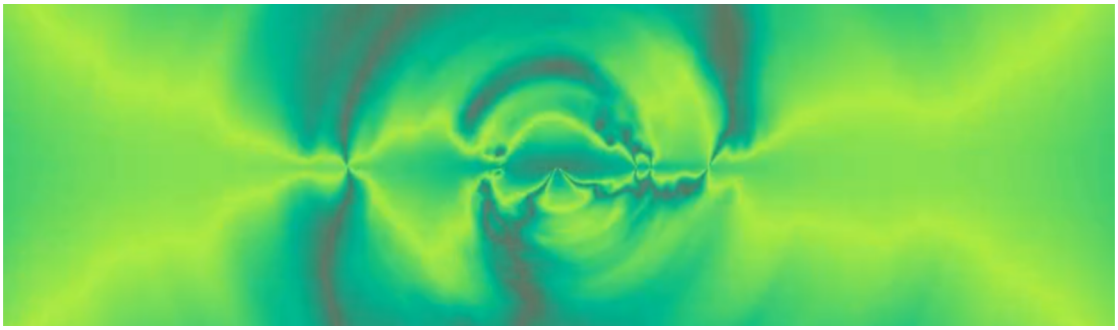
TÉCNICO LISBOA



From Music to Animations

A Computational Creativity Approach

Isabel Rego Raposo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Prof. Nuno Manuel Robalo Correia

Examination Committee

Chairperson: Prof. Manuel Fernando Cabido Peres Lopes
Supervisor: Prof. Helena Sofia Andrade Nunes Pereira Pinto
Member of the Committee: Prof. Amílcar Cardoso

October 2021

Abstract

Music visualization expands the musical sensations to the visual realm. If the audio and visual domains align, the whole experience is enhanced. We present a system that creates two-dimensional abstract animations based on audio files, using a Computational Creativity approach. We start by segmenting the audio, such that each audio section generates its own animation. The animations are created from audio-dependent mathematical functions, and colormaps translate the output of the functions into pixel colors. The animation functions are obtained by randomly assembling elementary functions and variables into an expression tree. The variables depend on the frequency intensities of the audio, its estimated tempo, and the pixel coordinates. We developed three animation analysis methods to select the final colormaps and functions from a pool of randomly generated ones. Mood Matching looks for audio-animation matches by evaluating their mood. The Preference Model emulates the aesthetic preference of the user, after training on classifications of audio-animation pairs, for which we obtained a validation precision of 84.6%. Manual Selection corresponds to a co-creation method, where the user chooses the final colormaps and functions from a set of high fitness samples. The methods were evaluated through online forms, revealing that participants deemed the videos creative, and the visuals were considered to align well with the audio. The Preference Model outperformed Mood Matching, and Manual Selection exhibited potential for its use as a co-creation tool to generate music visualizations. We believe our results further motivate the exploration of Computational Creativity systems based on human preference modeling.

Keywords

Computational creativity; Music visualization; Computational aesthetic measures; Human preference modeling.

Resumo

A visualização musical expande sensações musicais para a esfera visual. Se os domínios áudio e visual estiverem alinhados, a experiência fica aprimorada. Apresentamos um sistema que cria animações abstratas bidimensionais baseadas em ficheiros áudio, usando uma abordagem de Criatividade Computacional. Começamos por segmentar o áudio, tal que cada secção de áudio gera uma animação. Estas são criadas a partir de expressões matemáticas dependentes do áudio, cujos resultados são traduzidos para cores através de esquemas de cores. As funções são obtidas através da organização aleatória em árvore de funções elementares e variáveis. As variáveis dependem das intensidades das frequências e do andamento do áudio, e das coordenadas dos píxeis. Desenvolvemos três métodos para selecionar os esquemas de cores e funções, dado um conjunto gerado aleatoriamente. O primeiro método procura correspondências entre áudio e animação através do humor. O segundo método aprende a preferência estética do utilizador com base em classificações prévias de pares áudio-animação, tendo obtido uma precisão de 84.6%. O terceiro método corresponde a um processo de co-criação, onde o utilizador escolhe os esquemas de cores e funções dado um conjunto de amostras consideradas adequadas. Os métodos foram avaliados utilizando formulários online. Os participantes consideraram os vídeos criativos, e que os visuais alinhavam com o áudio. O modelo de preferência superou o método baseado em humor, e a seleção manual demontrou potencial para ser usada como ferramenta de co-criação para gerar visualizações musicais. Os resultados encorajam a continuação do estudo de sistemas de Criatividade Computacional baseados na modelação de preferência.

Palavras Chave

Criatividade computacional; Visualização musical; Métricas computacionais de estética; Modelação de preferência humana.

Contents

List of Figures

List of Tables

Acronyms

CC	Computational Creativity
CNN	Convolutional Neural Network
CPPNs	Compositional Pattern Producing Networks
DT	Decision Tree
DFT	Discrete Fourier Transform
DTFT	Discrete-time Fourier Transform
FFT	Fast Fourier Transform
GANs	Generative Adversarial Networks
GMM	Gaussian Mixture Model
HSL	Hue, Saturation, Lightness
KNN	K-Nearest Neighbors
LR	Linear Regression
MFCC	Mel Frequency Cepstral Coeficients
MLP	Multilayer Perceptron
PCA	Principal Component Analysis
RGB	Red, Green, Blue
RMSE	Root Mean Square Error
STFT	Short-time Fourier transform
SVM	Support Vector Machines
SVR	Support Vector Regression

Chapter 1

Introduction

Being creative is arguably one of the most valued skills in modern society. Although scholars have questioned whether there will ever be a consensus regarding the definition of creativity [1], it is nevertheless a topic of intense study in numerous fields that range from psychology to computer science. A consensus is even harder to achieve regarding whether machines can be creative, yet the study of creativity through a computational lens can generate useful tools and insights relating to the creative process, which can, in turn, facilitate the production of new and valuable artifacts. Computational Creativity is the field that explores such systems.

1.1 Motivation

The technological advancements of modern society have allowed us to listen to songs on demand, and provide a variety of mediums that enhance the auditory experience with the aid of the other senses. Music videos are short videos that complement a song or album with visuals. When the perceived visual mood matches that of the audio, the emotional impact of both the audio and the visual experience can be reinforced [2].

To create a music video, a person is usually required to manually assemble a video stream that visually matches the music in question in some regard. This task can be quite time-consuming and requires knowledge of both animation generation (in case artificially generated videos are used instead of video recordings) and video manipulation [3]. In addition, the desired aesthetics can greatly vary depending on the song, and even between producers. With a tool that facilitates video generation based on audio, this task can become less tedious and demanding, and to achieve this with a system that takes the person's taste into account would be even more useful.

In today's personal computer and smartphone era, music visualization tools are widespread across platforms[1], and can range from a set of pre-existing visuals to visualizations generated by the combination of certain effects. The goal of music visualization is to display animated imagery that changes

[1] Compilations of music visualization tools: https://en.wikipedia.org/wiki/Music_visualization, and https://github.com/willianjusten/awesome-audio-visualization

according to the audio's progression, and should be highly correlated to music features over time. Features frequently used as input to animate the visuals are the audio amplitude and frequency spectrum.

Typical music visualizations created by such tools are distinct from conventional music videos, as they are often real-time generated, and some tools can even display distinct visuals every time the program runs. Compared to music video creation tools, music visualization systems lack flexibility in terms of user manipulation of outputs, and are less likely to generate visuals that adequately represent high-level audio features.

Some concerns arise from the challenging task of creating visuals to accompany audio. On one hand, the animation should be related to the audio in some form and, on the other hand, the animation should be to the liking of the human user. A system that achieves these goals would combine the advantages of music visualization and music video generation tools, minimizing the disadvantages. For such a system to generate novel and valuable results requires some level of creativity. This work explores possible solutions to the problem of animation generation from audio, motivated by the above-mentioned concerns, through a Computational Creativity approach.

The term visualization often refers to the usage of visuals to communicate a concrete message. Our work does not approach visualization from this perspective, as we do not present a one-to-one mapping between the audio and what is displayed. This would be the case if, for instance, the musical notes had a direct visual correspondence. Since our work entails creativity, we looked for an unpredictable representation process. Therefore, we refer to visualization in its more broad definition, that is, as an expansion into the visual domain of audio data.

1.2 Problem Definition

In this thesis, we sought to develop a mechanism for generating abstract animations based on provided audio artifacts. The animations should have the following properties:

- the visual features should be aligned with the audio features in some manner;
- the animations should be considered creative;
- the animations should be enjoyable.

Since we aimed for a system that generates artifacts that could be considered creative, our work is inserted in the field of Computational Creativity. We drew inspiration from such systems to design methods of animation generation and analysis. In addition, we studied automatic music video generation tools and music visualization systems to inform our method of audio-video mapping.

Over the progress of this work, the system suffered continuous evaluation and adaptation in response to empirical observations. As a result, different animation analysis methods were conceived. To evaluate

if the methods created videos with the desired animation properties stated above, we used online forms to collect feedback from volunteers.

1.3 Contributions

Inspired by other systems that generate visual imagery with a Computational Creativity approach, we developed a system that creates abstract videos based on audio input. Our system creates original abstract audio visualizations based on heatmaps of mathematical functions. The functions are animated according to the frequency intensities of the audio and its estimated tempo. Videos created in the context of this work can be found in https://vimeo.com/frommusic2animations.

The audio is analyzed to find novelty peaks, which determine the segmentation of the audio, such that different audio sections generate different animations. In our solution, the produced animations have the same duration as the audio they were based on. In an attempt to visually represent some of the repetitive structure in the audio, the same colors are used for parts of the audio with similar mood, and the same drawing function can be used for parts of the audio with similar frequency distributions. For both colormap selection and animation function selection, a collection of random samples is first created, and only then one of them is selected, using one of the animation analysis methods developed in this work.

Our first method of automatic animation analysis is based on a cross-domain mapping between audio and animation features through mood matching. This method uses the predicted arousal and valence values of each audio section, estimates the valence and arousal of the colormaps and animations, and ranks them using empirical rules, and finally selects the top-ranked ones.

To take into account the aesthetic preference of the user, the preference method of automatic animation analysis was developed, which can be trained to learn what animations the user is more likely to prefer given an audio clip. This model uses audio features and the aesthetic features extracted from the frames to estimate the fitness of each audio-animation pair.

A co-creation tool to generate videos to accompany the audio was developed based on the previous animation analysis methods. The colormaps and functions are filtered before they are shown to the user, which then has the freedom to select the ones to be used in the final output. This corresponds to the manual animation analysis method.

Our methods were analyzed through online forms to gather feedback from volunteers on the performance of generated videos. Five songs from different genres were selected for this purpose, and one video was created for each song-method combination. Overall, the system's creations were considered music videos and abstract in nature, often described as colorful and psychedelic. The participants tended to consider the results creative, even if the visuals did not align with the audio, and even when

the results were not particularly liked. Our results show that the videos that were preferred by the participants were also the videos that they deemed had a better alignment with the audio. On average, the control videos were the least preferred, followed by videos obtained from the mood matching method, then using the preference model, and the video created through manual selection of animations was usually the preferred one. The subjects' receptivity to the videos also varied with the audio used.

Our analysis demonstrates the potential of creating animations to accompany audio using a Computational Creativity approach, showing promising performance when the automatic preference model is deployed, which can be further boosted using the co-creation method.

As we do not aim to present an optimal solution, we hope our work informs on how one can go about generating animations from musical input which can be considered creative, and further inspire discussion around computational creativity and the creative method itself. In particular, our results encourage further study of human preference modeling, for its potential to improve the automation and usability of co-creation tools.

1.4 Document Outline

We now briefly describe the structure of the remainder of this document. In Chapter 2, we review related work in terms of Computation Creativity systems, systems that generate abstract imagery, and systems that analyze the correspondence between audio and visual content.

We introduce our approach to solve the problem of translation of audio into video in Chapter 3. The next chapters detail the methods developed for different challenges. In particular, Chapter 4 describes the audio analysis process, Chapter 5 details our approach to animation generation, and Chapter 6 specifies the different mechanisms developed for animation analysis.

The evaluation method and the obtained results can be found in Chapter 7. Finally, we conclude the document by highlighting its main conclusions and future work in Chapter 8.

Chapter 2

Related Work

In the following chapter, we explore previous works that informed and motivated the development of our approach. In particular, Section 2.1 presents creativity-related concepts and Computational Creativity, while Section 2.2 introduces audio and visual media. Section 2.3 highlights systems that tackle the mapping between the audio and visual dimensions. Section 2.4 explores works with relevant techniques to create visual media, and Section 2.5 provides an overview of the studies that motivated our solution. Some useful definitions, models, and techniques mentioned throughout this work can be consulted in Appendix A.

2.1 Computational Creativity

2.1.1 Creativity

As our goal was to develop a system that had the potential to be perceived as creative, we must first clarify what we mean by creativity. Even though the collective understanding of the term has evolved over the years, it is still common to see different definitions by different authors. In this work, we will talk about creativity as it was defined by Boden [4] - the ability to create **ideas** or **artifacts** that are **new**, **surprising**, and **valuable**. Although the simplicity of this definition is appealing, it begs further explanation regarding the terms idea, artifact, new, surprising, and valuable.

The distinction between ideas and artifacts merely serves the purpose of clarifying that creativity can refer to the generation of more conceptual creations, such as poems, jokes, and scientific theories, as well as the generation of objects such as paintings, origami, and vacuum cleaners. With regards to newness, there can be two types of creativity: psychological creativity (P-creativity) refers to the generation of something new to the person that came up with it, whereas historical creativity (H-creativity), which also encompasses P-creativity, requires the creation to have never occurred in history before. For something to be considered surprising, one of the following must be the case: the occurrence of an event is unlikely; it wasn't obvious that an idea fits into a style of thinking; or something is surprising because it seems impossible that anyone could have come up with it, but they did. The valuable aspect of some-

thing has to do with having qualities worthy of respect, admiration, or esteem, or to be of considerable use or importance [5]. Because what gets to be called valuable depends highly on the context of the situation, time period, culture, and even on the person evaluating the creation, it is common to disagree on the creative status of something due to disagreements regarding its value.

