2.4 Waveform

Waveform describes the amplitude of the audio signals over time. It is the most basic form of audio visualization and the general audience is usually familiar with it. Unlike tempogram and spectrogram, in the waveform, the sound is visualized exactly as it is without preprocessing. This makes the visual very intuitive for the general audience, the wave rises as the sound increases and it falls as the sound decreases.

In this case, the wave has been separated into two main components harmonic and percussive waves. Harmonic waves are created by the melodic sounds and percussive waves are created by the beats like hits on drums. This separation helps users analyze which component is dominant over time.

4.1.2.5 Dominant Musical Instruments and Musical Instrument Families

While analyzing a song, one of the most important tasks is to automatically detect the musical instruments of the song. Musical instrument detection in polyphonic songs is still an open challenge for which no definitive solution exists in the literature. There exist neural networks that predict the dominant musical instruments and instrument families. In this work, a pre-trained sound classification model is used to extract this feature.

4.1.2.6 Dominant mood and genre

A common way to structure playlists is according to the mood and genre of the songs. Hence one can say that the mood and genre can be considered to be crucial when analyzing songs. These properties are usually related to tempo and musical instruments but are not completely determined by them. Therefore it is important to explicitly extract and present them. This thesis uses a pre-trained neural network to predict the dominant mood and genre.

4.2 Data Model

To store the metadata and the extracted features first a data model was designed based on the input data and the web application, then a database was structured according to the designed model.

The input data of this work consists of performance videos recorded at Montreux Jazz Festival and metadata about the concerts. For each concert, there are multiple videos, each of which contains one song. The metadata of the concerts consists of the date of the concert, in which hall the concert was performed, and the performing artists.

For the design of the data model, it was planned to keep concert information in one table. The information about the date and the location of the concert is planned to be stored on the same table. Then in a separate table, the information about the video recordings will be stored. Some data planned to be stored in this table are the duration of the video, the title of the song performed on the video, and to which concert the video belongs to. For each concert, there could be zero to multiple videos but for each video, there exists only one concert it belongs to. Finally, the extracted features were planned to store in a separate analysis table. The features are planned to be encoded and stored in binary format. Hence important information such as the data type, or data shape to decode the data from binary to original form must be stored.

The implementation section provides more information on the table details, relationships between the tables, columns, and data types.

4.3 Similarity Metric

To be able to compare different video inputs this work proposes a similarity metric. To define a similarity metric a literature review was performed and various distance metrics were evaluated. One of the first distance measures considered is the Minkowski distance and its specific case Euclidean distance. The Euclidean distance is one of the most common measures, it is easy to implement and interpret. One of the main disadvantages is the role played by the units. If one of the features has greater values due to its units, it will dominate the metric. It is possible to apply normalization beforehand to get unbiased outputs. It performs well when the data has highly separated clusters but it does not consider correlation. Since extracted features are expected to have correlation it is better to use a distance designed to consider the correlation.

Another popular similarity metric is cosine similarity. It neglects the vector lengths and calculates similarity solely according to the angle between two vectors. For example, lets define two example feature vectors $u = [0.5, 0.5]$ and $v = [0.98, 0.98]$ which represent the probability of guitar appearances on first two frames of two different videos. These vectors lie in the same line. Hence according to the cosine similarity, they are the same. However, they represent significantly different probabilities and semantically the compared frames are apart.

Chebychev and Manhattan distances were other considered metrics but they suffer from the same unit problem as the Euclidean distance. In the concert video feature space, not all features share the same unit. Also, both of these distances neglect possible correlations in data.

Mahalanobis distance is a measure that computes the distance between a point and a data distribution. It computes how far a point is from the mean of a distribution in terms of standard deviation along the principal component axis. The formulation is as follows:

$$d_{mah}(p, D) = \sqrt{(p - \mu)C^{-1}(p - \mu)^T}$$

where p is a data point, D is a distribution, μ is the mean value vector of D and C is the covariance matrix of D.

Figure 4.1 compares Euclidean distance and Mahalanobis distance graphically. Both in (a) and (b) 20 data points are shown. The circles in (a) are isolines where the Euclidean distance to the origin along the circle is the same. For the given data the ellipses in (b) are again the isolines where the Mahalanobis distance to the origin along the curve is the same [14].

Since Mahalanobis distance uses principal component analysis, it considers data correlations. It is expected to see correlations between different features extracted from the videos. For example, musical instruments detected from the frames and the audio would be correlated. In this regard, Mahalanobis distance is one of the best options. Also, other important properties are that it is scale-invariant and unitless. In our dataset, there is no consensus on the units of the feature outcomes. Most features result in probability values, but, for example, visual clutter returns clutter scalars. Using Mahalanobis distance there won't be any need for a normalization step to make the measure unitless.

To be able to define a similarity metric some features were prioritized to compare and structured as follows: For the musical instruments, the result is a matrix, M, of size (30 x Number of Frames in the Video). t is a decision threshold function where

$$t(x) = \begin{cases} 0 & \text{if } x \leq 0.5 \\ x & \text{if } x > 0.5 \end{cases} \tag{4.1}$$

and p_{ij} represent the probability of instrument i appearing on frame j. Then we can express $M_{ij} = t(p_{ij})$.

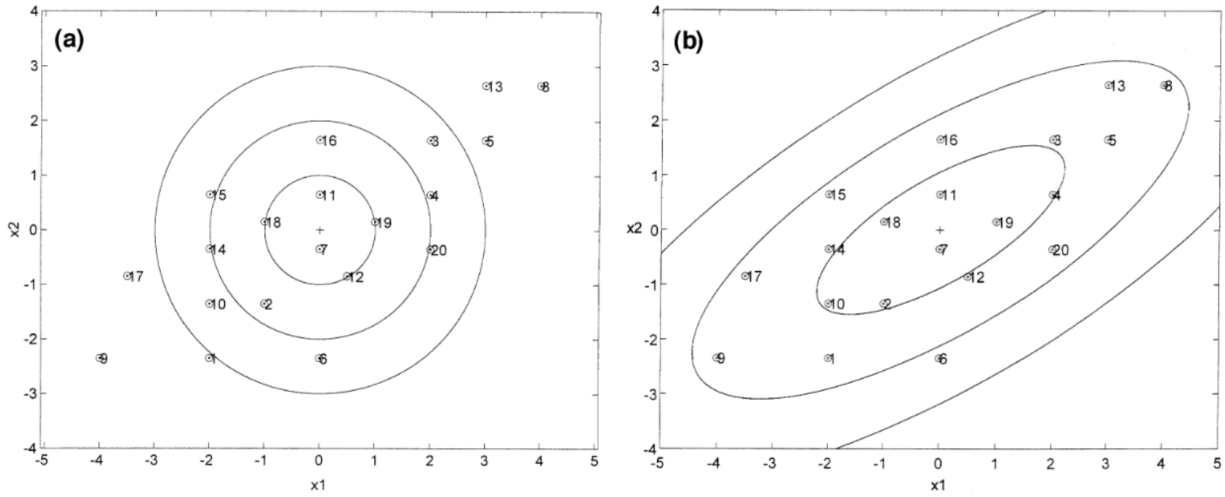


Figure 4.1: Euclidean and Mahalanobis distances comparison [14]

For the camera motions the outcome is a matrix, C, of size (3 x Number of Frames in the Video) where C_{ij} is 1 if motion i is detected on frame j, 0 otherwise.

The visual clutter feature is a vector of scalars with elements as many as the frames of the video.

The outputs of Dominant Musical Instruments and Dominant mood and genre features for all videos are vectors of probabilities for each one of the instruments, mood, and genre classes. Overall the lengths of these vectors are the same for each video.

To implement the similarity metric two possible options were considered. The first option is to flatten all of the features and combine them into one long vector for each video. Then apply the Mahalanobis distance formula to the vectors. The other option is to apply the Mahalanobis distance formula to individual features and combine the resulting scalars into one distance value. With the second option it is easier to interpret the role of each feature on the distance and also the result is simpler to explain to the users. Hence the second option was chosen.

One of the most important points to consider is that for some features the vector or matrix sizes are related to the length of the video. The formulation of the Mahalanobis distance assumes that the vectors are of the same length. One easy fix for this issue is to use zero filling to make all the vectors the same length as the longest video. Geometrically speaking this would include the given vectors in a higher dimensional space in a standardized way.

The algorithm to calculate the distance is explained in Figure 4.2. After feature extraction, feature vectors are formed. For musical instrument detection, visual clutter, and camera motion, since the lengths of the vectors are related to the length of the video, zero filling is applied to make vectors of the different videos the same length. For dominant musical instruments, mood and genre detection is not needed. In Figure 4.2 p_{ij} represents the feature vectors where i is the index of the data point and j is the index for the feature.

After calculating feature vectors, for each feature, one matrix is formed by combining the vectors for all the data points. Feature matrices are represented by V_j where j is the feature index. Each row of V_j consists of a feature vector of one data point for the j^{th} feature

In the next step, Mahalanobis distance matrices are calculated. The covariance matrix needed for the calculation are defined individually for each feature. For example, for guitar detection, the guitar detection feature matrix is used to calculate the covariance matrix. For the calculation of the point to point distances the following definition was used

$$d_{mah}(p_{ik}, p_{jk}, C_k^{-1}) = \sqrt{(p_{ik} - p_{jk})C_k^{-1}((p_{ik} - p_{jk})^T}$$

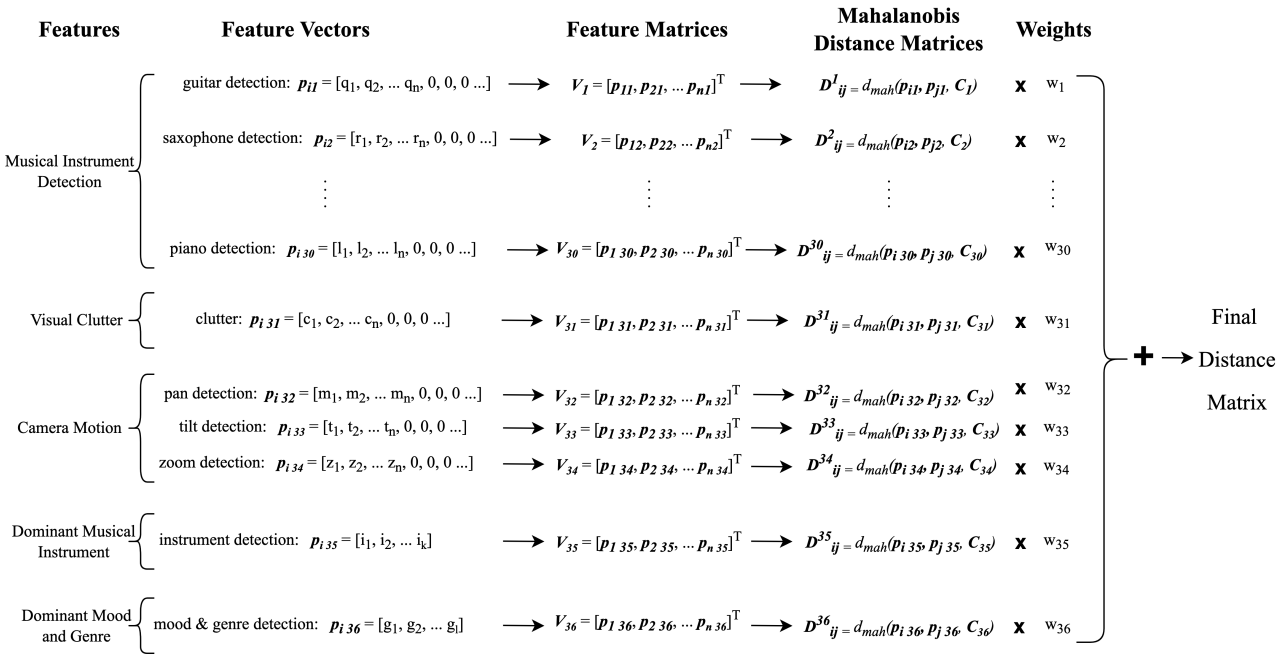


Figure 4.2: A visual explanation of the distance metric calculation.

where p_{ik} and p_{jk} are i and j^th data points' k^{th} feature vectors. For each pair of data points i, j the distance is calculated for every feature k and the results are combined into one distance matrix. ij^{th} element of the distance matrix D^k gives the distance between feature vectors p_ik and p_jk.