This view on creativity is focused on the creative value of ideas or things, but many would argue that, even if computers can generate new, valuable, and surprising creations, they couldn't possibly be considered creative themselves, for a number of reasons. Some argue that creativity originates from the programmer's mind, not the program itself. Others use the computer's lack of consciousness, desires, and value system as the justification for such statements. Some state that creativity is uniquely a feature of the human mind. Because this discussion is philosophical in nature, in this work we take an agnostic stance on whether computers can be creative themselves, and focus instead on the creativity level of their output. We now investigate processes that may lead to the generation of creative outputs.

2.1.2 The Creative Process

A conceptual space is the collection of quality possibilities that characterize a concept. For instance, a style of painting has its own conceptual space. According to Boden [4], there are three forms of creativity: taking familiar ideas and combining them in an unfamiliar way; exploring a conceptual space; or transforming a conceptual space.

Koestler [6] characterizes the creative process as the bisociation of matrices, i.e., the integration of two or more previously unconnected frames of thought. The author exemplifies this process for different categories of creative activities: humor arises by replacing one matrix with an unexpected one to create punchlines; scientific discovery creates knowledge by fusing matrices together; and art can be interpreted as the juxtaposition of matrices. The creative process is then one of abstraction, comparison, and finding previously unseen links. Koestler states that conscious work towards a goal precedes most creative insights, although the insights themselves often occur while no effort is being directed towards the creative task.

Sawyer [7] describes the stages of the creative process as follows: the preparation phase involves collecting data, considering related works, and discussion; the incubation phase entails an internal organization of ideas; insight is the moment of realization (the *eureka* moment); and verification is the evaluation and elaboration of the final work, based on the insight. Sawyer also draws attention to the socio-cultural components of creativity, emphasizing the aspects of creativity that go beyond an algorithmic approach.

2.1.3 Computational Creativity

Given the above-mentioned discussion, some descriptions of creative methods appear to be implementable through programs, such as the exploration of a conceptual space, or the bisociation of matrices. Other forms of creativity, such as transforming a conceptual space, are of increased difficulty for a program, like redefining what is considered a painting [4]. The study of modeling, simulating, and replicating creativity through computational systems has become its own interdisciplinary field, the field of Computational Creativity (CC)[1]. This study often has one of the following goals: design systems that exhibit some level of creativity; study human creativity through computational models of it; construct programs that boost creativity in humans.

2.2 Audio and Visual Media

2.2.1 Sound

Sound, at its core, is a wave motion in elastic media. More specifically, it can correspond to an oscillation in pressure or stress, propagated in a medium through internal forces, or the combination of oscillations of this kind [8]. Besides its physical definition, sound is also the term used to describe the auditory sensation evoked by such oscillations. Sound that is represented in a digital form is called digital audio, which is just the representation of the audio signal through the storage of numerical samples from the continuous sound wave. The mp3 file format is an example of a method for storing digital audio.

2.2.2 Video

A video is an electronic display or storage of moving visual images, which can be combined with audio in a multimedia format such as mp4. A video is made up of a sequence of frames (digital images) that are displayed for a very brief moment before the next frame is shown, so that the appearance of movement arises from a fast transition between frames according to a frame rate. Each frame, in turn, stores information regarding the colors to show in each point in the frame (the pixels).

2.2.3 Color

In terms of physical phenomenons, a color is a combination of pure spectral colors, and a pure spectral color is, in turn, a light wave with a single wavelength. Color vision is then enabled by the perception of differences in light consisting of various wavelengths. Colors can be represented through several color models. A commonly used color model is the Red, Green, Blue (RGB), an additive color model, where

[1] http://computationalcreativity.net/iccc21/

each of its primary colors, red, green, and blue, are usually associated with an integer value between 0 and 255 (larger integer intervals are available for high-end digital image equipment). The [0,255] range is often represented in hexadecimal format, or using fractional values between 0 and 1. The Hue, Saturation, Lightness (HSL) color model was designed to better approximate the human perception of color attributes. The Hue dimension, usually represented as an angle or an equivalent representation, is based on the visual sensation that allows us to distinguish the colors red, yellow, green, blue, and their combinations. The Saturation dimension is inversely related to how gray the color is perceived, while the Lightness dimension can be seen as the color's proximity to white and distance to black.

2.3 Audio and Visual Correspondence

A considerable amount of software has been developed to match music and visual content. While most music visualization tools only require frequency distributions and intensity features from the audio, other systems rely on additional information to create the final result.

Some systems tackle the problem of music and video matching to create music videos using a set of available footage. Hua et al. [9] created music videos from personal home videos by matching the aural structure and repetitive patterns between the music and the videos. Yoon et al. [3] described a method of multi-level segmentation to increase the likelihood of having similar length audio and video sections, and analyzed video through velocity and brightness features. Nakano et al. [10] created music videos by reusing available dance videos from the web that best matched the beat of the music. Wang et al. [11] trained a framework to map music to emotion and video to emotion from annotated datasets, and used this mapping to look for the best matches between music and videos.

Some works looked for matches between music and images. Cai et al. [12] used lyrics files to fetch web images and create videos through slideshows. Wu et al. [13] mapped images and music segments to the respective feature spaces, performed clustering in each space, and learned relations between different domain clusters.

From the studied systems, two frequent problems to consider are: when to transition from one movie clip to the next, and what audio features may be important for comparing audios and videos. We now briefly describe two of the works that greatly inspired the way we process audio in our approach, based on these two concerns.

2.3.1 Creating music videos using automatic media analysis

Foote et al. [14] developed a method for the automatic creation of music videos, by extracting clips from home videos and aligning them with the desired song. First, the song is provided, along with the home videos to be used. The audio is analyzed in terms of novelty score, which is measured based on the

analysis of cross-similarity in the audio, and peaks in novelty are detected. The video is examined for suitability, and parts with excessive camera motion or poor exposure are discarded. To align audio with video, the resulting video boundaries are counted, and the largest novelty peaks are iteratively selected to be audio boundaries, until there are the same amount of video clips and audio sections. The length of each clip is adjusted to fit the audio section, by choosing the portions with the lowest unsuitability rates.

2.3.2 Automatic mood detection and tracking of music audio signals

Lu et al. [15] developed a hierarchical model to classify audio clips according to Thayer's model of mood [16]. The extraction of mood is based on intensity, timbre, and rhythm features, and each one is analyzed in terms of mean and standard deviation over a time window (except the rhythm features, which concern the entire audio), and those properties are used by the model. The audio is classified into four mood classes, depending on its arousal (energy/intensity) and valence (pleasantness, opposite of stress): Contentment, Depression, Exuberance, or Anxious/Frantic. This is illustrated in Figure 2.1.

Gaussian Mixture Models (Appendix A.3) are used to estimate the audio moods. The hierarchical framework first classifies audio clips according to low or high arousal, based only on intensity features, having one model for this purpose. Then, according to the arousal category, one of two models is deployed to predict if the valence is high or low.

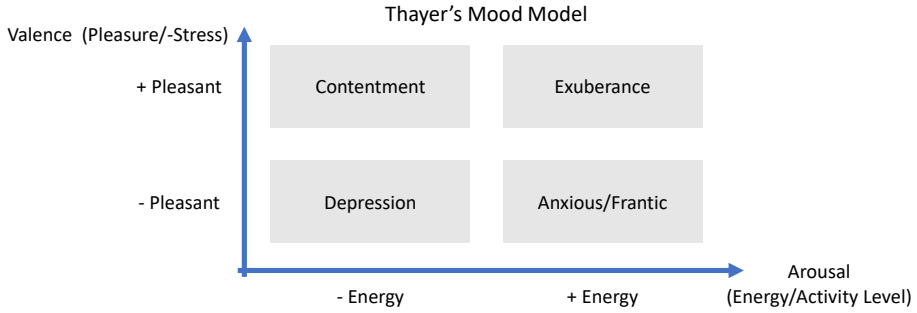


Figure 2.1: Representation of the arousal-valence plane and the corresponding clusters of moods: Depression (low arousal and low valence), Contentment (low arousal and high valence), Anxious/Frantic (high arousal and low valence), and Exuberance (high arousal and high valence). The alternative terms for the stimuli that influence the mood are also indicated in the axis labels. Adapted from Lu [15].

2.4 Creation of Visual Media

The usage of computers to create visual art has been taking place since at least the late 1950s'. This was when an anonymous IBM employee drew a pin-up girl on a glowing cathode ray tube screen[2]. By the 1960s', there were already computer-aided creations that had an element of chance, such as

[2] https://bit.ly/3BDfwYi

the abstract mechanical drawings generated by the machines of Desmond Paul Henry[3]. Due to the development of the personal computer and computer graphics as we know them today, the field of computational art has bloomed.

2.4.1 Exploration of Creative Visuals

Over the last decades, artists and engineers have been exploring the idea of writing programs that could function as artists in their own right.

Cohen [17] developed AARON, a program that uses machines to produce physical paintings, intending to generate original artworks as autonomously as he could program it to. Cohen provides the reference scripts, and the program spends nights producing different outputs that afterward must be analyzed and selected by hand. Although AARON is unable to autonomously filter outputs, Cohen states that its abilities as a colorist surpass his own.

Colton [18] introduced The Painting Fool, a program developed within a project with the goal of creating software that is recognized as an autonomously creative system. In the *Amélie's Progress* gallery[4], an exemplary instance of its work, different digital painting styles were used to create portraits, some manually derived, and some randomly generated. Interesting findings of this project include that the perception of skill and appreciation is crucial for the software to be regarded as an artist.

The CC approach to the study of visual art has provided interesting insights into different methods of generating original artifacts automatically, as well as what visual features might deem a work aesthetically pleasing.

The computer programs described by Dawkins in his 1986 book *The Blind Watchmaker* [19] inspired a lot of the artificial evolution systems that came after it. Dawkins describes the creation of simple fractal structures that can be manually selected for asexual reproduction through small random mutations to the fractal parameters, leading to the formation of interesting line drawings that resemble natural or human-made structures. Since then, the creation of visual media that is generated through artificial evolution has flourished. These systems are capable of creating original results in each iteration, plus allow the incremental improvement of results by means of fitness evaluation, either through user-provided selection, or automatic analysis of results.

For instance, the Mutator software created by Latham and Todd [20] developed art in the form of digital 3D sculptures inspired by biological forms. Their system used the building components of different forms, and combined and modified components from selected individuals to create new ones.

A currently popular approach to computer-generated visual media was initially described by Sims in his seminal work "Artificial Evolution for Computer Graphics" [21], which introduced the use of symbolic

[3] http://www.desmondhenry.com/gallery/
[4] http://www.thepaintingfool.com/galleries/amelies_progress/index.html

expressions as the genotypes that create original visuals. These expressions can be represented as a function tree formed by the random assembly of mathematical symbols as the nodes. This approach has since been widely used to develop systems in the context of CC, with the purpose of developing media creation software or studying aesthetic measures [22–24]. Sims later developed a similar approach to evolve virtual 3D creatures [25], and since then has explored different approaches to procedural animation, such as Julia set fractals, solid noise functions, and emergent shapes from reaction-diffusion systems [26].

A variety of other approaches to the creative generation of visual media have also been deployed. The Electric Sheep software presented by Draves [27] creates and evolves animated screen-savers generated through fractal structures called *sheep*. Using a distributed system, users vote on their favorite *sheep*, which means their genetic code consisting of 160 floating-point numbers is more likely to be reproduced. Users may also submit their own *sheep* creations for evaluation, since they can manually edit the genetic information, in addition to exploring random mutations.

In contrast, Berov and Kühnberger [28] presented a system that alters input images to contain patterns from another image category. The system uses a Convolutional Neural Network (CNN) model trained to identify a specific category, like a dog image. An image from another category, such as an image of a person, is provided as input to the CNN. Then, the pixels of the input image are altered to make it more similar to a dog image, until the model classifies it as such.

Using a similar method, the creation of images with multiple levels of detail according to a category was also explored by Mordvintsev et al. [29]. Additionally, Gatys et al. [30] developed a system that can edit an input image to acquire a style from a given painting.

Elgammal et al. [31] described a method that creates original images through the use of Generative Adversarial Networks (GANs). The system learned about art through a training set of human-generated paintings, and was also trained to recognize different styles. Then, the goal of the image generation module was to create new images that were not considered to belong to any of the styles learned, but that would be classified as art by the art classifier.

2.4.2 Creation of Visuals based on Music

Although the vast majority of music visualization software uses a set of hand-crafted effects which can often be combined to accompany the audio, some works aimed to include a creative component in the process of visual media generation based on audio.

Using music metadata, Zorić and Gambäck [32] created a method for optimizing mappings between music and image features, based on user preference of music-image matching. The mappings are recipes for digital brush strokes, but can be selectively ignored to create novel and interesting results.

Aleixo et al. [33] introduced a system that creates images based on geometric forms, for which the

colors, shapes, sizes, and compositions are determined from cross-domain associations with information from MIDI files. While the background is determined by the harmony of the music, the foreground is determined by the melody, and the sizes of the image components are associated with the rhythm, which also determines the luminosity of the background. Two images per music input are created: one generated with a focus on the distribution of the shapes, and one that applies a genetic algorithm that evaluates the music-image match based on music and color theory.

A method to create visuals that move according to the music was described by Markuš [34], and uses Compositional Pattern Producing Networks (CPPNs) [35] (randomly instantiated neural networks) to create visuals. The inputs are the pixel location and different frequency intensities, while the output represents the color of that pixel. A video is obtained by passing the different frequencies over time and saving the resulting outputs. While the network can be used with randomized weights, the user can tweak the network's parameters to create an animation that is to their liking. Each new weight initialization creates a new animation.

2.4.3 Aesthetic Measures

Since a critical component of creativity is the evaluation aspect, it is important to reflect on how the best visuals out of a random collection may be recognized. Galanter [36] provides a thorough analysis of the different types and implementations of computational aesthetic metrics. A comprehensive overview of the styles created by specific measures and their combinations was presented by den Heijer and Eiben [37]. Some of the previously mentioned works already resort to a variety of different analysis mechanisms. We now provide a few additional works with notable aesthetic measures.

Ross et al. [38] apply Ralph's model of aesthetics to evaluate computer-generated graphics. This model was developed through the empirical study of common features in fine art paintings, and privileges images with bell curve distributions of color gradients. Their work also implements an aesthetic measure that favors images with a color histogram similar to that of a provided target image.

Norton et al. [39] described a system that can be commissioned to create images that respect a list of adjectives. The system contains a set of trained Artificial Neural Networks that recognize whether an adjective is adequate to describe an image, such as *fiery*. The quality of the generated images is measured according to the adjectives that are estimated to apply. New images are generated using image filters, and each adjective will be directly associated with a set of possible filters. In this way, when the system is commissioned to create images that are *fiery*, the filters associated with this adjective (e.g., a red filter) are used to create those images.

In the work by Correia et al. [40], the goal was to achieve computer-generated abstract images that were classified as a target category (such as leaves or faces) by a trained classifier. This approach showed potential for the creation abstract art that is meant to represent some desired content.

Li et al. [41] developed a system that, through interactive evolution of images, the user's aesthetic taste was approximated by learning to map a set of image features to image classifications of preference. The image features used were based on features estimated to represent aesthetic values of humans.

Works deemed particularly useful for the development of our system are described in further detail in the following sections.

2.4.4 Artificial Evolution for Computer Graphics

In his seminal work, Sims [21] described an evolutionary process used to create complex simulated structures, textures, and motions meant to be used in computer graphics and animation. Of particular importance to our work is the methodology presented for the evolution of symbolic expressions that represent textures, which was used to evolve images, volume textures, and animations.

By using symbolic expressions as genotypes, there is no longer a fixed number of genetic parameters, and the structure of information itself is now flexible. Sims used symbolic Lisp expressions, and the function set included: arithmetic functions, exponential, logarithm, trigonometric functions, boolean and conditional functions (and, or, xor, if), noise generators, image warping operations, among others. The possible expression arguments were: scalar values; three-element vectors (RGB values); pixel coordinates such as `x` and `y`; another symbolic expression.

The functions were adapted to receive images and return images, and the final expression determined the color of each pixel. To create animations, the author suggests five approaches, one of which is the use a time variable as a possible argument of the symbolic expression. Sims further described how the expressions could be evolved.

2.4.5 Learning aesthetic judgments in evolutionary art systems

The work presented by Li et al. [41] analyzed the capacity of a set of features to predict an image's aesthetic value. The features considered were based on previous works that studied models of human visual preference.

The authors described an evolutionary art system which, similarly to the work developed by Sims [21], builds images by creating trees constructed from a lexicon of mathematical functions and terminals, which are subject to changes between generations. The trees are evaluated based on the content of the created images, according to color distributions, texture features, and image complexity measures.

This framework was shown capable of learning distinctions between painting styles by different artists. Additionally, it was able to learn different users' preferences regarding the value of images, after being trained using the user's classification of images with low, medium, and high value.

2.4.6 SBArt4 for an Automatic Evolutionary Art

Unemi [24] described a system that automatically evaluates and evolves images and animations based on a set of aesthetic measures, with user-adjustable parameters[5]. The creation of the initial images and animations follows the method described by Sims [21], and the aesthetic measures of images focus on spatial arrangement and color variation. For evaluating animations, 10 random frames from the animation are considered and their image measures are averaged, and the color variation between consecutive frames is also taken into account.

2.5 Influential Systems

The works presented in this chapter relate in one way or another to the task we aim to tackle. Considering the presented systems, we now go over a few key systems that guided our approach.

The automatic music video creation system described by Foote et al. [14] served as a guide for determining audio sections, which dictate the display of different animations. The system of mood detection of music described by Lu et al. [15] inspired the audio features that our model extracts to detect mood from audio, and the features are also used as input for learning user preference of audio-animation pairs.

We owe the inspiration for our animation generation method to the work of Sims [21], in particular to the symbolic expression method introduced to create images and animations. The system by Li et al. [41] provides an extensive description of aesthetic features that proved useful to learn a user's preference towards images, which we used to analyze frames. Finally, the system of automatic evolutionary art described by Unemi [24] further inspired our method for automatic animation creation and evaluation.

The following chapters describe in detail how our system was developed, taking these contributions into account.

[5] The automatic daily production of ten 20-second videos is still ongoing (at the time this work was written), and can be consulted in http://www.intlab.soka.ac.jp/~unemi/sbart/4/DailyWebGL/.

Chapter 3

Approach and System Architecture

The following chapter briefly describes our proposed solution to the challenge of creating animations that use audio as the starting point. Section 3.1 introduces the main ideas behind our approach, and Section 3.2 specifies the architecture of the system implemented in this work. A discussion of the choices made in this work is presented in Section 3.3.

3.1 Approach

Our approach takes an audio file as input and outputs a video file, which presents abstract animations that were created based on the audio features, accompanied by the original audio. The method used to create animations was based on the seminal work of Sims [21], and informed by similar works such as [22–24]. We build upon Sims's idea of generating textures based on symbolic expressions.

A frame consists of a graphical representation of a function on a two-dimensional canvas. Each function results from the composition of elementary functions randomly selected from a candidate set, which take as arguments the pixel coordinates and time-dependent variables. Some of the variables that change according to the frame depend on audio features, so the animation can be directly influenced by the audio itself. In this way, in each frame, each pixel is mapped to a number, and a colormap is used to decide what color that value corresponds to - in fact, our animations are animated heatmaps.

To make the final video more compelling, the audio is sectioned by determining peaks in audio novelty. These sections produce one animation each, which can be generated using unique functions and colormaps, or repeated ones. To reflect the structure of the audio in the video, audio sections that exhibit a similar mood use the same colormap (defining the colormap groups), and audio sections with a similar frequency distribution can use the same animation function (defining the function groups). The computations of colormap groups and function groups occur independently.

Throughout the system, the processes used to determine the colormaps and functions to be used are based on the selection of the best sample out of a large sample set. Particularly, to determine a colormap to be used, first a set of 250 random colormaps is gathered, and then a selection method is deployed to pick one. The same applies to functions, where one is selected out of a batch of 100 new

ones.

New colormaps are generated by gathering a sequence of random RGB values and linearly interpolating between the colors. Already available colormaps may be used as well.

The time-dependent functions are generated by assembling a tree in which the nodes can be mathematical functions and operators, the pixel locations, time-dependent variables such as the subband intensities of the audio, constants, and image patterns. Each type of node has a certain probability of being the child of another node.

The root of the tree is one of the function or operator nodes, and it will have as many child nodes as it has arguments. After the child nodes are selected, their own child nodes need to be determined if applicable, and so on. The tree is complete once there are no more child nodes to be determined. The final function receives a pixel location and the frame index, and returns a value that is then mapped to the [0,1] range. This quantity determines the RGB value of the pixel, according to the colormap. Figure 3.1 shows a few frames of videos produced by the system.

Figure 3.1: Frames extracted from videos produced by the system, using different functions and colormaps, manually chosen for their appeal and diversity.

During the development of our approach, we designed several methods to select colormaps and functions. We started by formulating the Mood Matching method, which aims to select the colormap with a mood similar to that of the audio, and the selected function will be the one that presents a level of detail and an amount of movement proportional to the arousal of the audio.

Since the former method did not produce results as satisfactory as we intended, we went on to develop the Preference Model method. This method attempts to emulate the user's taste regarding what animations are a good choice to accompany an audio, after training a regression model on a dataset with audio-animation pairs and the corresponding classifications provided by the user. The regression model is used to evaluate the fitness of unseen audio-animation pairs. Out of the function sample set set, the function that produced the highest animation fitness is selected. To select the colormaps, random functions are animated using the candidate colormaps, and then the colormap which obtained the highest average animation fitness is selected.

Finally, to allow the system to be used as a co-creation tool, the Manual Selection method was implemented. For each colormap group, this method allows the user to select a colormap out of a collection of 5 candidates gathered by using the Mood Matching method. Similarly, for each function group, 15 animations that use different functions are shown to the user for selection. These functions were considered the top ones for that group according to the Preference Model.

Once the colormaps and animation functions are determined for every section, all the section animations are parsed to create the high-resolution frames that will be used in the final video. The audio sections are processed sequentially and, as the frames are computed, they are assembled into the final sequence. Once the audio is added, the final video is complete.

3.2 System Architecture

We now provide a brief overview of the modules and interactions needed to compute the final video, according to our approach.The code for this thesis was developed in Python.

Figure 3.2 presents an overview of the system's architecture, indicating the main modules that constitute the system, and the elements that are passed between components. For readability, the details of the modules were left out of this architecture overview, and presented in the Figures 3.3, 3.4 and 3.5. The main components required to produce a video out of an audio file are then: Audio Analysis; Animation Analysis - which encompasses Colormap Analysis and Function Analysis and depends on the method chosen by the user; and Final Video Generation. Colormap Generation and Function Generation are also key components of Animation Analysis.

Firstly, the user must provide the audio file in mp3 format, and indicate which animation analysis method they wish to deploy. In case the user opted for the Manual Selection method, further input will be necessary later on, regarding the choices of colormaps and animation functions.

The first computations involve the extraction of all the necessary audio features in Audio Analysis (Figure 3.3(a)). The first audio features to be extracted, called primary features, are computed. Then, in Section Analysis, the audio is segmented by searching for peaks in audio novelty, which will match the

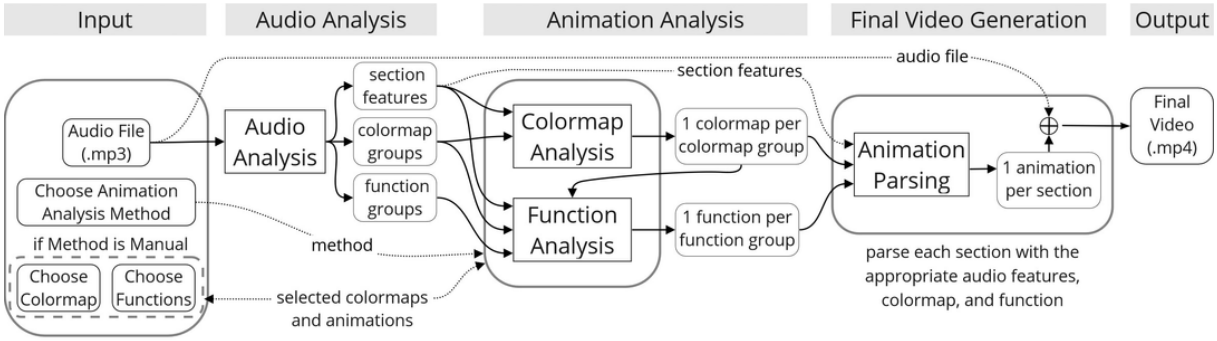


Figure 3.2: Overview of the architecture of the system. The main modules that constitute the system are: Audio Analysis, Animation Analysis - which encompasses Colormap Analysis and Function Analysis, and Final Video Generation. Further details of the architecture are provided in Figures 3.3, 3.4 and 3.5.

points in time where there will be a transition from one animation to another in the final video. The audio novelty peaks are determined based on the method described in [14].

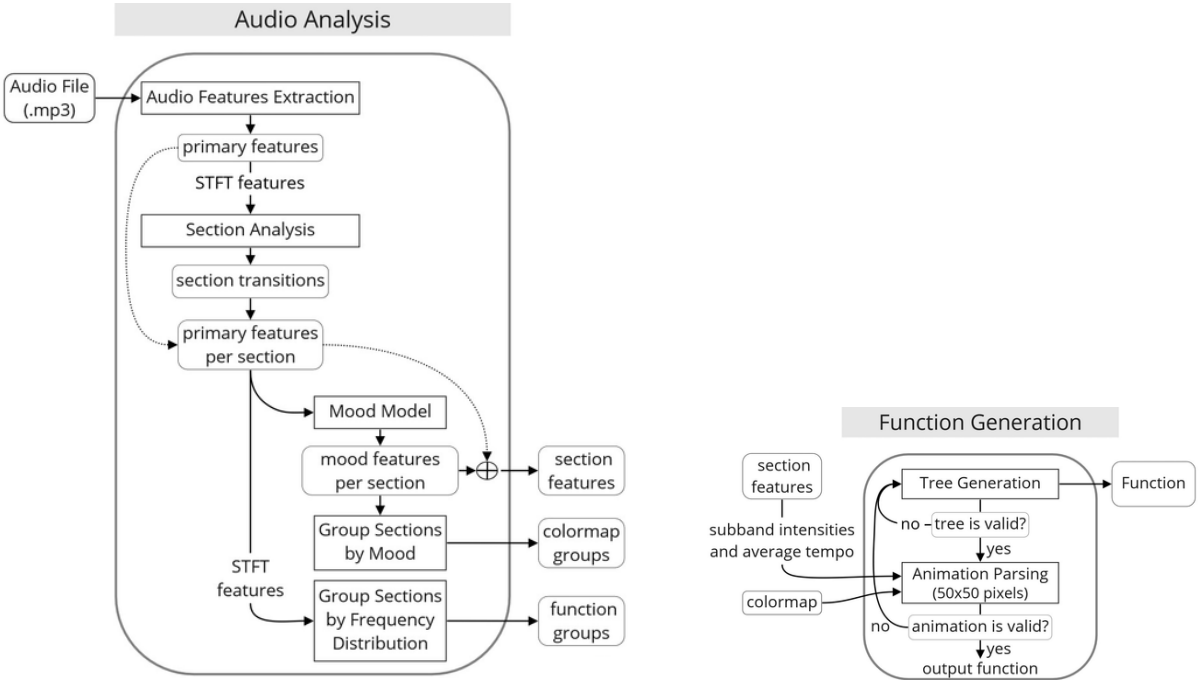
Later on, each audio section will be assigned a colormap and animation function that will be used to create its animation. As an attempt to emulate some structural organization of the audio, some sections will use the same colormap, and some will use the same animation function.