Finally, a weighted sum of the distance matrices is performed to get the final distance value. The default values for all the weights would be 1/36 to make the weighted sum into an averaging function. However having this weighted sum in the algorithm, allows users to choose which vector and feature have more influence on the result. According to the use case, some features might be more important than others and their roles in the distance can be manipulated.

For visualization purposes, it was aimed to calculate the locations of the songs on the two-dimensional space using the pairwise similarity values. For this purpose, multi-dimensional scaling is used. Multi-dimensional scaling is an algorithm to map objects with known dissimilarities into a configuration of points in a Euclidean space where original dissimilarities between objects and Euclidean distances between points match as well as possible [15]. Hence with the help of multi-dimensional scaling, after having pairwise similarity values, it is possible to project the performance videos into two-dimensional Euclidean space.

After calculating the two-dimensional positions of the performance videos in the Euclidean space K-means clustering algorithm is run to detect the clusters in the dataset. K-means clustering algorithm determines k clusters where the total distance from data points to their nearest cluster center is minimized [16]. K in the K-means clustering algorithm is a hyper-parameter and is chosen to be three in this thesis.

The outcome of the similarity calculation phase is the locations of the performance videos in the two-dimensional Euclidean space and the three clusters constructed using the K-means algorithm.

4.4 Visual Design

To visualize the Jazz concert videos a three-layered visualization scheme was designed. The first layer is called similarity based overview plot. It aims to visualize all of the songs and compare them based on the proposed similarity metric. This visualization follows the structures of a clustering graph and

the individual features of the songs are not relevant.

The second layer is called the performance overview plot. It acts as an attractive summary of to song and presents a selected subset of the features. Using this plot it is possible to compare the features of a few songs. However, this view does not present all of the features and is not ideal for a detailed analysis of individual songs.

The final layer is called the performance detail plot. All of the extracted features are presented in detail, it is even possible to visualize the frames of the video over time. It provides insight into the song and is designed for a detailed analysis of individual songs.

In this section, the visual design of all three layers is explained in detail.

- Section 4.4.1 introduces how the similarities between songs are being communicated to the users.
- Section 4.4.2 explains the overview plot designed for performances.
- Section 4.4.3 explains the detail view for performances.

4.4.1 Similarity Based Overview Plot

The aim of the Similarity Based Overview Plot is to present all the songs in a single visualization to allow comparison. The design should be attractive, easy to understand and should present clusters in the data. This thesis proposes a similarity metric and as explained in Section 4.3 after calculation of pairwise song similarities, the songs were mapped into two-dimensional locations and clustered using K-means clustering. To present the clusters, a scatter plot is designed as it is the simplest way to visualize the song locations depending on the similarities.

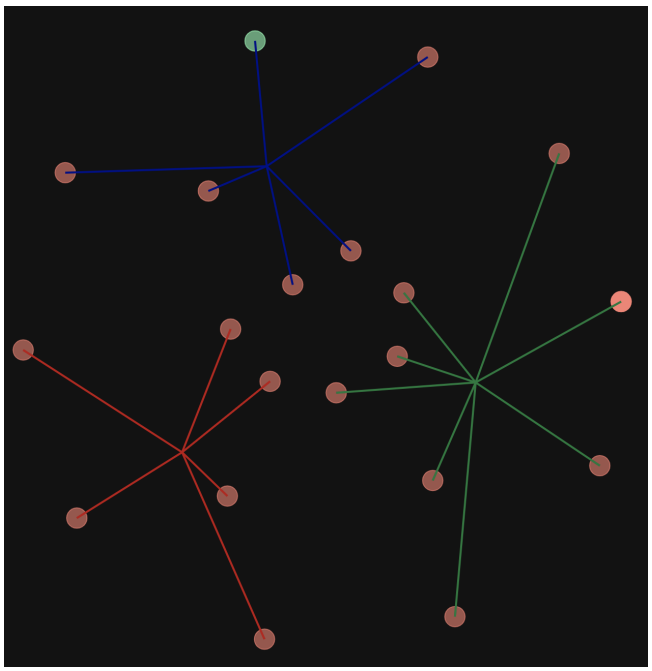


Figure 4.3: An Example Similarity Based Overview Plot

Figure 4.3 presents the design. The locations of the circles of the scatter plot directly encode the calculated locations using multidimensional scaling. Hence the visual distance between songs maps to their similarity that is measured according to the proposed metric.

The clusters determined using the K-means clustering algorithm are presented to users via the lines. The lines are drawn between the song-glyphs and the centroid of the cluster to which they belong. The color of the lines encodes the clusters.

Instead of using circles to represent the songs, musical instrument icons that encode the dominant musical instrument throughout the song were planned to be used. However, the implementation limitations which will be explained in Section 7.1.5 did not allow it. Hence circles are used to represent songs and the dominant instrument family through the song is encoded to the circle colors. The transparency of the circles was kept low to avoid over-plotting issues.

4.4.2 Performance Overview Plot

For the performance overview plot, an engaging and attractive design was needed. The features to visualize in this visualization were camera motion, musical instruments detected from images, and visual clutter. Since all of these features are time dependent, to propose an appealing and practical design, timeline design space was investigated.

To search the timeline design space in a structured manner the outline that was proposed by the work [17] was referenced. The paper proposes an outline that divides the design space for timelines into several representation types. In this thesis, all representation types defined in the mentioned paper were considered and the best option for the performance overview plot was chosen.

For this design, it is important to make the visualization more engaging and attractive. Hence grid, spiral, or arbitrary representations were considered as they are more appealing [17]. Each one of these representations has its advantages in different use cases. Grid and spiral layouts make periodic patterns more visible and arbitrary representations increase memorability. However, this work's scope is to create a music representation that will appeal to the general audience. In this case, it is important to use a simple and easily understandable layout. Grid, spiral, or arbitrary layouts might be visually overwhelming and too analytic for this work's target users and they might create difficulties in visually measuring distances.

Radial timeline representations are great for presenting periodic features and are appealing to the human eye. In the context of music visualization using a radial layout would align with the periodicity of the music. Also, the appearance of the radial graph resembles a phonograph record which is endearing for a music visualization aimed at a general audience.

In some cases, radial representations can be difficult to understand for the users. However, since this layer only serves as an overview and will be guided with a detail view plot, it is not considered to be a problem.

Figure 4.4 presents the performance overview visual design. Very simple glyphs (squares, circles, and triangles) were used to differentiate between different features (camera motion, musical instruments, and visual clutter). The location of the glyphs codes the time in which the feature occurs. Since the visual clutter is calculated for all the frames, the triangles look continuous, appearing almost like arrows. The size of the triangles encodes how cluttered the related frame is.

In radial layout, it is important to explain to users where the origin point of time is and in which direction the time flows. To help with this issue white beginning and ending points were added. Icons were used to express the features to make the overview graph more appealing.

The gap between the beginning and ending points encode the length of the song so that it is possible to visually compare the duration of different songs.

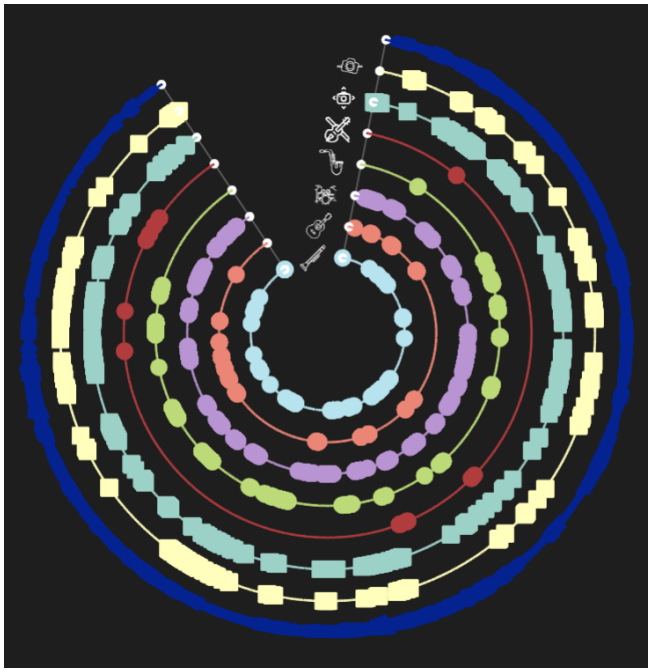


Figure 4.4: An Example Performance Overview Plot

4.4.3 Performance Detail Plot

To enable an individual analysis of all features in a comprehensive structure a performance detail plot was designed. For this visualization linear layout was chosen since they are clear and easy to understand.

The features to visualize in the performance detail plot consists of camera motion, musical instruments detected from video frames, visual clutter, waveform, tempogram, and power spectrogram. The performance detail visual design can be seen in Figure 4.5.