Each audio section is analyzed to determine its arousal and valence values, obtained from a Mood Model that analyzes every second of the audio. This model is in fact composed of two Support Vector Machines (SVM) regression models (one for arousal and one for valence) trained on the dynamic annotations provided by the DEAM dataset [42], and uses the features described in [15]. The mood features (arousal and valence) of each section, together with the primary features, represent the section features.

The audio sections are compared between each other in two ways for two different purposes. First, the arousal and valence values of the sections are clustered, and each cluster represents a colormap group, which dictates the sections that will use the same colormap. In addition, the audio frequency distributions are compared between sections to determine similar audio sections, which will be part of the same function group, i.e., sections that will use the same animation function. The colormap groups and function groups are the output of the Audio Analysis module, along with the section features.

The generation of random animations by Function Generation (Figure 3.3(b)) starts with the assembly of a function tree from scratch, using mathematical functions and operations as internal nodes, and constants, variables, and patterns as the leaves. A colormap and the features of a section are required to test whether the animation is valid. The final function is only provided if the tree and animation are valid, otherwise, a new tree is assembled. This validation process is also required for Colormap Analysis, in case the chosen method is the Preference Model or Manual Selection.

The colormap for each colormap group is determined in Colormap Analysis (Figure 3.4), depending on the method decided by the user. A random set of 250 colormaps is first generated using the Colormap Generation process. The generation of random colormaps is a non-deterministic assembly of a



(a) Audio Analysis

(b) Function Generation

Figure 3.3: Module details: (a) the Audio Analysis module and its sub-modules; (b) the Function Generation process, emphasizing the points where the function's validity is evaluated.

sequence of RGB values that will later dictate the color of every pixel. The sequence of RGB values is manipulated to increase the probability of a good audio-colormap pair being found. In our implementation, the amount of unique colors in the colormaps is proportional to the arousal of the audio.

The system can use three methods for selecting the best colormaps. The Mood Matching method, using the average valence of the colormap group, ranks the colormaps based on how close its colors' valence is to the audio's valence. The valence of a colormap is estimated using hard-coded rules, where lighter and more saturated colors lead to higher valence.

The Preference Model method uses the top 5 colormaps provided by the Mood Matching method, and then determines the colormap to be used. To determine the best colormap, three sections are randomly selected from the colormap group, and three test functions are created. Then, one animation is generated for each combination of colormap, section, and function. The necessary features for preference estimation are extracted, and the fitness of the animations is computed. This method selects the colormap with the highest average animation fitness.

The Manual selection process also depends on the top 5 colormaps from Mood Matching: the user is shown these colormaps, and decides if they want one of those, or if they wish to see 5 new colormaps obtained with the same method. Once the user is pleased with the colormap samples, they indicate which one will be used in that colormap group.

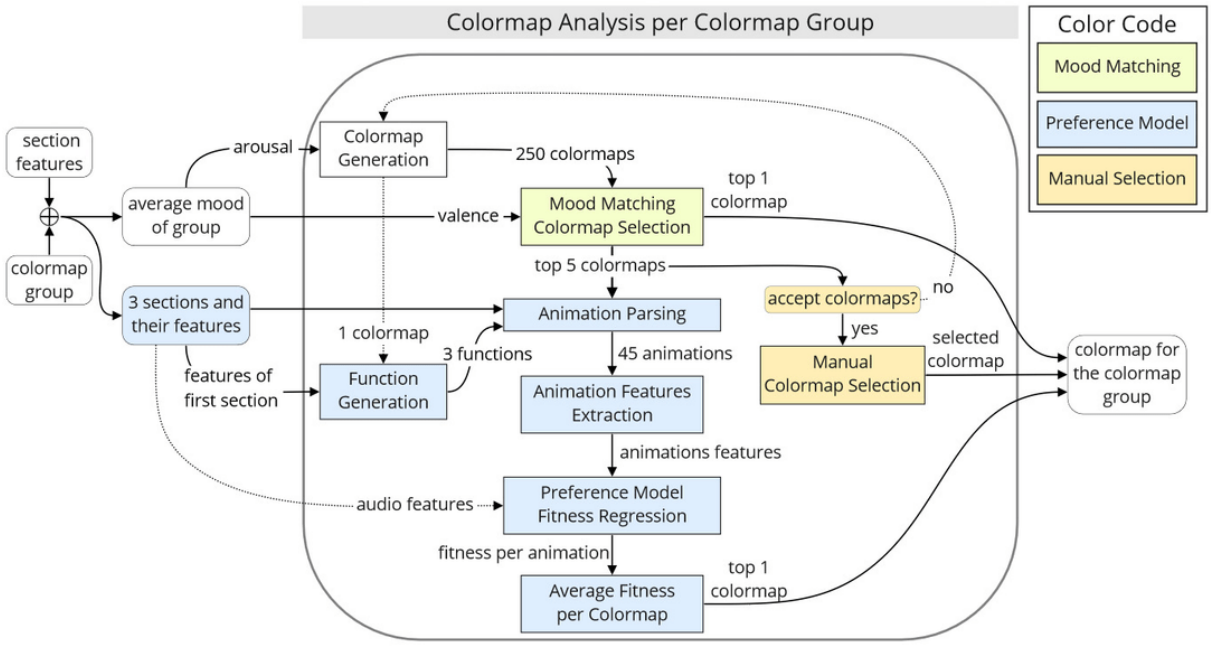


Figure 3.4: Colormap Analysis module, where a colormap is selected for each colormap group. Three methods to select colormaps are available: Mood Matching (green), Preference Model (blue), and Manual Selection (orange). White sub-modules are needed across all methods. Note that both the Preference Model and the Manual Selection methods depend on Mood Matching to filter the colormaps.

The Function Analysis module (Figure 3.5) inspects each function group to select the corresponding functions. The process begins with the generation of 100 different functions, which are all animated using the colormap and audio of the first section of the group. The features of the resulting animations are then extracted.

Again, three methods can be used to choose the functions. The Mood Matching method uses the average arousal of the group, and selects the animation function where the average level of detail in the animation and the average speed of color change, both represented in the [0,1] range, are the closest to the arousal value, a value between 0 and 1. The Preference Model method, after being trained on data provided by the user that represents their preference regarding audio-animation pairs, ranks the animation functions based on the confidence rate that they are according to the user's preference. The Manual selection of animation functions relies on the user to indicate which ones to use, from a set of 15 candidate functions curated by the Preference Model.

After all the colormaps and animation functions are chosen, in Final Video Generation (Figure 3.2), the frames are computed sequentially according to the information of each audio section, namely its features, animation function, and colormap. Once the frames are obtained, the original audio is added, and the final video is provided in mp4 format.

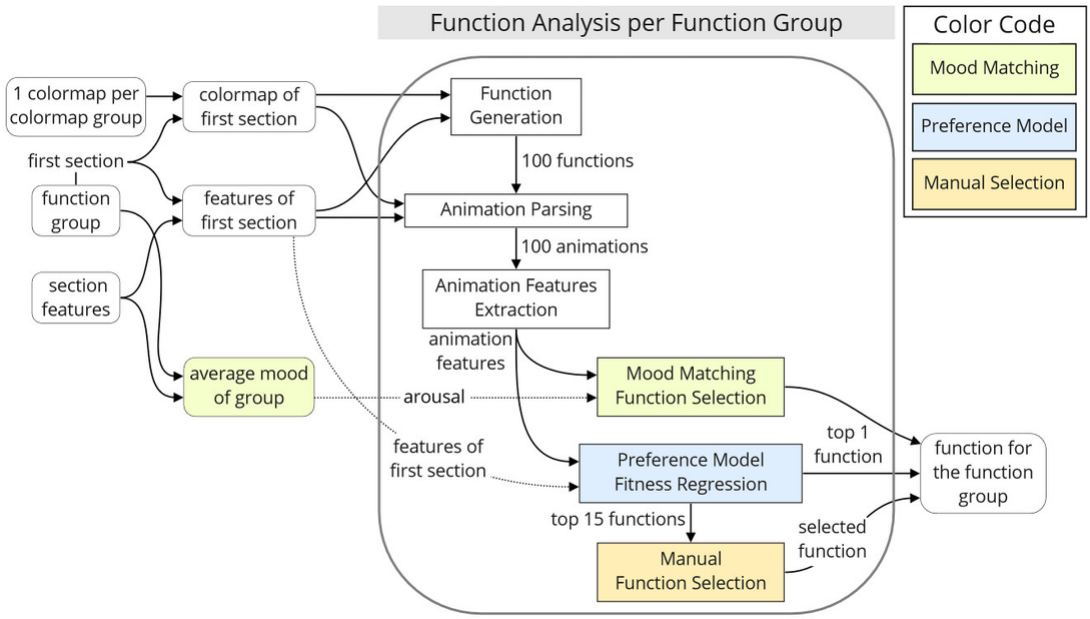


Figure 3.5: Function Analysis module, where a function is selected for each function group. Similarly to the colormap analysis module, three methods to select a function are available: Mood Matching (green), Preference Model (blue), and Manual Selection (orange). White sub-modules are needed across all methods. The Manual Selection process uses the top functions from the Preference Model.

3.3 Discussion

We have described our approach to the problem of animation creation based on audio files. We opted for generating a 2D canvas for each frame instead of creating 3D animations due to its simplicity and potential for building abstract animations from scratch through the use of heatmaps of random functions. The random assembly of unit functions was in turn chosen for its promising capacity to create novel and surprising drawings. In this way, without training or even a dataset, we could start creating new and interesting images that have some underlying structure from scratch. For these reasons, we chose this approach instead of using Generative Adversarial Networks such as in [31], which would require a training set, or Neural Networks such as the ones in [34, 35], which tend to create visualizations with a unique style. We also chose to avoid making hard-coded rules that would determine what would be drawn, in order to maximize the diversity and creative perception of the outputs of our system.

Given the nature of the problem addressed in this thesis, there are many possible satisfactory approaches. Therefore, we present one possible approach, and hope to contribute with a system that uses audio as the basis to generate videos, with creative animations that fit the audio. In the following chapters, we explain in more detail the implementation of each system component.

Chapter 4

Audio Analysis

Our problem starts with the input of an audio file for which a related video is expected to be generated. As such, it is important to consider what significant features the audio has that can be used to develop animations. In this chapter, we detail the audio features that are extracted from the audio to be further processed.

4.1 Audio Features Extraction

We start by extracting the audio features used in the work of Lu et al. [15]. Their work shows that these features can be used to classify the mood of an audio clip, with their model being able to reach an accuracy of 86.3% in mood classification.

In this work, we assume that if an animation is considered to match an audio, then their moods should match as well. Therefore, these features were considered to be a good starting point to describe the audio. We provide a short description of each one, and then move on to other features also considered relevant. Additional descriptions regarding audio analysis can be found in Appendix A.1.

To make sure that all audios are uniform, the audio is resampled to have the sampling rate $\omega_0 = 16$ kHz, converted to monochannel, and analyzed through frames of 0.08 seconds, without overlap. Each audio frame will correspond to an animation frame, resulting in a video of 12.5 frames per second. This animation rate was chosen to allow a somewhat smooth animation, and also minimize the computational power required to compute all the animation frames.

The frequency intensities are obtained through the Fast Fourier Transform (FFT). Following the method described in [15], the intensities are grouped into subbands in a way that is similar to the octave scale, according to the frequency intervals

$$\left[0, \frac{\omega_0}{2^{n_{subs}}}\right), \left[\frac{\omega_0}{2^{n_{subs}}}, \frac{\omega_0}{2^{n_{subs}-1}}\right), \ldots, \left[\frac{\omega_0}{2^2}, \frac{\omega_0}{2^1}\right],$$

where $\omega_0 = 16$ kHz is the sampling rate and $n_{subs} = 7$ is the number of subbands. This subband division is used throughout the audio analysis and animation generation.

The Subband Intensity, A_i, is a measure of the audio intensity of each subband for every frame, and it is computed as:

$$A_i(n) = \sum_{k=L_i}^{H_i} A(n,k), \tag{4.1}$$

where $A_i(n)$ is the Subband Intensity of the i-th subband, $A(n,k)$ is the absolute value for frame n of the FFT coefficient corresponding to the k-th frequency, and L_i and H_i are respectively the lower and upper bounds of the frequency indexes for the i-th subband. A representation of the Subband Intensity for the song "It Could Happen To You" by Chet Baker can be found in Figure 4.1.

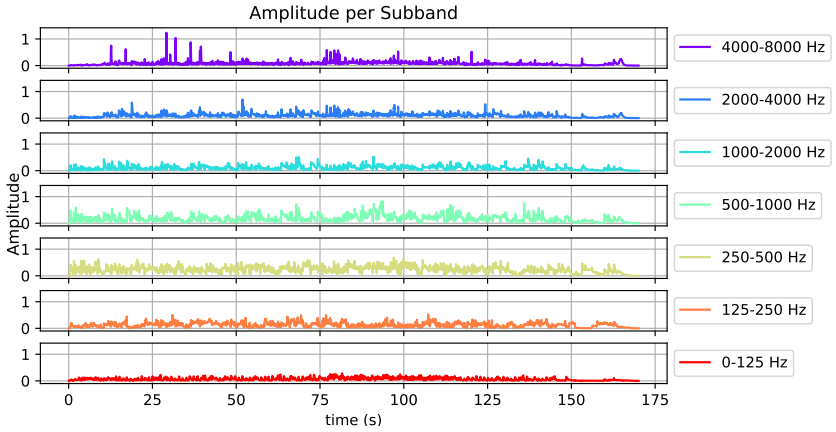


Figure 4.1: Representation of the seven Subband Intensities, for the song "It Could Happen To You" by Chet Baker.

Another essential feature used in our work is the Average Tempo, av_tempo, which is a measure of the average speed of the music performance in seconds. To compute it, we need to estimate the onset sequence of the audio, O, which is calculated by computing the following quantities:

$$\begin{aligned}
h_w(n) &= 0.5 + 0.5cos\left(2\pi \cdot \frac{n}{(2L-1)}\right), n \in [0, L_{hH} - 1], \\
A_i'(n) &= A_i(n) \otimes h_w(n), \\
C(n) &= \frac{n}{\sigma^2} e^{-\frac{n^2}{2\sigma^2}}, \quad n \in [-L_c, L_c], \\
O_i(n) &= A_i'(n) \otimes C(n), \\
O(n) &= \sum_i O_i(n),
\end{aligned} \tag{4.2}$$

where n is the frame index; $h_w(n)$ is the half-Hanning window with $L_{hH} = 12$ chosen empirically; $A_i'(n)$ is the amplitude envelope of the $A_i(n)$ Subband Intensity for the i-th subband; $C(n)$ is the Canny operator with a Gaussian kernel with $L_c = 4$ and $\sigma = 12$; and O_i is the onset curve of the i-th subband. After computing the onset sequence for the whole audio, the auto-correlation curve is computed, correspond-

ing to a self-convolution of the O curve. The four largest peaks in the auto-correlation curve, P_k, are used to compute the Average Tempo in the following manner:

$$av\_tempo = arg\ \min_{P_k} \sum_{i=1}^{N} \left| \frac{P_i}{P_k} - round\left(\frac{P_i}{P_k}\right) \right|, \tag{4.3}$$

where P_i corresponds to the time of the i-th peak in the auto-correlation curve, and the $round(\cdot)$ function transforms the value to the closest integer. Equation (4.3) computes the maximum common divisor of the onset peaks. More details regarding the computation of the Average Tempo can be found in [15].

Besides the features used in [15], the Mel Frequency Cepstral Coeficients (MFCC) were also extracted from the audio, similarly to the work of Tzanetakis and Cook [43]. We chose to use 20 coefficients as an attempt to keep most of the audio information before any feature reduction was performed by the animation analysis module.

For Section Analysis, following the work of [14], we compute the Short-time Fourier transform (STFT) coefficients of each audio frame and compute the logarithm of their magnitude. The final representation of each frame is then obtained by grouping them into 30 evenly spaced bins between 0 and $w_0 = 16$ kHz.

A brief overview of the audio features is represented in Table 4.1, and a more in-depth description of each one can be found in the works indicated in the last column. The presented list corresponds to what we call the primary features, as they precede audio segmentation and mood estimation.

Each rhythm feature corresponds to a unique value for the whole audio, while the other features are computed frame by frame. This information is represented in Table 4.1 by the Span column. Together with the column regarding feature count (Count), we can know how many values a feature encompasses. For instance, in the case of Subband Intensity, we have seven features per frame, since Count is 7 and Span is Frame, corresponding to one feature per subband per frame. Likewise, in the case of the Average Tempo, there is only one value to represent the entire audio, since Count is 1 and Span is Entire Audio. Since we will need to condense information to represent audio sections, we can represent these features by their mean and standard deviation over that time period.

The features presented in Table 4.1 will be used in Section Analysis (SA), Mood Analysis (MA), Animation Generation (AG), and Preference Model (PM). We will further specify their usage by the modules in future sections.

4.2 Section Analysis

To best capture the temporal audio structure, it is desirable that the animation also displays temporal patterns. For instance, when it comes to music videos, it is frequent to see drastic visual changes

Table 4.1: Primary audio features that are extracted to be used by the system (Feature) and their definitions (Description). The column Count indicates the number of different features a description corresponds to, and the Span informs about the time window that is used to compute each feature. Includes information about where the features are used in our system: Section Analysis (SA), Mood Analysis (MA), Animation Generation (AG), or Preference Model (PM). These features were adapted from [14, 15, 43], as indicated by the last column.

Feature	Description	Count	Span	SA	MA	AG	PM	From
Intensity	Spectrum sum of the signal	1	Frame		X		X	
Subband Intensity	Spectrum sum of the signal over each subband	7	Frame			X	X	
Subband Intensity Ratio	Spectrum distribution in each subband	7	Frame		X		X	
Brightness	Centroid of short-time Fourier amplitude spectrum	1	Frame		X		X	
Bandwidth	Weighted average of the differences between the spectral components and the centroid	1	Frame		X		X	
Roll off	95-th percentile of the spectral distribution	1	Frame		X		X	
Spectral Flux	2-Norm distance of the frame-to-frame spectral amplitude difference	1	Frame		X		X	
Subband Peak	Average of a percent of the largest amplitude values in the spectrum of each subband	7	Frame		X		X	[15]
Subband Valley	Average of a percent of the lowest amplitude values in the spectrum of each subband	7	Frame		X		X	
Subband Contrast	The difference between the Peak and Valley in each subband	7	Frame		X		X	
Rhythm Strength	The average onset strength in the onset sequence	1	Entire Audio		X		X	
Average Correlation Peak	The average strength (amplitude) of the local peaks in the auto-correlation curve	1	Entire Audio		X		X	
avr(A)/avr(V)	The ratio between the average peak strength and average valley strength	1	Entire Audio		X		X	
Average Tempo	Represents the average speed of the music performance	1	Entire Audio	X	X	X	X	
Average Onset Frequency	The ratio between the number of onsets and the corresponding time duration	1	Entire Audio		X		X	
MFCC	FFT coefficients grouped into 20 bins according to the Mel-frequency scaling	20	Frame				X	[43]
STFT	Logarithm of the magnitude of STFT coefficients, grouped into 30 bins	30	Frame	X				[14]

matching the beat. Based on the work of Foote et al. [14], we detect temporal peaks in audio novelty. These peaks determine a transition in time from one animation to another, forcing visual novelty to occur. Using this approach, the final result is a sequence of different animations concatenated, and in which the transition points match the estimated audio novelty.

Furthermore, since the audio can have portions that are very similar to each other, i.e., the chorus of a song, it is desirable that the visuals capture this structure as well. For this reason, we also estimate how similar the audio sections are between each other, and this will determine if and where certain portions of the visuals will be repeated.

We start by presenting the audio features that are used to detect audio novelty, then outline the self-similarity analysis method and the segmentation criteria. It should be noted that we are interested in

finding the frames where a new animation will start, so our measures will be in terms of frames, and the value of seconds per frame, $spf = 0.08$ seconds, will be used in the following discussion.

4.2.1 Audio Parametrization

To represent the audio content frame by frame for audio sectioning, we use the same vector representation of a frame as in [14]. For each frame, an STFT is applied to the audio, and the power spectrum is obtained by taking the logarithm of the magnitude of the amplitudes. The frequencies are binned into 30 evenly spaced intervals in the power spectrum, so each frame is represented by a vector with 30 features. The feature vector of frame i is denoted as $\boldsymbol{v_i}$. The frame representation for the song "Prelude In E Minor (opus 28, nº 4)" by Chopin is represented in Figure 4.2.

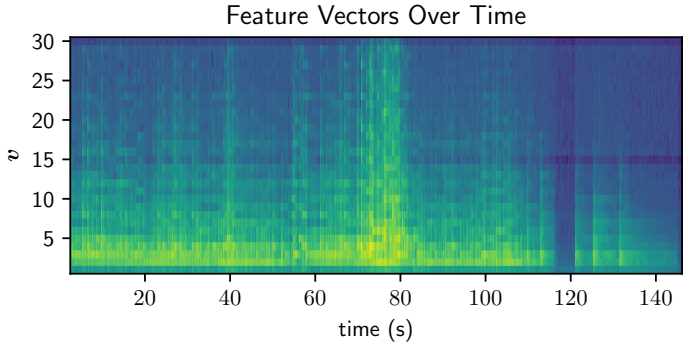


Figure 4.2: Representation of the feature vectors, $\boldsymbol{v}$, obtained from the power spectrum computed using STFT coefficients, binned into 30 frequency intervals. Representation of the song "Prelude In E Minor (opus 28, nº 4)" by Chopin.

Other possible feature vectors were experimented to represent frames, such as Subband Intensity and Subband Intensity Ratio, each using 7 features for frame representation (one for each of the 7 frequency subbands).

We also attempted several techniques to smooth the features over time to see if the sections would become more apparent using this approach, such as centered moving average where different window sizes were tested, and digital filters[1].

However, after experimenting different filter parameters, we found that, although the sections became more visually apparent after using filters, the audio novelty peaks were less precise. From all these experiments, the results obtained from the STFT representation without any smoothing filter aligned better with our perception of audio novelty, so this is the representation used to compute the frame similarity in the next step.

[1] We used the following filters implemented in Python's scipy module: scipy.signal.lfilter and scipy.signal.savgol_filter.

4.2.2 Self-Similarity Analysis of Audio

To achieve a measure of audio novelty over time, we first measure how each frame is similar to the other frames. Because we want to consider dissimilar frames as ones belonging to different notes or chords, cosine similarity is used, as small changes in amplitude are not important. To compare two frames, we take their feature vectors and compute the cosine similarity between the two, defining the Frame Similarity, $S(i, j)$, as:

$$S(i,j) = \frac{\boldsymbol{v_i} \cdot \boldsymbol{v_j}}{\|\boldsymbol{v_i}\|\|\boldsymbol{v_j}\|} \tag{4.4}$$

where i and j are the frame indexes, $\boldsymbol{v_i}$ is the feature vector of frame i, and $\|\boldsymbol{v_i}\|$ indicates the norm of $\boldsymbol{v_i}$. By computing the similarities between all pairs of frames, we can represent the Self-Similarity of the audio with a symmetric matrix, S, where each entry is the Frame Similarity $S(i, j)$ between two frames. Figure 4.3 shows the S matrix for the song "plyPhon" by Autechre.

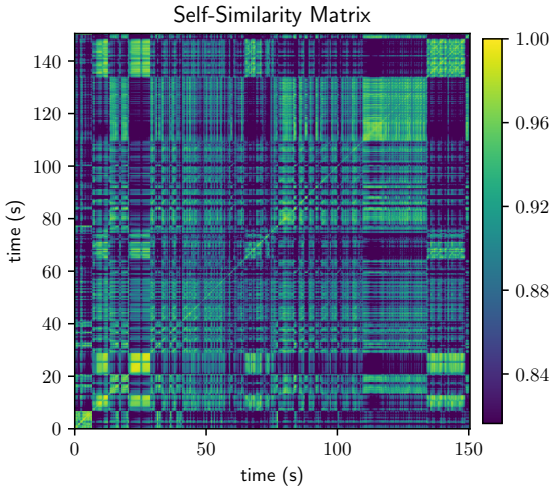


Figure 4.3: Self-Similarity Matrix for the song "plyPhon" by Autechre, where each frame comparison was performed using Equation (4.4).

4.2.3 Audio Segmentation via Kernel Correlation

To segment the audio, we now use the Self-Similarity Matrix to compute the audio novelty and look for its peaks, which will be used to determine a transition point between one animation and the next. The goal of the audio novelty is to approximate the perception of a significant change in the audio. To achieve that, we want to look for timestamps in the audio where the frames before that point are very similar between each other, and after that point the frames are also very alike, but the two sets of frames (before and after that timestamp) are considerably different.

Given the Self-Similarity Matrix, the timestamps with high audio novelty form a checkerboard-like

pattern along the diagonal (high similarity between the frames in the section before and the section after, but low similarity between the two sections). Therefore, to detect the peaks, we apply a convolution of a checkerboard-like kernel along the diagonal of the S matrix. The simplest kernel of this kind has the form $\left[\begin{smallmatrix} -1 & +1 \\ +1 & -1 \end{smallmatrix}\right]$, and larger kernels can be built by having all the values of each quadrant be either 1 or -1 in an alternate fashion. In our case, the kernel indexes will vary from $-L$ to L, and thus have a width of $2L + 1$.

Since the shape of the kernel affects the shape of the audio novelty curve, some experiments were carried out to assess what kernel shape would yield the best peak locations according to our personal preference. We wanted the sections to be related to the Average Tempo of the audio (Equation (4.3)), but also wanted sections short enough to create a dynamic animation. The kernel that best fulfills these goals reaches L frames from its center according to:

$$L = round\left(\frac{min(\{av\_tempo, 3\})}{spf}\right), \tag{4.5}$$

where $spf = 0.08$ seconds per frame, av_tempo is the average tempo of the audio in seconds, and $round(\cdot)$ makes sure that the result is an integer. This results in a maximum L of 37 frames (in case $av\_tempo \geq 3$), and thus a maximum kernel size of 75×75.

Since we wish to reduce the importance of frames further away from the center of the kernel, we use a 2D radially-symmetric Gaussian function to achieve this effect:

$$G(m, n) = \exp\left(-\frac{(r - \mu)^2}{2\sigma^2}\right) = \exp\left(-\frac{m^2 + n^2}{2L^2}\right), \tag{4.6}$$

where $m, n \in \mathbb{Z} : m, n \in [-L, L]$ are the kernel indexes, and where $r = \sqrt{m^2 + n^2}$, $\mu = 0$ and $\sigma = L$ lead to the final equality. The standard deviation of the Gaussian is set to $\sigma = L$ based on the same reasoning as with the kernel span. The kernel is finally defined as [14]:

$$C(m, n) = \begin{cases} G(m, n), & \text{for} \quad m, n > 0 \quad \vee \quad m, n < 0, \\ -G(m, n), & \text{otherwise.} \end{cases} \tag{4.7}$$

Figure 4.4 represents a visualization of the kernel, for $L = 37$ frames.