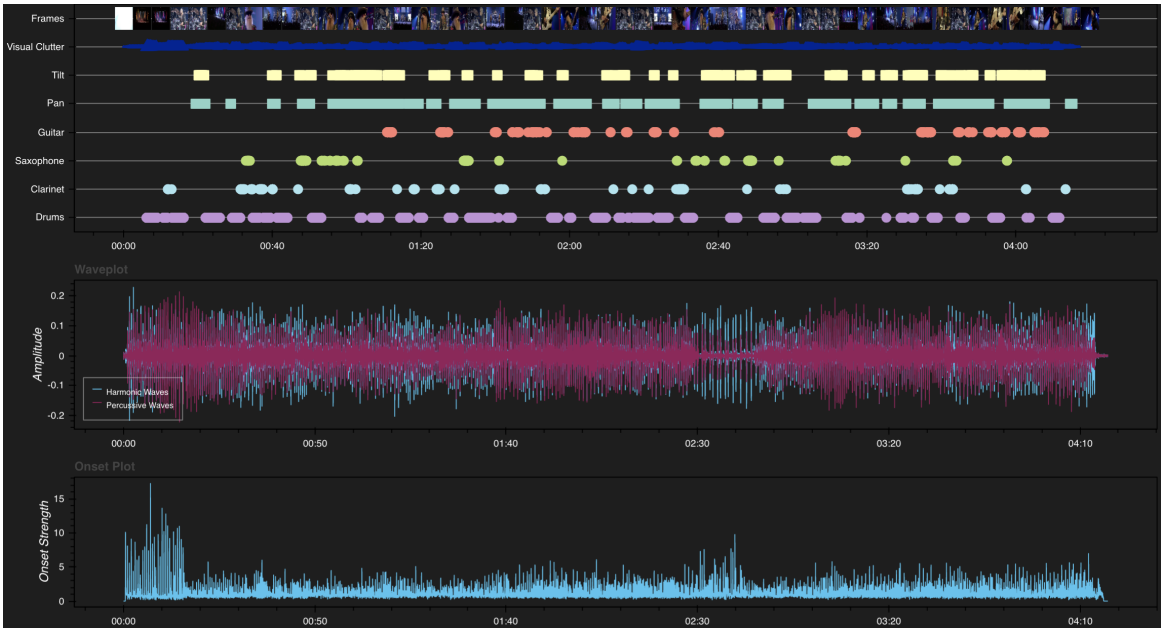


Figure 4.5: Performance Detail Plot

It can be observed that the top section of the performance detail view resembles the straightened-up version of the radial performance overview plot. Camera motion, musical instruments, and visual clutter are also visualized in the performance overview plot. To link the performance overview and detail plots, for these features the same glyph and color encoding is used. The x the position of the glyphs encodes the time when the feature has occurred. Instead of using icons, the names of the features were placed on the y axis to be more explanatory. The top row visualizes the frames over time.

For color coding, each feature is mapped to a color. A portion of the feature-color map can be seen in Figure 4.6. For the musical instruments, the instruments in the same family are represented by similar colors. For example, percussive instruments such as drums and tambourine are both encoded by shades of purple.

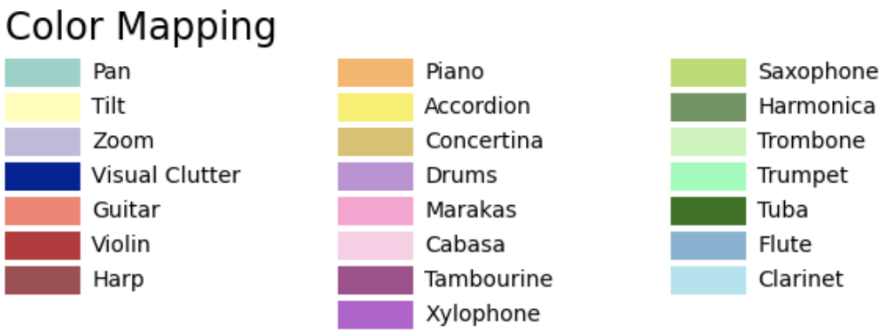


Figure 4.6: Color Map to Encode Features

For the waveplot and the tempogram the colors were selected to be bright and distinct to catch attention on the dark background. Again the x-axis encodes the time. The legend and axis labels were put to make the visualization understandable.

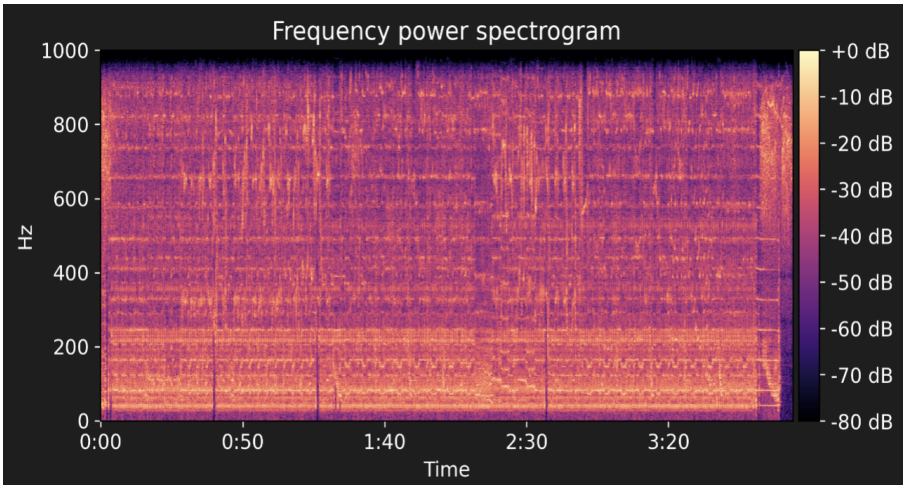


Figure 4.7: The visual design of the spectrogram

Power spectrogram visualization is commonly used in audio analysis. Therefore there exists a widely used visual design where the x-axis encodes the time and the y-axis encodes the frequencies. The strength of the frequencies is coded in color through time. The designed power spectrogram is shown in Figure 4.7. For color coding, a perceptually uniform color map was chosen. Since the application background is black, the lightest color was set to represent the highest number, because

the light colors attract more attention on dark backgrounds. A color bar was placed on the right with proper labeling to make the plot clear.

4.5 Web Application

This thesis proposes a high-dimensional data visualization tool in the form of a web application. The web application implementation bases itself on VIAN a film analysis and visualization web application [18]. In this work, VIAN was extended to analyze live performance videos and visualize them.

This section presents the final web application. The structure of this section is as follows:

- Section 4.5.1 explains the web application user flow.
- Section 4.5.2 introduces VIAN and presents its functionalities without this work's contribution.
- Section 4.5.3 introduces this work's contribution and explains how VIAN has been extended.

4.5.1 User Flow

In this thesis, VIAN is being extended. It had functionalities such as query, glossary lookup, corpus management, and film analysis page which are being preserved. The main change done to the user flow is highlighted in Figure 4.8.

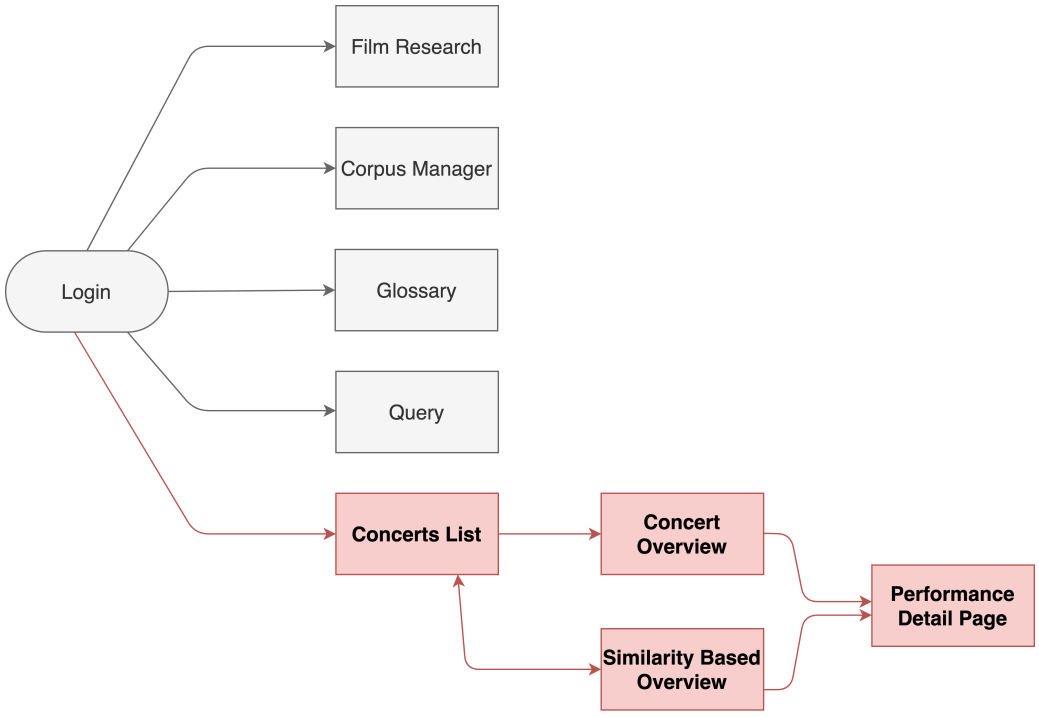


Figure 4.8: User flow diagram

Users can navigate through the existing pages via the sidebar menu shown in Figure 4.9. The "Projects" menu item was made expandable due to the addition of "MJF Concerts" navigation which leads users to the red path shown in Figure 4.8. Via the "MJF Concerts" menu item users can access the visualization tool that was implemented.

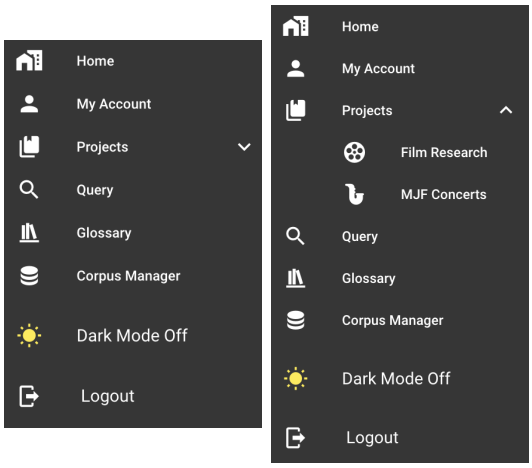


Figure 4.9: Sidebar menu of the application

The menu items "Film Research", "Query", "Glossary" and "Corpus Manager" navigates users to the basic functionalities of VIAN which will be explained in Section 4.5.2. The menu item "MJF Concerts" redirects users to the page where users can see all concerts in the database. This page will be explained in Section 4.5.3.1.

From the "Concerts List" page users might choose to go to the similarity based overview plot where all existing songs are compared or they can directly go to the concert overview page where they can see the songs performed during a specific concert. Selecting either way, users will be investigating a list of songs from different aspects. From that point on they might move on to the performance detail page to analyze the one selected song in detail and visualize the extracted features.

4.5.2 VIAN Basic Functionalities

This section introduces the pre-implemented functionalities of VIAN before this work's contribution.

- Section 4.5.2.1 introduces the login page.
- Section 4.5.2.2 explains the main functionality of VIAN which is film research visualizations.
- Section 4.5.2.3 introduces the search and filtering abilities of VIAN.
- Section 4.5.2.4 introduces glossary lookup page.
- Section 4.5.2.5 explains how to create corpora and manage them in VIAN.

4.5.2.1 Login

VIAN provides a standard login page that requires email and password inputs. Since the application is not accessible to public use, it is not possible to sign up automatically, only the administrator can create accounts. Keeping track of the users allows keeping track of private corpora, users might build their private corpora and manage them without sharing them publicly.

4.5.2.2 Film Research

The Film Research menu item on the sidebar navigates users to a page where all the movies in the database are listed. On the list, it is possible to see the title of the movie, the production year, and the origin country.

Clicking a movie from the list redirects users to the movie detail page. Here it is possible to see the metadata known about movies such as the genre, name of the director, or color consultant. The detail page also presents the segments of the movie along with their annotations and keywords. Finally, color visualizations are presented to perform analysis. It is possible to visualize chroma values along time throughout the movie, or it is possible to see color palette extraction from the movie. For the visualization, the color palette can be projected into the AB plane of CIELAB color space, or it can be projected into the LC plane of CIE LCh color space. It is also possible to create a color palette histogram. In summary, this page presents more information regarding the movie, the segments of it, and various color analysis visualizations [18].