The Novelty Score of the i-th frame, $N(i)$ is formalized as:

$$N(i) = \sum_{m=-L}^{L} \sum_{n=-L}^{L} C(m, n) S(i + m, i + n), \tag{4.8}$$

where S is defined in Equation (4.4), L in Equation (4.5), and C in Equation (4.7). To compute the novelty near the edges of the matrix, S is padded with the closest valid value with the intent of minimizing novelty right at the beginning and the end of the audio. To minimize the amount of peaks near the borders, we find the first local minimum and set to zero the audio novelty before that point, and find the last local

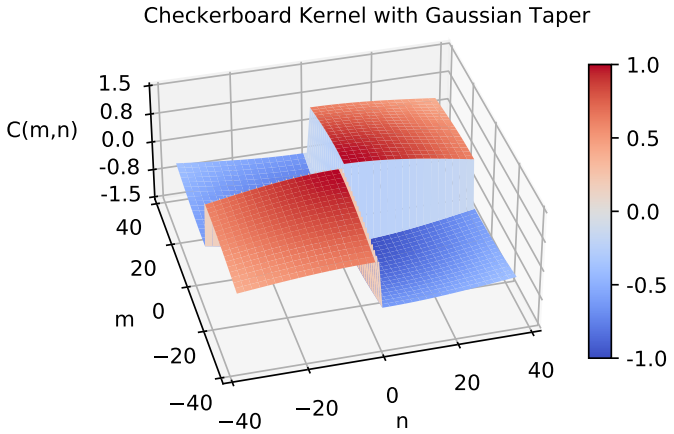


Figure 4.4: Representation of $C(m, n)$ (Equation (4.7)) used to compute the Novelty of the song "Prelude In E Minor (opus 28, nº 4)" by Chopin, with $L = 37$ ($av\_tempo = 3.2 > 3$ s) and kernel size of 75×75 frames.

minimum and set to zero the novelty after that point. The audio novelty for the song "Prelude In E Minor (opus 28, nº 4)" by Chopin is shown in Figure 4.5(a).

The final step is to compute the local peaks of the Novelty Score. To make sure that the section transitions were to our liking, we defined a minimum distance that two consecutive peaks should have, i.e., minimum time interval between transitions, according to:

$$min\_peak\_distance = round\left(\frac{min(\{0.9 \cdot av\_tempo, 2\})}{spf}\right), \tag{4.9}$$

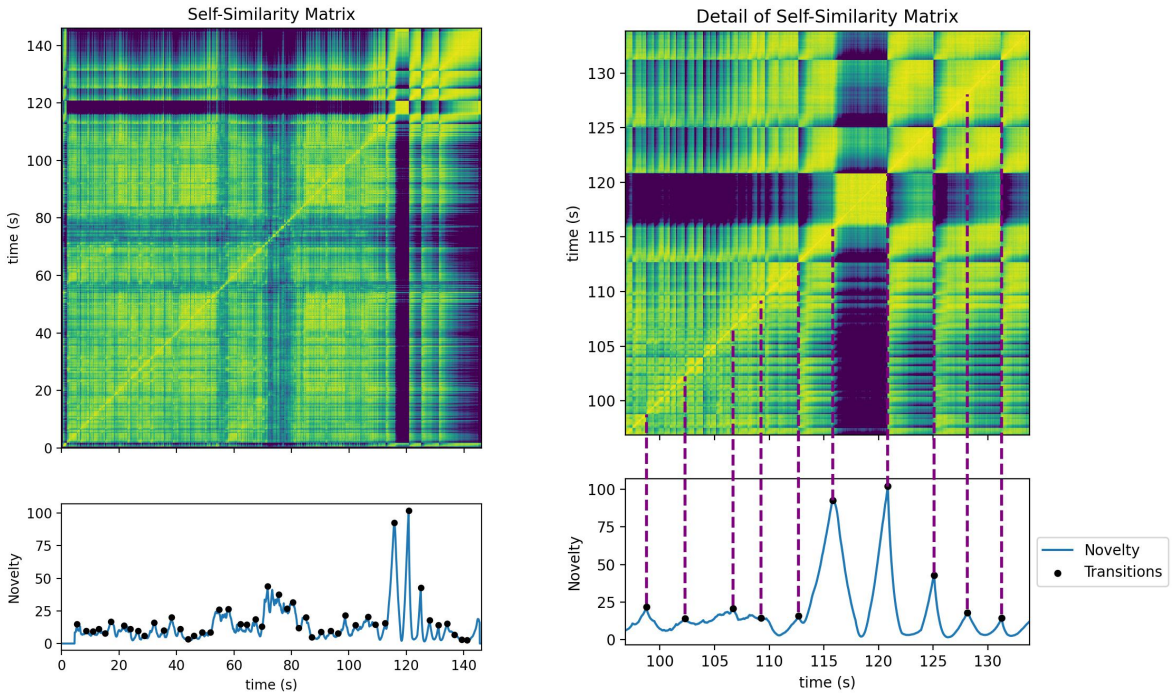
so that audios with a smaller tempo are able to have a more dynamic animation.

To make sure that this constraint is satisfied, after computing all the local maximums in the audio novelty, the smaller peaks are removed first until this condition is fulfilled between two consecutive peaks. In addition, we disregard peaks that are too close to the beginning or end of the audio, that is, closer than $min\_peak\_distance$.

The resulting peaks determine the starting frames of new animations. Figure 4.5(a) shows an example of a Self-Similarity Matrix and the corresponding audio novelty and transition points, with a close-up being displayed in Figure 4.5(b). By analyzing the figure, we can see that the more evident the checkerboard-like appearance of the S matrix, the higher the novelty score. The obtained audio sections are processed individually in the following modules.

4.3 Mood Model

In this work, we considered that the mood expressed in the audio should also be expressed in the visuals generated. For this reason, a regression model was trained based on the audio features to estimate the mood present in an audio segment. We now provide an overview of the mood regression model.



(a) Self-Similarity Matrix and Novelty Score **(b)** Detail

Figure 4.5: Self-Similarity Matrix and Novelty Score for the song "Prelude In E Minor (opus 28, nº 4)" by Chopin: (a) overview; (b) close-up showing the alignment between the Novelty peaks and the corresponding matrix points. The Novelty peaks determine the section transitions used in our approach.

Our approach is based on Thayer's model of mood [16], according to which the mood can be characterized by two quantities, valence and arousal, defined within the [0,1] interval. The valence is a metric of how pleasant or unpleasant a mood is, yielding larger values for more positive moods, while the arousal is proportional to how intense or energetic a mood is.

From these two quantities, the mood can be classified into four categories based on which quadrant of the arousal-valence plane it is located: Depression (arousal < 0.5 and valence < 0.5), Contentment (arousal < 0.5 and valence ≥ 0.5), Anxious/Frantic (arousal ≥ 0.5 and valence < 0.5), and Exuberance (arousal ≥ 0.5 and valence ≥ 0.5). These relations are represented in Figure 2.1. We adopted this mood taxonomy for being able to capture the most common emotional responses to music [15], in addition to its simplicity. We opted for building two regression models, one for arousal and one for valence, in order to maximize the granularity of the mood extracted.

4.3.1 The DEAM Dataset

We obtained the training data from the DEAM dataset [42], which contains 1744 song excerpts and 58 entire songs. The excerpts last 45 seconds and are from a random starting point in the song. We

considered the dynamic annotations provided in the dataset, which contain a score between -1 and +1 for arousal and valence of each time window of 1 second, and were sampled every 0.5 seconds (excluding the first 15 seconds). These values were acquired based on the average scores provided by multiple payed annotators. Given all the provided 1 second window annotations, a total of $N = 129987$ data points were used.

The dataset was randomly split between training and test sets in the proportions 80-20 $\%$, respectively, and the same sets were used for arousal and for valence. For convenience, we converted the values between -1 and +1 into the range [0,1] by applying the linear transformation $x \rightarrow 0.5x + 0.5$, where x is either the original valence or arousal.

The samples are represented in terms of annotated arousal and valence (after the domain transformation) in Figure 4.6(a). Although the data points are not spread evenly across the entire plane, we assumed that this distribution is a good representation of the audios that we would work with, due to the number of audios and genre diversity in the dataset.

4.3.2 Data Preprocessing

In our approach, we estimate the valence and arousal of an audio segment based on the audio features used on the work of Lu et al. [15], as indicated by the column MA of Table 4.1, since these allowed them to achieve state-of-the-art results in mood classification. Although many other audio features are said to be related to mood perception, we focus on intensity, timbre, and rhythm features for their notorious influence on mood perception. These were also chosen for being measurable from acoustic data, which is not always the case [15].

Every feature was transformed to have a mean of 0 and a standard deviation of 1, based on the training set, and Principal Component Analysis (PCA) [44] was performed to remove any correlation between the features. No feature reduction was performed. The scaling and PCA coefficients were saved and are always applied as a preprocessing function before using the features in the mood model.

We extracted the audio features for each audio sample, and then processed the features in order to represent each second in the following manner. Since the intensity and timbre features are collected for every frame, these are represented by an average and a standard deviation for every second. Since the Rhythm features concern the entire audio, they are constant across every second of the audio.

4.3.3 Prediction of Audio Mood

Once we had the feature representation and the target valence and arousal for every one-second audio segment from our dataset, we trained and analyzed different models to choose the one with the best results.

We started by training classifiers using Gaussian Mixture Models [45] with 16 mixtures, inspired by the work of [15], by converting the continuous annotation of arousal and valence of each data point to the values 0 or 1 based on the annotation being below 0.5 (yielding 0) or above or equal to 0.5 (yielding 1). Since we weren't able to transform the resulting classifications in regression values in a satisfactory manner, we turned to regression models instead. We weren't satisfied with pure classification results since these would not provide the nuance we would get from mood regression.

The regression specialized models we tried were Linear Regression (LR), Multilayer Perceptron (MLP), Decision Tree (DT), and Support Vector Regression (SVR) (an overview of the models can be found in Appendix A.3). Python implementations of these models were used, resorting to the `scikit-learn` library[2]. We trained two models independently, one for arousal and one for valence.

Although we tried to divide the valence model in two according to the arousal (one valence model for low arousal and another for high arousal, inspired by the work of [15]), the models that in the end provided the best results correspond to one model for arousal and a unique model for valence. Although we anticipated that discriminating the models between low and high arousal would lead to better results for valence, the reduction in half of the training dataset might have caused the predictions to be less precise. In addition, this approach was also dependent on the performance of the arousal model.

We also attempted to use only intensity features for arousal and only timbre and rhythm features for valence (as described in the work of [15]), but similarly, this led to lower metrics. We obtained the final models by using all the features and by not discriminating the valence model based on the arousal results.

4.3.4 Results

To evaluate the models, we used Root Mean Square Error (RMSE), defined in Equation (A.1), and Pearson's correlation coefficient, ρ, defined in Equation (A.6), between the predicted and ground-truth mood values. Table 4.2 shows the final metrics obtained on the test set for the attempted regression models.

As indicated in bold, the best model results were obtained from SVR models for both arousal ($RMSE = 0.129$, $\rho = 0.890$) and valence ($RMSE = 0.145$, $\rho = 0.819$), which lead to the usage of these models when computing the mood metrics

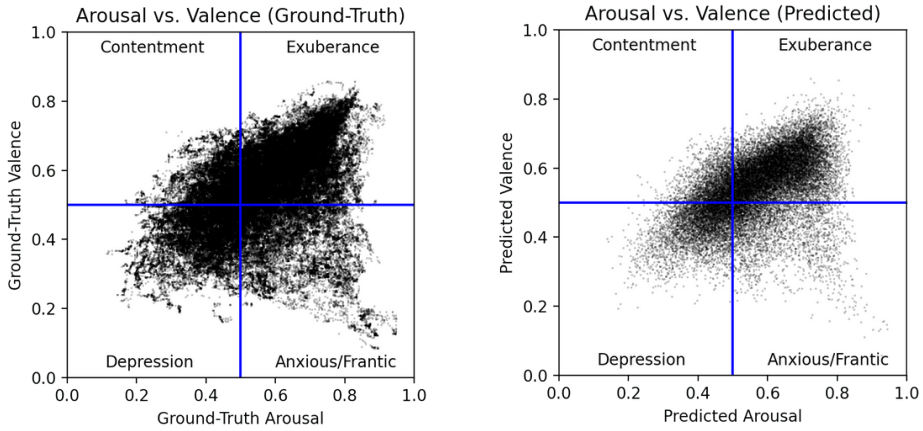
The test set results of the final model are laid out in the arousal-valence plane in Figure 4.6(b). We can see that the distribution of the data points is very similar to the distribution of the ground-truth ones according to Figure 4.6(a), which was expected, since the models tend to predict values close to the ones they were trained on.

The predicted arousal and valence on the test set are also compared to the ground-truth annotations

[2] https://scikit-learn.org/stable/index.html

Table 4.2: Mood prediction results on the test set for the attempted regression models based on $RMSE$ and ρ. Since the best results are indicated by the lowest $RMSE$ and highest ρ values (highlighted in **bold**), the final models correspond to an SVR model for both arousal and valence, highlighted in gray.