4.5.2.3 Query

The query page allows users to search and filter the movies according to the tags, year of production, or corpus. In the case of filtering by year and corpus, it is possible to see the list of movies that fit the query, all their segments, screenshots, and their color visualizations combined into one. In the case of filtering by tags, the segments that are tagged with given keywords, the movies they belong and their screenshots are returned. Again it is possible to see their color visualizations combined altogether.

4.5.2.4 Glossary

Users can manually and automatically tag video segments using keywords. These keywords might refer to technical terms such as camera direction techniques, color palettes, lighting schemes, etc. To ease the use of the application for non-experienced users there exists a glossary. Users can search for a concept, read the description, and even see example segments that are tagged using a given concept.

4.5.2.5 Corpus Manager

VIAN allows users to create new corpora to analyze a group of movies in detail. When creating a corpus users can choose to make it publicly available or private. It is possible to add the movies that exist in the application database to the created corpus. Adding a new movie to the application is only possible using a script but not via the user interface.

4.5.3 VIAN Extended Functionalities

Montreux Jazz Festival analysis functionalities that are implemented as the main contribution of this thesis are explained in this section.

- Section 4.5.3.1 introduces the main page to list all the concerts.
- Section 4.5.3.2 introduces the similarity based comparison visualization for performance videos.
- Section 4.5.3.3 explains the overview page which presents songs from one selected concert.
- Section 4.5.3.4 introduces the detail page to present features extracted from performance videos.

4.5.3.1 Concerts List

After login, users can navigate to MJF Concerts using the sidebar menu to see a list of concerts in the database. As the Figure 4.10 shows, the list is presented using a grid-like structure. For each concert, an information card is created stating the concert date and concert hall. Using the search bar on top it is possible to search concerts according to the concert hall.

This page aims to present all of the data in the database and the first way of doing this is by listing all the concerts. By clicking the icons on the top left it is possible to change the view. The first icon refers to the concert cards and when selected the concerts list view is shown. The second icon refers to the similarity based overview plot. When selected the view changes to present all songs using a clustering plot as will be explained in Section 4.5.3.2. Overall this page presents all data either by listing all concerts or visualizing all songs in a clustering plot.

Another path a user can take from this view, instead of going to a similarity based overview, is by clicking the concert card. The concert cards navigate users to the concert overview pages which will be explained in Section 4.5.3.3.

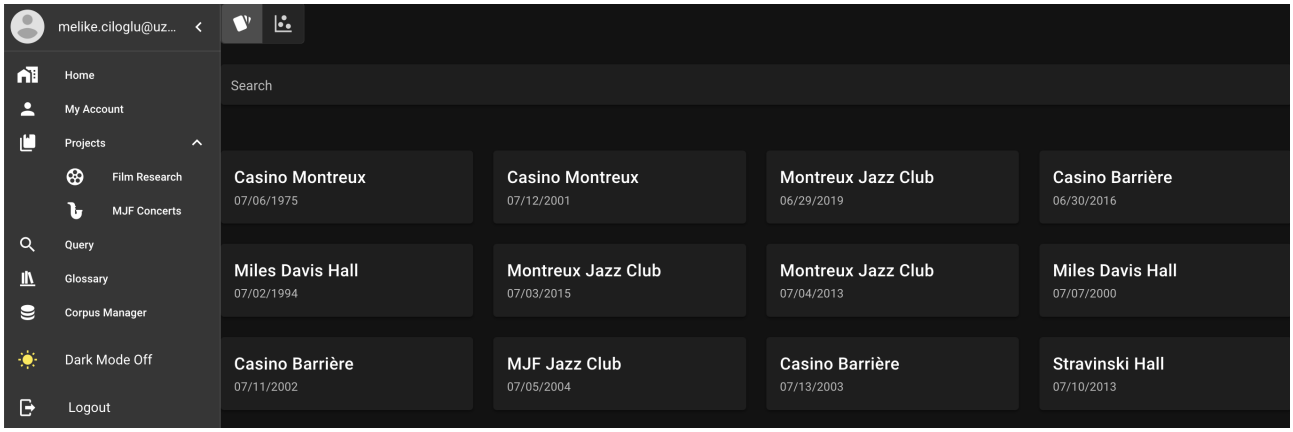


Figure 4.10: List of Concerts

4.5.3.2 Similarity Based Overview Page

Similarity based overview page aims to visually compare and cluster the songs and also individually visualize the features of them. It allows users to compare song videos with each other through a clustering visualization.

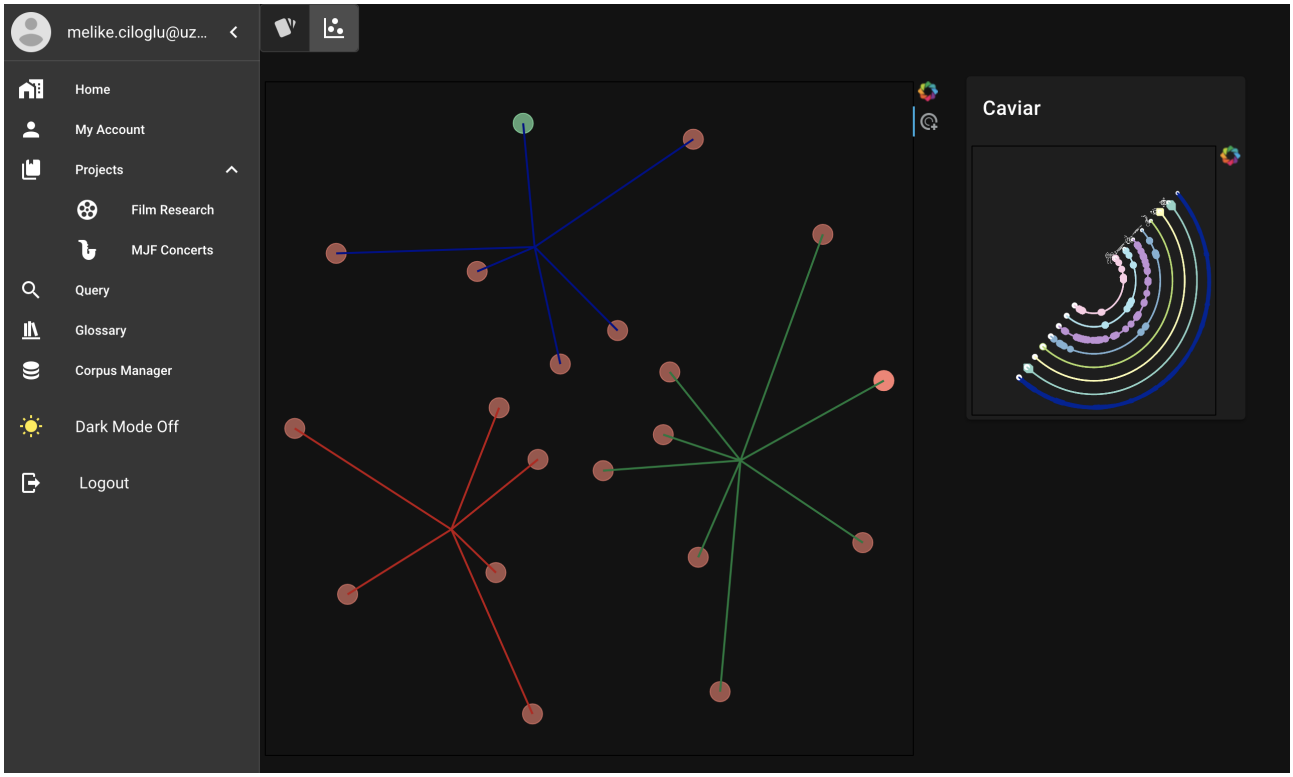


Figure 4.11: Similarity Based Overview Page

After pairwise song similarities are calculated multidimensional scaling is used to project the songs into two-dimensional space. K-means clustering is applied to form 3 clusters on the data. As shown in Figure 4.11 the songs were presented using a scatter plot. The color of the circles encodes the dominant musical instrument family detected from the audio. To visualize the clusters, lines between the circles and cluster centroids were drawn. The line colors encode the clusters. The scatter plot allows users

to select individual songs. When clicked on a circle, the overview plot for the song is shown on the right-hand side. Selected and unselected songs are distinguished using different transparency.

4.5.3.3 Concert Overview

The concert overview page is designed to compare the videos of the songs performed during a concert. For each song, a card is created and all the cards are visualized in a grid structure as shown in Figure 4.12. The song titles and the radial performance overview visualizations are presented for all songs. Since the number of songs performed during a concert is not too high it is possible to visually compare the features of the songs. The number of musical instruments played can be examined and compared. The songs' durations are also comparable via the gap between the ending and beginning points of the visualization.

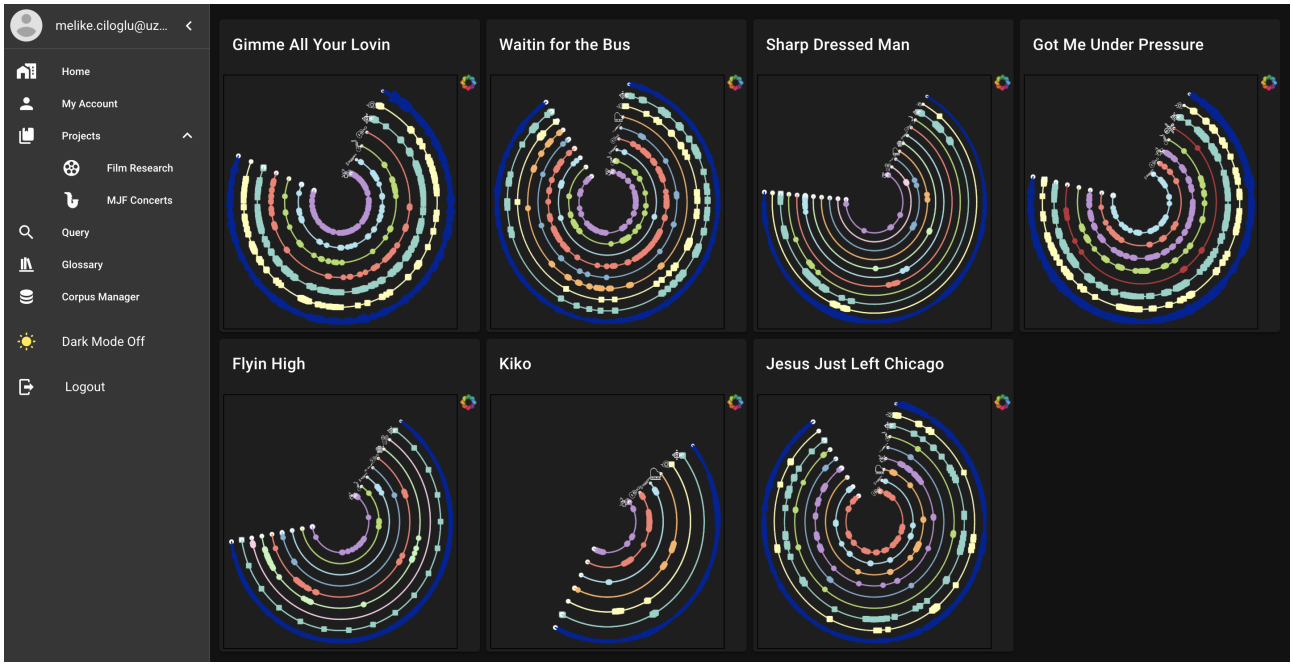


Figure 4.12: Concert Overview

From this page clicking a song's card would redirect users to the performance detail page for further investigation of the song.