Model	Arousal		Valence	
	RMSE	ρ	RMSE	ρ
LR	0.197	0.716	0.224	0.450
MLP	0.173	0.789	0.198	0.613
DT	0.166	0.810	0.149	0.806
SVR	**0.129**	**0.890**	**0.145**	**0.819**



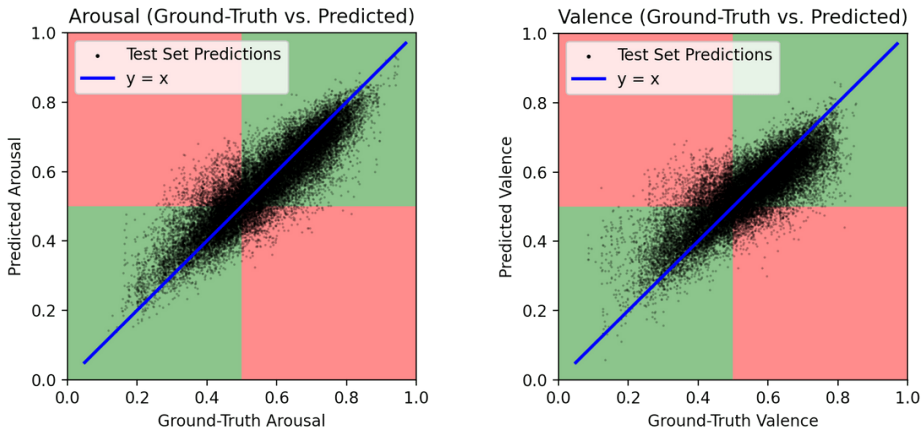
(a) Ground-truth **(b)** Predicted

Figure 4.6: (a) Representation on the arousal-valence plane of the ground-truth annotations from the DEAM dataset. (b) The predicted results obtained from the trained SVR models when applied to the test set.

in Figure 4.7(a) and Figure 4.7(b), respectively. As the images show, the distributions are very close to the $y = x$ line, which means that the model is able to approximate its mood estimations to the ground-truth.

4.3.5 Rescaling of Mood Measures

Because the mood metrics were so concentrated near the center of the arousal-valence plane, a linear transformation of the values was further imposed, with the purpose of amplifying the distinction between the moods predicted by the Mood Model. In essence, we looked for a transformation that would keep the central point of the plane unchanged, but would spread out the remaining points, while keeping most of the data within the $[0, 1] \times [0, 1]$ domain. We analyzed possible transformations based on the prediction results on the test set, since this allowed for a more realistic representation of what values would be obtained when computing the mood for new audios. The rescaling of the outputs is defined by



(a) Arousal Results

(b) Valence Results

Figure 4.7: The predicted results obtained from the trained SVR models regarding the test set, compared to the ground-truth annotations from the DEAM dataset: (a) arousal results; (b) valence results. The closer the data points are to the $y = x$ line, the closer the prediction was to the ground-truth. The colors of the quadrants indicate if the classification of the mood value - low/high arousal and low/high valence - is correct (green quadrants) or not (red quadrants).

the following transformations:

$$\begin{aligned} arousal' &= 2.5 \cdot arousal - 0.75, \\ valence' &= \frac{5}{3} \cdot valence - \frac{1}{3}, \end{aligned} \tag{4.10}$$

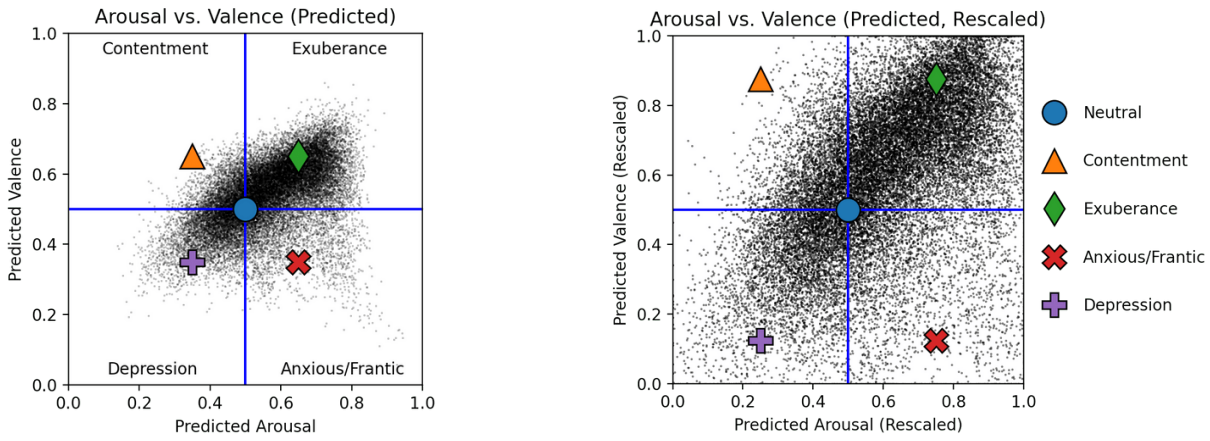
where $arousal'$ and $valence'$ are the final rescaled mood metrics, based on the original predicted values $arousal$ and $valence$. After rescaling, values of arousal and valence outside the interval $[0, 1]$ are clipped to the interval limits. This transformation in represented for the test set results in Figure 4.8.

Now that we have the final mood models, we can estimate the arousal and valence of each audio section by averaging the mood values over the seconds of the section. This section characterization is used for colormap generation and for animation analysis. Both usages are detailed in future sections.

4.4 Colormap Groups - Group Sections by Mood

Because we want to use the same colormap for sections with similar moods, the sections are clustered using arousal and valence as the features. Each mood cluster constitutes a colormap group. Since we want the number of clusters to be determined according to the features of the sections, clustering is performed in a hierarchical manner, using a Python implementation[3] of the OPTICS algorithm [46]. The restrictions are: a cluster should have at least 3 samples, and the minimum distance between cluster members is 0.5. Considering that this method classifies some samples as outliers, we assign a cluster

[3] sklearn.cluster.OPTICS



(a) Before Rescaling

(b) After Rescaling

Figure 4.8: (a) Representation of reference points $(arousal, valence)$ before the transformation: Neutral $(0.5, 0.5)$, Contentment $(0.35,0.65)$, Exuberance $(0.65,0.65)$, Anxious/Frantic $(0.65,0.35)$, Depression $(0.35,0.35)$. (b) Rescaled point $(arousal, valence)$, after applying Equation (4.10): Neutral $(0.5, 0.5)$, Contentment $(0.125, 0.75)$, Exuberance $(0.875, 0.75)$, Anxious/Frantic $(0.875, 0.25)$, Depression $(0.125, 0.25)$.

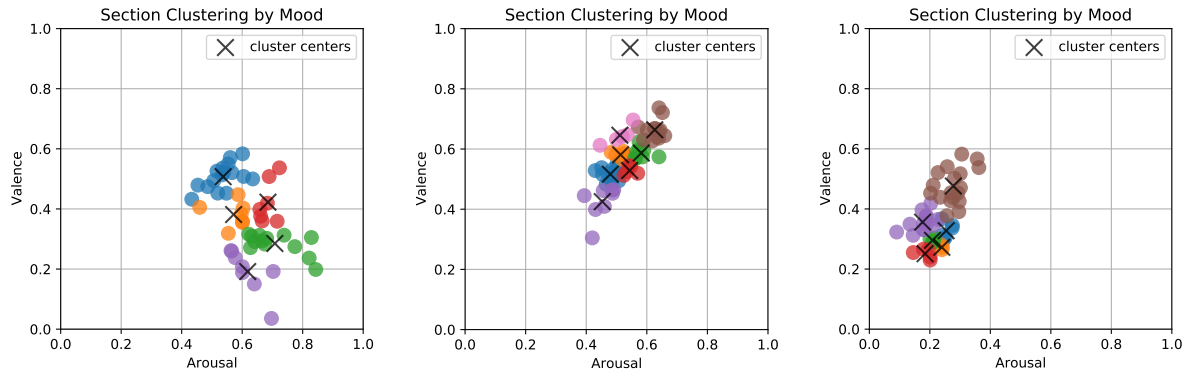
to each of these sections by finding the closest section in terms of arousal and valence. The cluster centers are computed by averaging over the arousal and valence of the sections of the cluster, and these new mood features are used to describe the section clusters. Figure 4.9 shows some examples of mood clusters.

4.5 Function Groups - Group Sections by Frequency Distribution

Because we are also interested in repeating animation functions in sections where the audio is very similar, we want to group the sections based on frequency similarity, to form what we call function groups. To do this, we define a metric for section similarity. A section is represented by the average feature vector of all the frames, $\boldsymbol{v_i}$ belonging to that section, where the average is computed per feature.

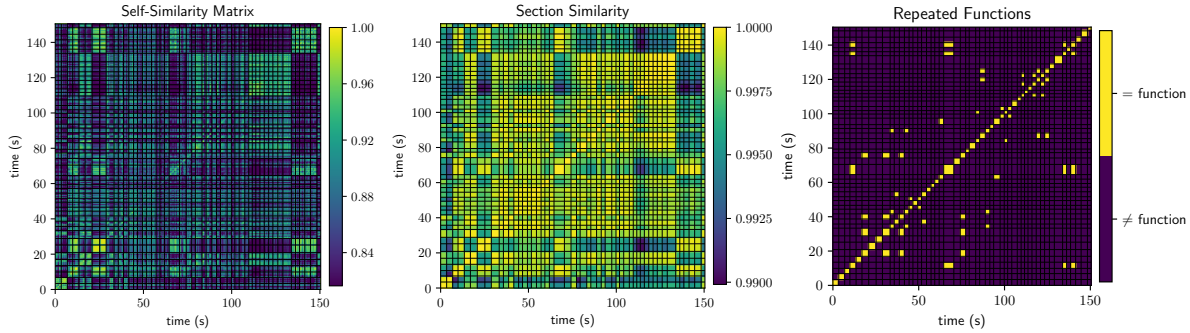
For measuring section similarity, we use the cosine similarity between the section vectors. After experimenting with different similarity thresholds to see which one would give us the best balance between matched and unique sections, the minimum similarity for considering a match was set to 0.9997.

We now describe the section matching algorithm. Each section is compared with the previous sections, and the section with the highest similarity is considered. If they have a similarity above the threshold, then the two sections are considered a match, and will use the same animation function. In this way, we iteratively form groups of matched sections. Figure 4.10 shows an example of the frame similarity of the song “plyPhon” by Autechre, with the indication of section transitions, the corresponding section similarity, and the final section matching results.



(a) "plyPhon" **(b)** "It Could Happen To You" **(c)** "Prelude in E Minor (opus 28, nº 4)"

Figure 4.9: Representation of mood clustering. Each circle is a section, and each color represents a different cluster, with its center indicated by a cross. The songs are: (a) "plyPhon" by Autechre; (b) "It Could Happen To You" by Chet Baker; and (c) "Prelude In E Minor (opus 28, nº 4)" by Chopin.



(a) Frame Similarity **(b)** Section Similarity **(c)** Repeated Functions

Figure 4.10: Representation of: (a) frame similarity; (b) section similarity; (c) function repetition. The transitions between sections are indicated with black lines. The color scales were chosen to highlight the similarity patterns. The song used corresponds to "plyPhon" by Autechre.

To avoid having some animations repeated too often, a few additional rules were enforced. Firstly, to avoid using the same animation in consecutive sections, a section is considered to not have a match if its best match is the one right before it. In addition, the same animation cannot be used more than two times in an alternate fashion, i.e., if a sequence of animations is currently of the form **A-B-A-C-A** (where **B** and **C** can be the same or not), then the last **A** should be converted to a new animation to make the sequence **A-B-A-C-D**.

When analyzing which function should be used for the function group, only the first section of the group is considered, for simplicity. Other sections of the function group will use the same function, but may use different colormaps. Even when using the same function and colormap, animations may differ for different sections significantly, since the animations depend on the audio frames as well.

Chapter 5

Animation Generation

The generation of a new animation, according to our approach, can be divided into three main aspects: colormap generation, function generation, and animation parsing. We discuss these steps in the following sections, and overview the animation features extracted for animation analysis.

5.1 Colormap Generation

A colormap is a function that maps values in the [0,1] range to RGB values. In our system, when fetching a new colormap, we either get an already existing colormap from Python's `matplotlib` library[1] with 30% probability, or create a new colormap from scratch with 70% probability.

Our method of random colormap generation obtains a random sequence of RGB values. Once we have a sequence, the first color is associated with the 0 value, the last one with the 1 value, and the intermediary colors are mapped to values evenly distributed between the two. Then, the points in the [0,1] range that do not exactly match one of these color locations are computed by linearly interpolating each RGB value between the two closest colors. The leftmost RGB graphs shown in Figure 5.1 represent this methodology. The color sequence undergoes further manipulation to grant it some desirable properties. We now detail how the final colormap is achieved. The several steps of this process are illustrated in Figure 5.1.

In this work, a random colormap will have between two and four distinct colors, n_{colors}. This amount can be provided to the colormap generator function as an argument, otherwise, it is determined by chance. After deciding on the number of colors, random values in the [0,1] range are gathered to form a matrix with the shape $n_{colors} \times 3$, where each row is a different color and the columns are amounts of Red, Green, and Blue. After getting the initial random RGB values, with probability 30%, one of the colors is changed to either black or white (with equal probability). This is done to assure that less colorful colormaps are created.