4.5.3.4 Performance Detail Page

The performance detail page visualizes all the features and is designed to perform a detailed analysis of the recordings. As the Figure 4.13 shows the metadata regarding the songs is presented at the top along with the video frame.

The radial performance overview plot and the frequency power spectrogram are located in the next row. The power spectrogram provides deep insights into the audio structure and the performance overview plot presents visual features. Hence these two visualizations serve as a condensed analysis of the performance video as they communicate a significant amount of information even within a quick look. Then at the bottom, the performance detail plot in linear layout can be seen. The frames over time are visualized along with the visual features. Then the audio features, the waveform, and the tempogram are visualized.

Using the vertical slider next to the radial performance overview plot, a range of the video can be chosen to be visualized. Then the selected range will be emphasized using colors in the radial overview plot, and the linear detail view will visualize only the selected range. Hence the linear detail view will act as a zoomed-in visualization of a specific time range along the song. The video frame next to the metadata block is synchronized to visualize the first frame of the selected time range.

To analyze further, it is possible to select a subset of features to visualize in the linear performance detail view. Clicking glyphs on the radial overview plot will make the clicked feature disappear from the linear detail plot if it is visible, or it will make it appear if it is not visible. Using this intuitive clicking interaction users can choose which features to visualize in the detail view.

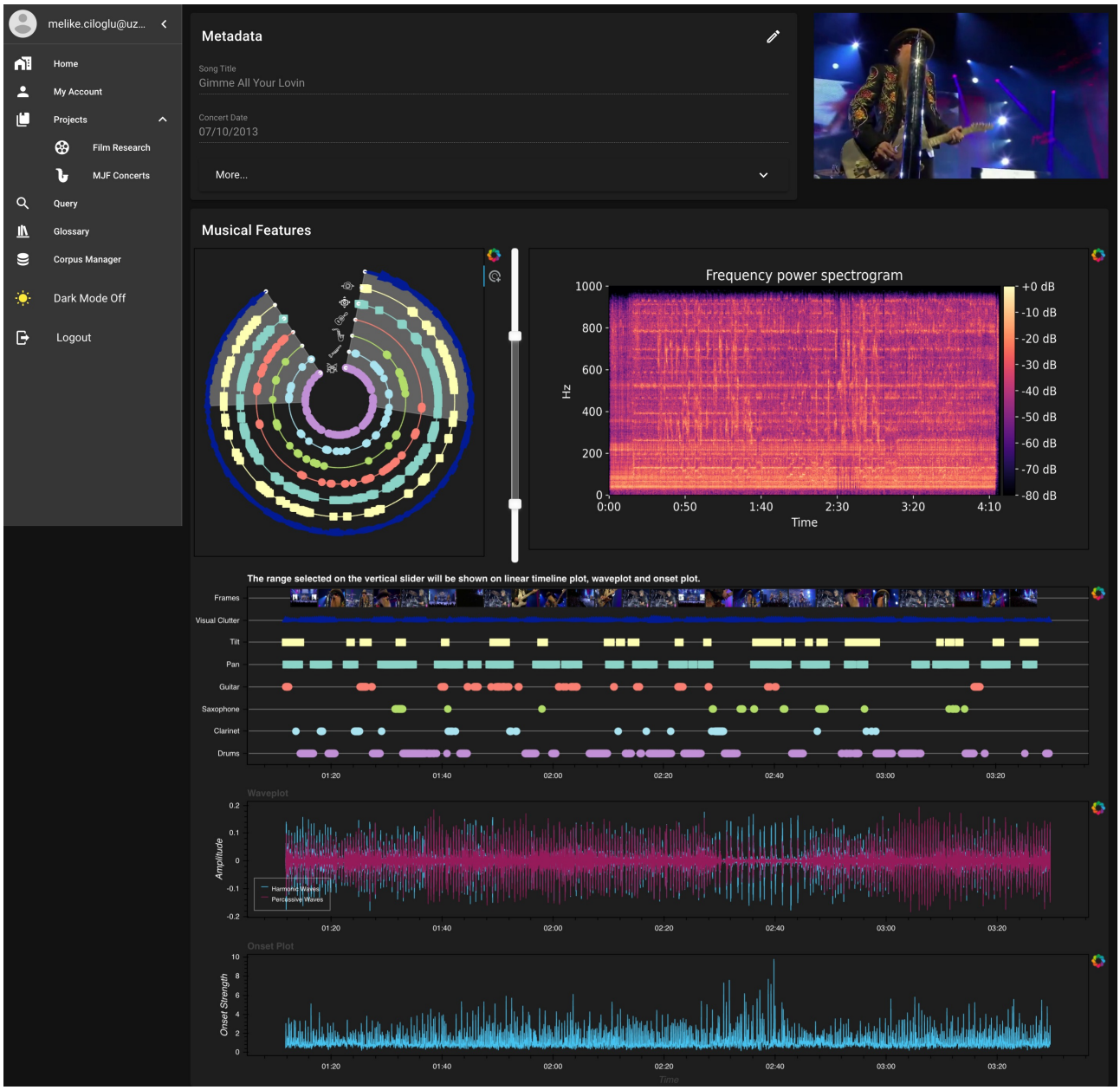


Figure 4.13: Performance Detail Page

5 Implementation

This chapter introduces all aspects related to the development of the solution explained previously. The outline of this chapter is as follows:

- Section 5.1 explains the implementation of the feature extraction phase.
- Section 5.2 presents the selected database and database management tool and explains the database structure in detail.
- Section 5.3 explains how the similarity metric is implemented.
- Section 5.4 introduces the libraries used during the implementation of the backend of the web application and explains the structure.
- Section 5.5 introduces the tools used for the frontend of the web application and explains the structure.

5.1 Feature Extraction

This section explains how the feature extraction phase was implemented. First, the libraries used were introduced, then the code structure was explained.

5.1.1 Feature Extraction Technological Decisions

This work focuses on exploring and using existing video analysis methodologies rather than discovering new ones. The implementation of the feature extraction step was done using Python on the Jupyter notebook. The following sections introduce the libraries or tools used for the feature extraction phase.

- Section 5.1.1.1 introduces OpenCV, a computer vision library both used for camera motion detection and musical instrument detection.
- Section 5.1.1.2 introduces a GitHub repository used for camera motion detection.
- Section 5.1.1.3 introduces a Python library for detecting visual clutter on images.
- Section 5.1.1.4 explains the neural network used for musical instrument detection from images.
- Section 5.1.1.5 introduces an extensive audio and music analysis library.
- Section 5.1.1.6 introduces an audio classification and analysis library.
- Section 5.1.1.7 introduces a Python-database adapter called Psycopg.

5.1.1.1 A Computer Vision Library: OpenCV

OpenCV is a computer vision library that is written using C but also has wrappers to be used with Python [19]. In this thesis, it was used for video processing. OpenCV can read mp4 files, extract frames from videos, and process images. Screenshots that are used in visualizations were taken using OpenCV. It was also used for camera motion and musical instrument detection.

5.1.1.2 Camera Motion Detection Repository

For camera motion extraction, the pre-implemented functions on a camera motion detection repository on GitHub [20] were used. These functions use OpenCV to extract optical flow from two consecutive frames of the video. Then it calculates the dominant flow angle and magnitude to finally decide whether the camera is moved or how it is moved.

5.1.1.3 Visual Clutter Library

To measure clutter a Python library was used [21]. This library defines visual clutter as a user interface metric and implements the feature congestion algorithm. It is possible to individually calculate color clutter, luminance contrast clutter, orientation clutter, or an overall clutter scalar considering all.

5.1.1.4 Convolutional Neural Network for Musical Instrument Detection

For the musical instrument detection from images, a Kaggle competition was referred [22]. The winner model of the competition that was trained to detect 30 different instruments was used. The model follows an EfficientNet convolutional neural network architecture. It was trained on 4793 images to identify 30 different musical instruments [22].

5.1.1.5 Music analysis library: Librosa

Librosa is an audio and music analysis library implemented for Python [23]. It provides various functions such as loading audio, extracting waveform, or for beat detection. Also provides a few audio visualization examples.

For the features related to audio processing like tempo, tempogram, spectrogram, and waveform detection librosa was used.

5.1.1.6 Audio description library: Essentia

Essentia is an open-source audio analysis and description library. It is implemented using C++ but also can run on Python and JavaScript [24]. It can classify sounds or audio features and provides a deep learning interface.

For dominant musical instruments, instrument family, mood, and genre detection Essentia was used.

5.1.1.7 Database Adapter: Psycopg

Psycopg is a popular PostgreSQL database adapter for Python [25]. It allows running queries on a database using Python. It is reliable, efficient and can run multiple concurrent inserts or update operations.

Psycopg is used in the feature extraction script to insert the extracted features into the database.

5.1.2 Feature Extraction Code Structure

The outcome of the feature extraction phase is a Python script that accepts a video path to an mp4 file as the input. The script directly inserts the extracted features into the database.

The script starts with camera motion detection, as pre-implemented functions from a GitHub repository is being used, there is not much freedom of choice on how to detect camera motions [20].

Next, the video is iterated from the beginning until the end. At each frame, the clutter scalar is calculated using the visual clutter library. Then the frame is fed to the convolutional neural network to detect the musical instruments. Hence one loop over the video results in both visual clutter and musical instruments features.

At this stage, all of the visual features are detected. Hence data is thresholded to eliminate noise and structured to be inserted into the database.

Afterward, the video is fed to the Librosa music analysis library and the audio is extracted from it. With the help of Librosa, tempo, onset information for tempogram, and power spectrogram is extracted from the audio. Finally, the audio is fed to the Essentia library and the mood, genre, and musical instrument detection function is run.

Having all the features extracted, the data is structured into dictionaries. A JSON string from the dictionary is formed and the string is encoded into binary format. Using the Psycopg library the binary data is inserted into the database.

5.2 Database

This section explains the tools used for the database implementation and describes the final structure.

5.2.1 Database Technological Decisions

To implement the database PostgreSQL relational database was used and the database server was monitored using the pgAdmin management tool.

- Section 5.2.1.1 introduces the PostgreSQL Database.
- Section 5.2.1.2 introduces the database management tool used for the implementation.

5.2.1.1 PostgreSQL Relational Database

PostgreSQL is an open-source relational database system [26]. The database is extensible, robust, and reliable. It uses SQL language and implements traditional database features and furthermore extends them. It runs on all major operating systems and is easy to manage using the pgAdmin management tool developed for PostgreSQL.

5.2.1.2 pgAdmin Database Management Tool

pgAdmin is a popular open-source management and development platform for the PostgreSQL database [27]. It helps organize and monitor multiple databases and provides a query editor with an auto-completion feature. It eases writing, running queries and viewing the results. In this thesis, pgAdmin was used to create and structure a PostgreSQL database.

5.2.2 Database Structure

For the data model and the database structure, it was intended to keep everything as simple as possible with minimal additions to the VIAN database.

Figure 5.1 shows the previous structure of the VIAN application database. For each project there exists a video that is stored in the db_movies table and movies might have segments, screenshots, and related analyses. It was intended to preserve the automated segmentation and segment, screenshot analyses that are implemented in the VIAN application. To achieve that live performance videos were uploaded into the db_movies database and all the relations between recordings and the other tables were preserved.

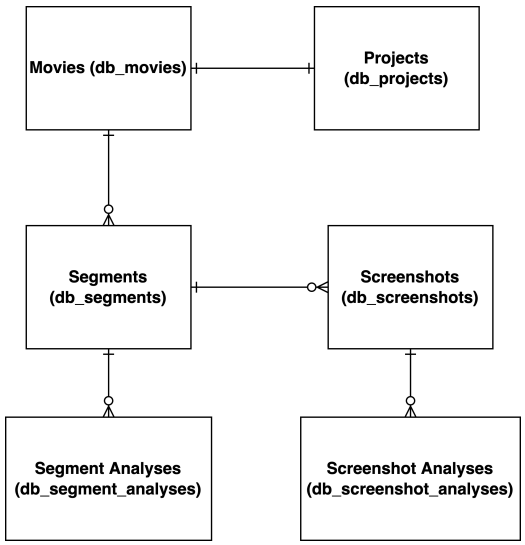


Figure 5.1: VIAN database modeling

Figure 5.2 presents the related tables in the final database implemented for the application. Two tables were added called db_concerts and db_mjf_analyses. For each concert, there might be zero to multiple videos where videos are stored in db_movies. Yet each video must belong to one single concert. Then for each video, again, there can be zero to multiple analyses performed on the video, and each analysis is strictly related to one video.

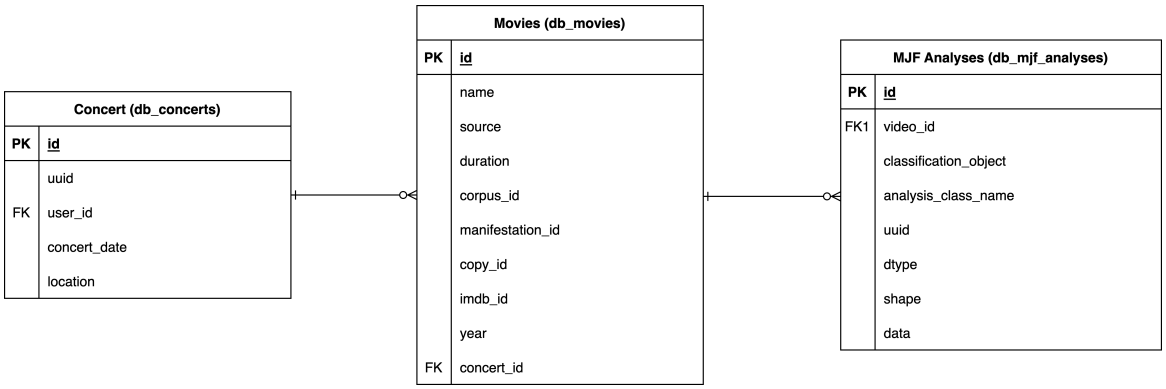


Figure 5.2: Database structure

The data model, as presented, makes minimal changes to the existing VIAN database and stores everything efficiently for the application.

The following sections introduce each of the tables in detail, explaining the columns and data types.

db_concerts

The concert table has a unique id and uuid columns to store the date and the location of the concert. Also, there exists a foreign key referring to the user table called user_id to keep track of the user who uploaded the concert data. The id column is set automatically when new records are being inserted using an incremental sequence.

The id and user_id columns' data type is an integer, while location and uuid accept characters and concert_date accepts DateTime data type.

db_movies

The movie table is mostly preserved so as not to disturb any process related to segmentation and analyses on application. However, one column was added called concert_id to connect the videos to the concert. For Montreux Jazz Festival videos this column contains a valid concert id, but for movies that are not related to this thesis, this column contains null values.

The duration, corpus_id, manifestation_id, copy_id, and year columns' data type is integer while imdb_id and source accept characters. All of these columns refer to the movie metadata which is not of the concern of this thesis. Columns related to this thesis are id, an integer automatically set using a sequence, name which consists of characters, and concert_id.

db_mjf_analyses

A table called db_mjf_analyses was needed to store the extracted features. This table contains a unique id, uuid, and a foreign key to be connected to the movies table. The columns classification_object and analysis_class_name help identify the analysis type applied to the video and consists of characters. Data is stored in binary format. Hence dtype and shape columns are required to decode back from binary format to numerical format.

5.3 Similarity Metric

This section explains how the similarity metric calculation was implemented. First, in Section 5.3.1 the libraries used were introduced, then in Section 5.3.2 the code structure was explained.

5.3.1 Similarity Metric Technological Decisions

The implementation of the similarity metric was done using Python on the Jupyter notebook. The following sections introduce the libraries or tools used for the feature extraction phase.

- Section 5.3.1.1 introduces the NumPy library.
- Section 5.3.1.2 introduces the Scikit-learn library.

5.3.1.1 NumPy

Pandas is a strong and open-source scientific computing library for Python. It was developed to handle multidimensional arrays and can run on GPUs for fast computing. It implements a comprehensive list of mathematical functions, including linear algebraic functions and random number generators.

During similarity metric implementation, it is used to handle the data, compute the inverse of the covariance matrix and handle matrix-vector multiplications. NumPy provides efficient functions to achieve the intended goals.

5.3.1.2 Scikit-learn

Pandas is an open-source Python predictive data analysis library. It provides various functionalities such as classification, regression, dimensionality reduction, and clustering. It is easy to use, efficient, has high performance and integrates well with other Python libraries.

After similarity metric calculation, it is used to calculate multidimensional scaling to project the data points into two-dimensional space and for K-means clustering. Scikit-learn implements both functions with high performance.

5.3.2 Similarity Metric Code Structure

The similarity metric phase is implemented as a Python script that reads data from the database. The script calculates the pairwise similarities between performance videos, applies multidimensional scaling and clustering, then inserts the calculated results back into the database.

First, the script uses Psycopg to collect all of the extracted features of the performance videos from the database. Next, the dominant musical instrument, instrument family, mood and genre classification data is separated. These feature vectors are the same size for all the performance videos. For each feature data matrices are formed where i^{th} row is the feature vector of the i^{th} performance video. Then using NumPy functions, covariance matrices of the feature data matrices are formed and the inverse of them are calculated. For the inverse calculation pseudo inverse function of the NumPy is used to avoid any possible errors caused by singular matrices. Finally, for each pair of videos, the difference between their feature vectors was calculated and they are multiplied with the inverse covariance matrix from the left, and their transpose is multiplied from the right. Taking the square root of the resulting number gives the pairwise similarities and allows forming a distance matrix where ij^{th} element is the similarity between i and j^{th} video.

Next, the visual clutter, musical instrument detection from video and camera motion detection features are considered. These feature vectors' size depends on the performance video length. Hence for each feature, first the size of the longest vector is determined and all other feature vectors are zero-padded to make them all the same length. Then for each feature, the previously explained methodology is repeated by first forming the feature data matrix, then calculating the inverse covariance matrix and finally calculating the distance matrix.

Having all the distance matrices for all of the features, they are combined into one final distance matrix by applying a weighted sum. Later, Scikit-learn's multidimensional scaling function is applied to the final distance matrix to position the performance videos in the two-dimensional Euclidean space. Finally, Scikit-learn's K-means clustering algorithm is run on the two-dimensional positions. The resulting positions and clusters are formed into one dictionary and inserted into the database as a JSON.

5.4 Web Application Backend

This section introduces the backend of the application. First, it explains libraries and their purposes in the backend, then introduces the structure.

5.4.1 Backend Technological Decisions

The most important libraries used for the backend of the application are being introduced in this section. Here is a list of the libraries which will be explained in detail.

- Section 5.4.1.1 introduces Flask web framework.
- Section 5.4.1.2 introduces SQLAlchemy which is an SQL toolkit for Python.

- Section 5.4.1.3 explains the Pandas library that is used for data manipulation.

5.4.1.1 Flask Framework

Flask is a Python web application framework, it can build scalable applications in a simple and readable way. It is lightweight and gives users flexibility thanks to extensions that can add different functionalities to the application.

This thesis uses Flask to build a backend API (Application Programming Interface). The API stands as a barrier between the frontend and the database. It reads required information from the database, applies data manipulation if needed, and redirects the outcome to frontend. For the data transmission to occur the frontend sends requests and the backend responds.

5.4.1.2 SQLAlchemy

SQLAlchemy is an open-source SQL toolkit written in Python. It allows provides users with all functionalities SQL has. It makes it possible to run queries on the database using Python.

It has an object-relational mapper which maps the tables of a database into Python classes. It maps columns of the tables into properties of Python objects and even keeps track of relationships between objects via foreign keys. With the help of it, it is possible to develop using an object-oriented model and have a clean, decoupled structure.

In this thesis, SQLAlchemy was used to connect the database to the backend and read information from the database when needed.

5.4.1.3 Pandas

Pandas is a strong, open-source Python data analysis library. It can handle multidimensional data fast, it is possible to apply data normalization and cleaning.

In this work, it is used to handle the data read from the database, and apply simple data manipulations to shape it for the visualization. Pandas provides efficient functions to achieve the intended goals.

5.4.2 Backend Code Structure

The data analysis is done during the feature extraction phase and the analyzed data is stored in the database. Hence the backend mainly serves as a connection between the frontend and the database structure and there is not much data analysis done here.

The database connection is explained in Section 5.4.2.1 and the API functionalities are explained in Section 5.4.2.2.

5.4.2.1 Database Connection

With the help of SQLAlchemy, the tables of a database were mapped into Python classes. database.py code file contains all mappings and transfers the data model implemented in the Postgres database into the Python environment. All the foreign and primary keys, the columns, and relationships between tables are mirrored in Python. Hence on the backend, concerts, videos, and users can be constructed as Python objects and their attributes can be accessed easily.

However, this setup does not carry all the data into the backend. It only creates the structure. Whenever the actual data values for the columns are needed, a query is run on the database. The mapping between the database tables and Python objects simplifies the syntax of the queries and makes it easier to track them compared to SQL queries.

A database session is maintained and a fixed connection URL is provided in the configuration file of the backend. The connection URL contains the name of the database server, port, administrator user name, and the password. If another database is needed to be used the connection URL could be simply changed and the application would not suffer any unwanted effects.

5.4.2.2 API

The application programming interface consists of 2 main files query and visualization.

Query API

The query file contains API functions that search and filter data from the database to construct the data model. For example, there exists a function returning a list of all concerts, or a function returning a list of songs belonging to a concert. Requests the data and metadata to fill in the Python objects created using SQLAlchemy are done in the query API.

Visualization API

The second API file is the visualization and as the name suggests the requests for the sake of visualizations are done here. To visualize the frames a function to provide the paths of existing frames is required and it can be found in the visualization API, or similarly, the paths of the icons can be requested here too. The request to get the extracted features from the database is located here and the also data is structured properly here to be directed to the frontend to be visualized.

5.5 Web Application Frontend

This section introduces the frontend of the application. It starts by explaining the libraries and continues by introducing the structure.

5.5.1 Frontend Technological Decisions

Here the libraries used for the frontend of the application are being introduced. The following is a list of the tools which will be explained in detail.

- Section 5.5.1.1 introduces NodeJS runtime environment.
- Section 5.5.1.2 introduces the Vue.js framework.
- Section 5.5.1.3 introduces the visualization library BokehJS.

5.5.1.1 NodeJS Runtime Environment

NodeJS is an open-source JavaScript runtime environment that can generate dynamic web pages. It uses an event-driven model to be lightweight and efficient. This means when an input-output operation is being performed, instead of blocking the thread and waiting for the response, NodeJS releases the CPU and resume the operations when the response comes back. It provides various libraries to simplify web application development. It allows running JavaScript both on the server-side in addition to the client-side.

In this thesis, NodeJS is used to implement and execute code locally.

5.5.1.2 Vue.js

Vue.js is a JavaScript framework for building user interfaces. It is based on standard HTML, CSS, and JavaScript, and eases component-based user interface development. It follows the model–view–viewmodel structure and is efficient, flexible, and adaptable. With Vue.js it is possible to extend existing HTML attributes called directives.

In this thesis, it was used to implement attractive user interfaces efficiently.

5.5.1.3 BokehJS

BokehJS is a library for creating interactive visualizations. It helps users build complex plots quickly and via simple commands. It is possible to embed Bokeh visualizations into web application frameworks. BokehJS allows implementing client-side interactions and in this thesis, it was used to implement all visualizations.

5.5.2 Frontend Code Structure

This application follows the model–view–viewmodel where the model is the backend, the view is the graphical user interface and the viewmodel is responsible for data manipulation and transportation between them. Each page has one view file in .vue file format, written mainly using HTML and Vue extensions. The views are completely independent of the model and any specific platform. For this application the model refers to the API explained in the previous section. For each view there exists a viewmodel, they make API calls and get the data returned by the model. Then they execute the plotting code written using BokehJS based on the data. Finally, they embed the created plot into view files HTML elements.

The implementation details of the plotting and interactions are explained in the following sections.

- Section 5.5.2.1 explains how the concerts list page presented in Figure 4.10 is implemented.
- Section 5.5.2.2 describes the implementation details of the similarity based overview page.
- Section 5.5.2.3 describes the implementation details of the concert overview page shown in Figure 4.12.
- Section 5.5.2.4 describes how the performance detail page in Figure 4.13 implemented.

5.5.2.1 Concerts List

This page presents a list of all concerts stored in the database in a grid structure as previously explained in Section 4.5.3.1. For the implementation, new Vue components called ConcertList and ConcertCard were created. ConcertList has a variable named mode and the mode can be controlled via two buttons placed on the top of the screen. If the mode is cards then ConcertList page presents ConcertCard. If not ConcertList presents the similarity based overview page. To implement the switch between the modes, the existing watch function of Vue.js is used.

ConcertCard extends v-card, the card component of Vue.js. As the title, the location of the concert is set and the subtitle is chosen to be the concert date. The grid was structured to contain at most four columns and is responsive to the page size. With the help of a viewmodel, a call to the backend was made to load the list of concerts.

A Search bar is implemented using a text input field combined with the watch function of Vue.js. When user enters a text into the input field the concerts with matching names are filtered according to the input and only related ConcertCards are shown.

5.5.2.2 Similarity Based Overview Page

As introduced in Section 4.5.3.2, similarity based overview page visualizes the song similarities and clusters in the Montreux Jazz dataset along with the individual performance video overviews.

The implementation of this page is integrated into the ConcertList Vue component that is explained in Section 5.5.2.1. If the button on top of the ConcertList page is clicked to show the similarity based scatter plot then the screen gets divided into two columns. For the scatter plot which will be placed in the left column, an API call is made to load the list of all songs, similarity, and clustering information. For the right column, a Vue component called SongCard was created extending the existing card component of Vue.js. SongCard presents the title of the song and an overview plot for it. Then BokehJS code is called to create the scatter plot and the radial performance overview plot. TapTool of BokehJS is used to give the ability to select data items from the scatter plot. When clicked on a circle a JavaScript callback function is fired to update the information displayed on the SongCard.

For the scatter plot, the line and the circle glyphs of BokehJS are used. The color of the circles represents the dominant musical instrument families for the given song. The SongCard implementation will be further explained in Section 5.5.2.3.

5.5.2.3 Concert Overview

The concert overview page, as explained in Section 4.5.3.3, presents the songs performed during one concert side by side to allow users to compare them. For the implementation, a Vue component called SongCard was used that extends the existing card component of Vue.js. An API call to the backend is being made to gather a list of all songs performed during the concert. After that, for each song, a SongCard is constructed and put together in a grid with at most four columns.

The title of the SongCard is the title of each song and for the description, a song overview plot is presented.

The song overview plot was implemented using BokehJS. The radial layout was achieved by mapping the time when the features occurred into polar coordinates using the sine and cosine functions in JavaScript's built-in Math module. The visualization was implemented using the basic glyph implementations of BokehJS and there are no interactions.

Icons to introduce the features are placed using imageUrl component of BokehJS and the size of the icons adjust according to the available space.

5.5.2.4 Performance Detail Page

The performance detail page provides a visual analysis tool for live-performed song videos as introduced in Section 4.5.3.4. For the implementation, two Vue components were created called SongDetailView and SongMetadata. SongMetadata is designed to present the concert and song related metadata and is contained inside of SongDetailView. It is placed on the top left corner of the page and presents the song title, concert date, location, etc. Right next to the SongMetadata section, the frames of the video are shown.

API calls are made to the backend to get the feature data vectors. Then the radial performance overview diagram is plotted using standard BokehJS glyphs. The overview diagram implementation is the same as in Section 5.5.2.3. Frequency power spectrogram visualization is placed next to the performance overview diagram. It is implemented using Librosa's display functions instead of BokehJS, hence it does not support any interactions.

Next linear performance detail view is implemented and placed under the overview plot and the spectrogram. For the implementation, the basic glyph implementations of BokehJS are used. The color encodings of the detail plot are the same as the overview plot. The video frames are also being

visualized using ImageUrl function of BokehJS on the top row. The waveplot and the onset plot follow the standard line plot implementation of BokehJS.

A vertical slider is placed between the radial overview plot and the power spectrogram. It is possible to select a time range from the video to visualize the features on the linear timeline plots. The slider triggers a JavaScript callback function. When the slider is used, the selected range is accentuated in the radial overview plot using the difference in background colors, and in the linear detail plots only the selected range is visualized. Also, the first frame of the selected range is shown in the upper right corner next to the metadata section.

As an additional interaction, users can choose which features to visualize in the linear detail plot. BokehJS TapTool is used to implement this functionality. When clicked on a glyph on the radial overview plot, the clicked feature disappears from the linear performance detail plot. Then again, if a hidden feature is clicked on the radial overview plot, it re-appears. This implementation is done using a JavaScript callback function for the TapTool.

6 Use Cases

In this section, some example use cases are described.

6.1 Dominant Instrument and Similarity

This use case would be to explore the effect of the dominant musical instrument on the similarity of songs. A user might want to see how similar the songs sharing the same dominant musical instrument families are. Another question the users might ask would be the effect of the dominant musical instrument on the data clusters. To find the answers to these questions, the similarity based overview page can be used. Users can see the colors of the song glyphs representing the dominant musical instrument families and compare the similarities between different songs via visual distance between glyphs.

6.2 Variations in Length and Musical Instruments

In this scenario, the users might want to compare the song lengths and musical instruments played during different songs in a concert. Users can question whether the musical instruments played vary during one concert. For this purpose, it is possible to use the concert overview page. On this page, all songs of a concert are listed with their performance overview plots. It is possible to compare the length of the songs and the instruments played during a concert.

6.3 Temporal Variations

A possible use case is when a user wants to explore temporal variation during a song such as tempo changes or solos. For this use case, it is possible to analyze the song using the performance detail page. The timeline visualization helps to see which musical instruments are playing at which parts of the song. The tempo changes are visible via the onset plot. The waveplot and the spectrogram help further analyze the audio. To investigate a time of focus, for example a range when a solo is suspected to be happening, the time range slider can be used. Via the slider, only a specific time range can be visualized which helps to see the details. Finally, the visualized video frames can also be used during the investigation.

7 Limitations and Challenges

To present a complete overview of the thesis in this chapter limitations of the end product and the challenges that are encountered during the work are introduced.

7.1 Caveats and limitations

This section presents the caveats and the limitations of the end product. The outline is as follows:

- Section 7.1.1 introduces the caveats of the musical instrument detection method from video frames.
- Section 7.1.2 discusses the feature extraction analysis processing time.
- Section 7.1.3 explains the limitations of customizability of the similarity metric.
- Section 7.1.4 discusses how the distance matrix can be updated on the new data.
- Section 7.1.5 explains why icons were not used on the similarity based scatter plot.

7.1.1 Musical Instrument Detection From Video Frames

Musical instrument detection from the video frames is achieved using an existing convolutional neural network that is trained to detect 30 different instruments. In machine learning, the generalizability of the models is an ongoing research problem. Generalization is the ability of a machine learning model to perform well on data that is not used for the training [28]. In this work, the model is being used on a completely different dataset. The used model was trained on individual images of the musical instruments [22]. Meanwhile, the musical instruments on the video frames are in a concert setting being played by musicians on a scene. The lighting of the scene and the camera angles add to the problem. The model's accuracy is expected to be significantly lower on video frames than it was on the reported test data.

In this thesis to eliminate the effect of this limitation, instead of detecting musical instruments only from the video frames, dominant musical instrument detection from audio is used as well. All the analyses are being presented together to the users so that the users can interact with the visualization and infer.

7.1.2 Analysis Processing Time

For feature extractions existing analysis libraries are being used, which limits the efficiency of the analysis phase. For example, for camera motion detection the used library iterates over all the video frames of the video and returns the detected motions. If custom implementation was done it would be possible to calculate the visual clutter and detect musical instruments over the same frames in parallel as the camera motions were being detected. However in the current implementation, since each outsourced method goes through the frames once multiple iterations over the video are required. This increases the analysis processing time and rules out the possibility of the analysis being made in

real-time. Since performance is not a priority for this thesis, the feature extraction phase is provided as a script to be run prior use of the web application.

7.1.3 Similarity Metric Weight Adjustability

The similarity metric calculation contains a weighted sum function over all the distance matrices calculated for each feature. The main reason behind using a weighted addition was to make it more interpretable and adjustable for different scenarios. According to the use case, some features might be more important than others and their roles in the distance were intended to be manipulated. However, the similarity metric is not openly adjustable for the users. It can only be adjusted inside the script by the web application administrator and after adjustments, it should be run again. This is a limitation to be considered for the users and can be improved in future work.

7.1.4 Distance Matrix Update on New Data

The Mahalanobis metric calculation which is the base of the proposed similarity metric depends on the data distribution and the covariance matrix. Hence when new performance videos are added to the dataset, it is not enough to only calculate pairwise similarities between the new video and the existing ones. All of the similarities between the existing videos should be re-calculated including the covariance values for the new video. This, overall, decreases the efficiency of the similarity metric. However, in this thesis' context, the Montreux Jazz Video Archive is selected to be analyzed and it is safe to assume that new videos will not be added frequently.

7.1.5 Icon Usage on Scatter Plot

On the similarity based overview visualization, it was intended to use musical instrument icons to represent the performance videos that encode the dominant musical instrument through the performed song. In this plot, TapTool of BokehJS is being used to implement selection by click and to visualize the performance overview plot of the selected song. For the icon implementation, ImageURL was planned to be used since it is the only image visualization construct of the BokehJS. However, it was realized that ImageURL glyphs can not be selected using TapTool. The implementation of the interaction was prioritized over the usage of icons and instead colored circles were used to represent the performance videos. The colors of the circles encode the dominant musical instrument family throughout the song.

7.2 Open Problems and Challenges

This section presents the challenges encountered during the work and still open problems. The outline is as follows:

- Section 7.2.1 discusses the lack of ground truth for features.
- Section 7.2.2 introduces the challenges related to the loading time of the visualization.
- Section 7.2.3 discusses the interactivity response time.

7.2.1 Lack of Ground Truth

For the feature extraction, state-of-the-art libraries and techniques were used. However, there is no quantitative way to measure the accuracy of the techniques on Montreux Jazz Festival data as there is no ground truth. The performance of the extracted features can not be measured.

7.2.2 Visualization Loading Time

For each visualization one or more API calls are done to the backend and data is fetched from the database. Afterward, BokehJS renders visualizations with the fetched data. The loading time of the visualization includes both the API calls and the rendering time. For each visualization, loading time varies between five to fifteen seconds depending on the number of visualization elements. In pages where multiple visualizations are placed side by side, for example in the concert overview page Figure 4.12, the overview plots appear one by one with five to fifteen seconds in between. In a real-case scenario, this limitation would affect users' reactions to the application severely.

7.2.3 Interactivity Response Time

The implemented visualization tool is interactive and allows users to select the time range or which features to visualize. For the implementation, the standard TapTool or the slider constructs of the BokehJS are used. To achieve the interaction after the user input a JavaScript callback function is triggered. The interactivity response time takes one to five seconds after the user clicks because of BokehJS rendering process. This response time is noticeable to the users and limits the smoothness of the interactions.

8 Conclusion

The main goal of this work is to explore the Montreux Jazz archive and explore and implement high-dimensional data analysis and visualization techniques. The end product provides a feature extraction script to analyze MP4 videos and extract visual and audio features, a similarity calculation script that implements a newly proposed similarity metric and calculates pairwise distances between all the videos, and a web application with interactive visualizations to help music enthusiasts explore the Montreux Jazz Festival video archive. Each video of the archive includes one song performed live by artists during the festival.

The web application is based on VIAN, a film analysis and visualization web application since it already provides various color and segmentation analyses. Using VIAN as a base eased the implementation process since building an app from scratch was unnecessary. The existing architecture was adopted and the functionalities of VIAN are preserved. However, it also increased the learning time since initially the code structure of VIAN needed to be learned. The existing database structure and the code were studied and multiple questions and answers sessions were needed with the VIAN developers.

The literature review conducted failed to find other examples of live concert video visualizations and the found existing tools mostly focus on either similarity based comparison or detailed song visualizations. Unlike the existing music visualization tools, this thesis provides both an overview to compare multiple videos and a detailed view to analyze the audio and video features. Nevertheless, this thesis also presents the existing limitations of the implemented solution and provides possible future work ideas.

The feature extraction phase of this thesis is limited by the currently implemented video and audio analysis technologies in Python. Custom analysis algorithms and machine learning models are not designed or implemented. Also, existing methods' accuracies on Montreux Jazz Festival data are not quantitatively measured. Hence as the research evolves, this work can always be improved using more current video analysis algorithms.

The work is implemented as multiple scripts and a web application. Due to the analysis time of the videos, it was not possible to integrate the feature extraction or similarity metric calculation phases into the web application. The analysis was not run on the complete dataset because of the time and computation power constraints. Instead, a small sample set of 20 videos was selected to be analyzed and visualized using the web application.

This master thesis provides the following contributions to the research field:

- Essential features that depend on video frames and audio to consider during the analysis of performance videos are listed. Existing feature extraction technologies are searched and the best options are implemented.
- A similarity metric to compare different performance videos is defined and implemented in Python. A script to calculate pairwise distances between all videos in the dataset is presented.
- Appealing, easy to understand, and engaging interactive visualizations are designed to present the extracted features to the general audience.

- A visual analysis tool has been developed that uses state-of-the-art technology. The tool allows the exploration of the videos by similarity and analysis of them in depth.
- Limitations of the implemented visual analysis tool and overview of known challenges are studied and further research opportunities are identified.

9 Future Work

Here is a list provided presenting possible improvements for the suggested solution or where further research is required.

- For the problem presented in Section 7.1.1, to improve the accuracy, transfer learning can be applied to the existing convolutional neural network. Example video frames from the Montreux Jazz video archive can be annotated and used for fine-tuning the pre-trained network to make it perform better on the determined video archive.
- To overcome the limitation explained in Section 7.2.1, a subset of the dataset can be manually annotated, for example to measure the accuracy of the musical instrument detection.
- For the limitation described in Section 7.1.2 instead of using the existing libraries, the features can be implemented from scratch. The code can be written using C instead of Python for faster execution and is designed to be run on GPUs. It can be integrated with the backend of the web application and then the backend can be deployed in a server with high processing power. This would significantly reduce the analysis time and even could make the analysis real-time.
- Regarding the limitation introduced in Section 7.1.3, the role of each feature on the similarity calculation can be presented to the users and a selection tool can be added to the frontend so that the users can choose which features should have greater weight in the final calculation.
- About the limitations described in Section 7.2.2 and 7.2.3, some research proposed that deploying the web application to a server and including BokehJS files as a static source to the server would help decrease the loading time. Also using a server with high processing power would decrease the API latency and help with this limitation as well.
- To decrease the information load on the users, the songs can be divided into pieces using audio features such as the tempogram or the spectrogram. Another possibility is using the auto segmentation functionality of VIAN. Created segments can be annotated automatically according to the musical instruments. Finally, visualizations can be adjusted to present the segmentation.
- Instead of presenting the video frames on the performance overview page (the top right corner on Figure 4.13) a video player can be placed so that also the audio can be played.
- It was intended to run the color analysis of the VIAN application. However, within the specified time frame it was not feasible. For future work, the addition of color visualizations to the performance visualizations can be considered.

Bibliography

[1] UNESCO, "The montreux jazz festival: Claude nob's legacy," Available at https://en.unesco.org/memoryoftheworld/registry/597 (28/07/2022).

[2] O. Hilliges, P. Holzer, R. Klüber, and A. Butz, "Audioradar: A metaphorical visualization for the navigation of large music collections," in *Smart Graphics*, A. Butz, B. Fisher, A. Krüger, and P. Olivier, Eds. Springer Berlin Heidelberg, 2006, pp. 82–92.

[3] R. Dias and M. J. Fonseca, "Muvis: An application for interactive exploration of large music collections," *ACM International Conference on Multimedia*, pp. 1043–1046, 2010. [Online]. Available: https://doi.org/10.1145/1873951.1874145

[4] C. Walshaw, "A visual exploration of melodic relationships within traditional music collections," *IEEE International Conference Information Visualization*, vol. 22, pp. 478–483, 2018.

[5] J. H. P. Ono, D. C. Correa, M. D. Ferreira, R. F. de Mello, and L. G. Nonato, "Similarity graph: Visual exploration of song collections," *IEEE Conference on Graphics, Patterns and Images*, 2015.

[6] H. Lima, C. Santos, and B. Meiguins, "Visualizing the semantics of music," *IEEE International Conference Information Visualisation*, pp. 352–357, 2019.

[7] N. Kosugi, "Misual: Music visualization based on acoustic data," *International Conference on Information Integration and Web-Based Applications & Services*, pp. 609–616, 2010.

[8] T. Bergstrom, K. Karahalios, and J. Hart, "Isochords: visualizing structure in music," *ACM Graphics Interface*, pp. 297–304, 01 2007.

[9] J. Snydal and M. Hearst, "Improviz: Visual explorations of jazz improvisations," *ACM Extended Abstracts on Human Factors in Computing Systems*, p. 1805–1808, 2005. [Online]. Available: https://doi.org/10.1145/1056808.1057027

[10] A. Soriano, F. Paulovich, L. G. Nonato, and M. C. F. Oliveira, "Visualization of music collections based on structural content similarity," *IEEE Conference on Graphics, Patterns and Images*, pp. 25–32, 2014.

[11] S. Zhang, Q. Huang, S. Jiang, W. Gao, and Q. Tian, "Affective visualization and retrieval for music video," *IEEE Transactions on Multimedia*, vol. 12, no. 6, pp. 510–522, 2010.

[12] P. Chiu, A. Girgensohn, and Q. Liu, "Stained-glass visualization for highly condensed video summaries," *IEEE International Conference on Multimedia and Expo*, vol. 3, pp. 2059 – 2062, 2004.

[13] J. Martinho and T. Chambel, "Colorsinmotion: Interactive visualization and exploration of video spaces," *International MindTrek Conference*, p. 190–197, 2009.

[14] R. D. Maesschalck, D. Jouan-Rimbaud, and D. Massart, "Tutorial: The mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, pp. 1–18, 2000.

[15] M. A. A. Cox and T. F. Cox, *Multidimensional Scaling.* Springer Berlin Heidelberg, 2008, pp. 315–347.

[16] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[17] M. Brehmer, B. Lee, B. Bach, N. H. Riche, and T. Munzner, "Timelines revisited: A design space and considerations for expressive storytelling," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[18] G. Halter, R. Ballester-Ripoll, B. Flueckiger, and R. Pajarola, "Vian: A visual annotation tool for film analysis," *Eurographics Conference on Visualization*, vol. 38, 2019.

[19] "Opencv," Available at https://opencv.org/ (03/08/2022).

[20] C. Cho, "Camera motion detection," Available at https://github.com/chuckcho/camera-motion-detection (26/07/2022).

[21] A. H. Kargaran, "Visual clutter," Available at https://github.com/kargaranamir/visual-clutter (02/08/2022).

[22] Kaggle, "30 musical instruments -image classification," Available at https://www.kaggle.com/datasets/gpiosenka/musical-instruments-image-classification (02/08/2022).

[23] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenbergk, and O. Nieto, "librosa: Audio and music signal analysis in python," *Python in Science*, vol. 14, pp. 18–25, 2015.

[24] "Essentia," Available at https://essentia.upf.edu/ (02/08/2022).

[25] "Psycopg – postgresql database adapter for python," Available at https://www.psycopg.org/docs/ (02/08/2022).

[26] "Postgresql," Available at https://www.postgresql.org/ (03/08/2022).

[27] "pgadmin," Available at https://www.pgadmin.org/ (03/08/2022).

[28] Y. Chung, P. J. Haas, E. Upfal, and T. Kraska, "Unknown examples & machine learning model generalization," *ArXiv*, 2018.