Then, the colors in the color sequence are repeated. With equal probability, either the colors are mirrored at the end of the original sequence, or the colors are repeated in the same order as the original

[1] https://matplotlib.org/

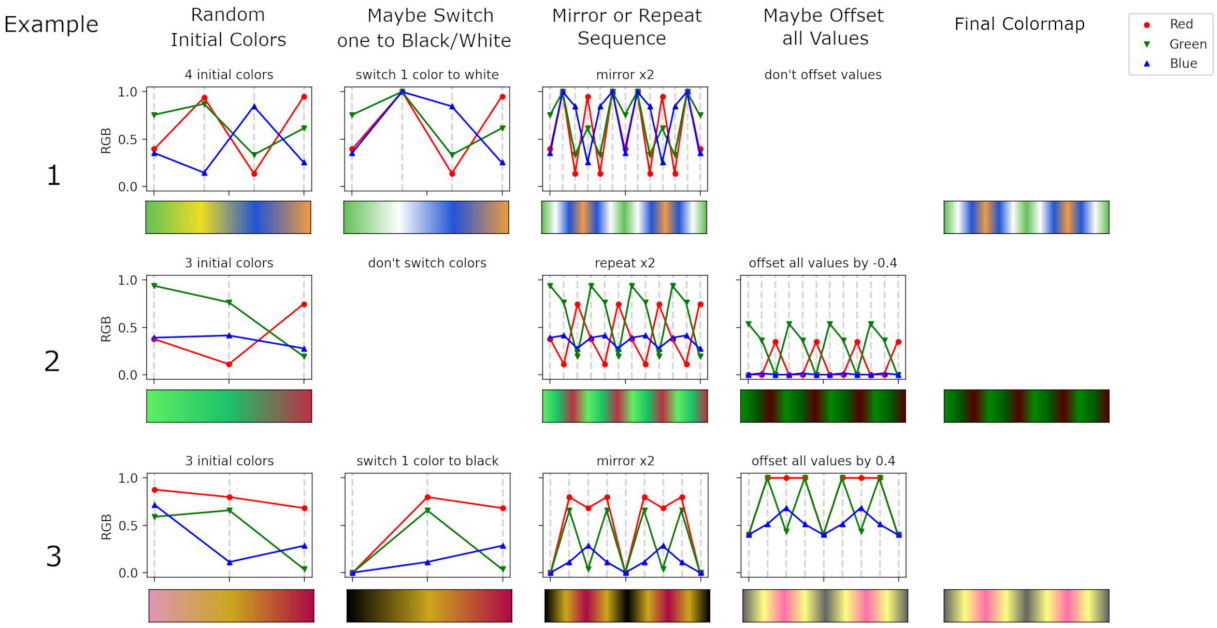


Figure 5.1: Demonstration, using three examples, of how the colormaps are achieved, by following four steps: acquire a list of $n_{colors} \in \{2, 3, 4\}$ random colors; with 30% probability, switch one of the colors to black or white with equal likelihood; with equal probability, repeat or mirror the color sequence two times; with equal likelihood, all values are offset by -0.4, 0, or +0.4 (and adjusted back to the [0,1] range if needed).

sequence. The same operation is performed twice. This is done to increase the probability of having a lot of variance in the drawings, since now the colormap changes color at a higher rate than before.

The next step will either keep the colormap unchanged, or shift all the Red, Green, and Blue values up by 0.4, or down by 0.4, with equal probability. The new values are corrected to stay in the [0,1] range. This step aims to generate overall brighter and darker colormaps. The final color sequence is then used to create a new `matplotlib` colormap by using the `LinearSegmentedColormap` method. Examples of colormaps generated using this method are shown in Figure 5.2.

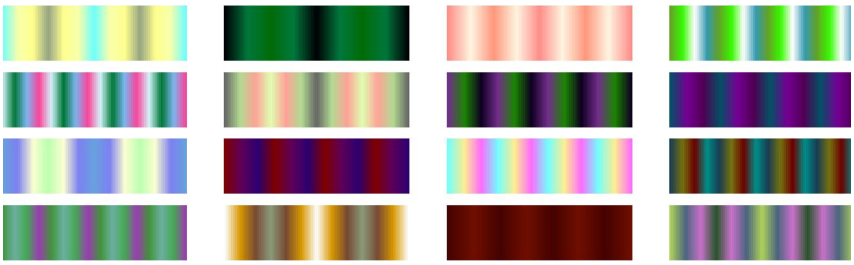


Figure 5.2: Examples of randomly generated colormaps.

5.2 Function Generation

We now turn our attention to the construction of a function that takes each pixel position and frame index and returns a value in the [0,1] range, inspired by the seminal work of Sims [21] and similar systems. We achieve this by organizing in a tree-like format a few nodes that represent either functions, variables, or constants. The final result constitutes the expression tree that determines what is drawn in each frame. The root node outputs values in the [0,1] range that are then mapped into colors. Since this node is a function with N arguments, then it must have N children nodes, which can be functions, variables or constants, and so on. The terminals (leaves) of the tree must be variables or constants, and the non-terminal nodes (interior nodes) must be functions.

To know which color should be shown at each pixel location for every frame, the function is parsed starting from the terminal nodes, then their parent nodes are computed, until the root is reached. The root outputs a floating-point value which can be any number between $-\infty$ and $+\infty$. To make sure that we have a consistent mapping between any output and a color from the colormap, we perform a transformation of the output of the root into the [0,1] space as follows:

$$output' = \frac{1}{2} \cdot \frac{output}{1 + |output|} + \frac{1}{2}, \tag{5.1}$$

where $output$ represents be the original output of the root node, and $output'$ is the transformation into the [0,1] range. In this way, $output = -\infty$ is mapped to $output' = 0$, $output = 0$ is mapped to $output' = 0.5$, and $output = +\infty$ is mapped to $output' = 1$. To deal with functions with restricted domains, all invalid values from the final output are corrected to 0, i.e., `NaN` $\rightarrow$ 0.

Now we just check which RGB value this new $output'$ corresponds to according to the colormap provided, and we can determine every pixel for every frame. Figure 5.3 shows a few trees and the frames obtained from parsing the corresponding functions. Figure 5.4 shows a few example frames, generated using more complex trees.

In our implementation, we treat nodes as objects of the class `Node`, and there are classes that inherit from other classes to reduce code redundancy. Every node has a `parse` method, which can be different depending on the type of node, and can be dependent on the instance's attributes, defined randomly upon instantiation. We now specify the different classes and how they are parsed.

The possible terminal nodes are as follows. A constant value, `Const`, is determined when assembling the tree and remains unchanged for all pixel locations and frames, and its value is determined by sampling from a normal distribution ($\mu = 0, \sigma = 3$). There are variables related to the location of the pixel, `LocVar` (local variables). The `x` and `y` variables will have values raging from -2 to +2 from left to right in the case of `x`, and from bottom to top in the case of `y`, and the `r` variable is the radial distance to the center of the canvas as determined by $r = \sqrt{x^2 + y^2}$.

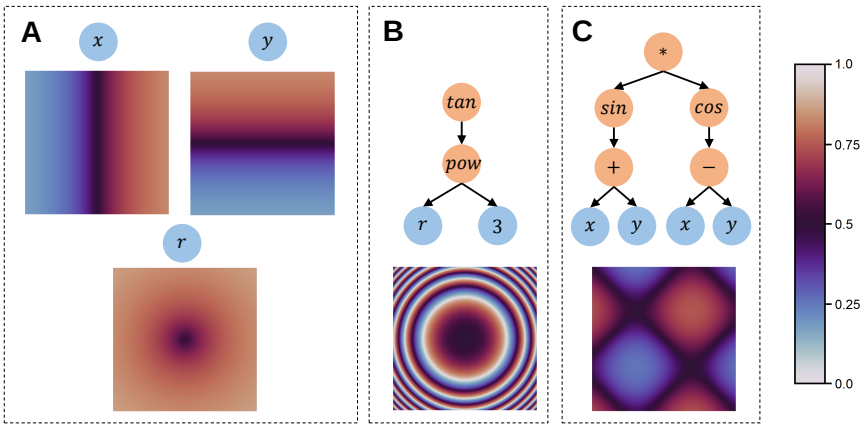
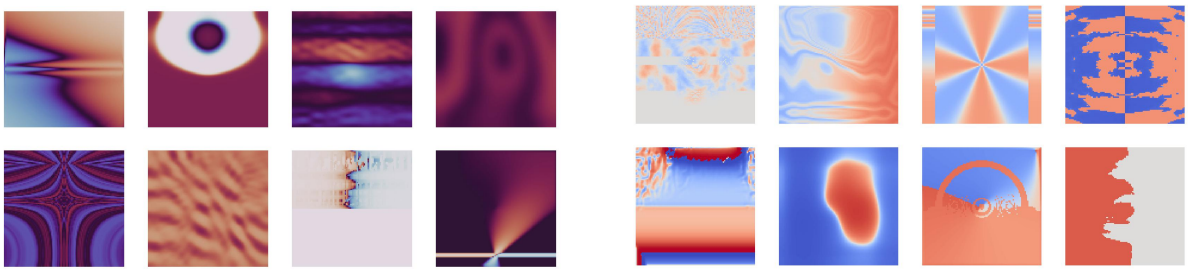


Figure 5.3: Examples of function trees and the resulting frames, obtained using the `twilight` colormap. All the values were transformed through Equation (5.1). (A) The base local variables that the system can use as terminal nodes: $x \in [-2, +2]$ (linearly varying from left (-2) to right (+2)); $y \in [-2, +2]$ (linearly varying from bottom (-2) to top (+2)); $r = \sqrt{x^2 + y^2}$. (B) The frame obtained from $tan(r^3)$. (C) The function $sin(x + y) \cdot cos(x - y)$.



(a) `twilight` colormap

(b) `coolwarm` colormap

Figure 5.4: Examples of frames generated by random assembly of tree functions: Figure 5.4(a) used the `twilight` colormap to compute the frames, while Figure 5.4(b) used the `coolwarm` colormap.

The variables that change with time are regarded as `TempVar`, and can be directly obtained from the Subband Intensities (Equation (4.1)) associated with the frame being drawn, or can be sinusoidal functions with a periodicity of 0.5, 1, or 2 times the Average Tempo (Equation (4.3)) of the audio. The `LinTime` node corresponds to a linear function of time, $a \cdot t \cdot dt + b$, where t indicates the frame index, $dt = spf$ is the time elapsed in seconds between frames, and a and b are determined by a normal distribution ($\mu = 0, \sigma = 6$).

Another category of terminal nodes, `Pattern`, corresponds to a set of built-in pattern generators, and its purpose is to increase the likelihood of the final drawing displaying enough local variation to be considered interesting. These patterns include: 2D Perlin Noise with varying arguments, `NoisePattern`; angle and radius dependent patterns such as spirals and flower-like textures, `AnglePattern`; and kernel generated patterns mcodeKernelPattern. The `Pattern` nodes are treated as terminals because the val-

ues for every pixel location are computed when the function is assembled, so their values are constant.

The `AnglePattern` node is dependent on `x` and `y`, with `r` being computed from these two; this method is also used by the non-terminal node `AngleFunc`, a binary function, where the values of its two child nodes are used instead of `x` and `y`[2].

The kernel generated patterns are formed by taking a canvas with random values and performing a convolution on it with a kernel with random entries several times, `KernelPattern` (this method is also used by the internal node `Kernel` by performing the convolutions over the output of another node instead of a canvas with random values, so it is considered a unitary function).

The non-terminal nodes represent a function randomly chosen from a function set, and require one or more arguments, up to four. We will describe the functions by using arbitrary arguments such as n_i, which can represent any node. Each argument they receive can correspond to a floating-point value or a two-dimensional matrix. Some functions are adapted to guarantee that the function is well defined for all values, while in other cases, when the arguments fall outside the domain of the function, zero is returned. There are multiple possible types of non-terminal nodes. `StdFunc` corresponds to a set of functions that treat each pixel location and frame index independently. Other non-terminal nodes include the `Kernel` node, `AngleFunc`, and `Diffusion`.

The unitary function set of `StdFunc` includes trigonometric functions ($sin(n_1)$, $cos(n_1)$, $tan(n_1)$, $arctan(n_1)$), exponential (e^{n_1}), absolute value ($|n_1|$), adapted square root ($\sqrt{|n_1|}$), the power of two (n_1^2), natural logarithm ($ln(n_1)$), adapted natural logarithm ($ln(|n_1| + 10^{-6})$), positive sine ($0.5sin(n_1) + 0.5$), positive cosine ($0.5cos(n_1) + 0.5$), 1D Perlin Noise [47], blur (blur n_1 using a normalized box filter of size 15×15), and Prewitt [48].

Besides nodes of type `StdFunc`, the other unitary function nodes are a multiple passage kernel convolution, `Kernel`, and a diffusion operator, `Diffusion`, inspired by reaction-diffusion systems [49].

Regarding the `Diffusion` node, the new value in a pixel location at a given frame is dependent on the previous frame and on the surrounding pixels (using a constant D_{const} that is obtained from a normal distribution ($\mu = 0, \sigma = 10$) and the kernel $D = D_{const} \cdot [[0.05, 0.2, 0.05], [0.2, -1, 0.2], [0.05, 0.2, 0.05]]$, the output of the node, new_output, is dependent on its output on the previous frame, old_output, and on the current output of the child node, $child$, according to $new\_output = old\_output + spf \cdot (child + old\_output \otimes D)$).

The binary functions include basic arithmetic functions ($n_1 + n_2$, $n_1 - n_2$, $n_1 * n_2$, n_1/n_2, complex arcsin and arccos (the real value of the complex functions $arcsin(n_1 + in_2)$ and $arccos(n_1 + in_2)$), the $arctan$ of the angle formed by (n_1, n_2), adapted power ($sign(n_1) \cdot |n_1|^{n_2}$), distance ($\sqrt{n_1^2 + n_2^2}$), 2D Perlin Noise [47], minimum between two arguments ($min(\{n_1, n_2\})$), blend ($(n_1 + n_2)/2$), and the function that

[2] It can be just a pattern of centered circles using $cos(\alpha \cdot r)$; spirals are created using $cos(\alpha \cdot \pi \cdot r + arctan(x, y))$; flower patters are generated using $cos(\alpha \cdot arctan(x, y))$; a decay function, $1/(1 + exp(signal \cdot \alpha(r - \beta)))$, can be multiplied to the output, where $signal \in \{-1, +1\}$ determines if the decay is towards the center or outwards; symmetric variations of x and y and switching them are possibilities, and all the constants are sampled from uniform probability distributions defined empirically.

generates angle patterns but with n_1 and n_2 instead of `x` and `y`, `AngleFunc`.

The ternary functions correspond to smoothclamp (let $x_{aux\_1} = (n_1 - n_2)/(n_3 - n_2)$; if $x_{aux\_1} < 0$, then $x_{aux\_2} = 0$; if $0 \leq x_{aux\_1} \leq 1$, then $x_{aux\_2} = 3x_{aux\_1}^2 - 2x_{aux\_1}^3$; if $1 < x_{aux\_1}$, then $x_{aux\_2} = 1$; the output of smoothclamp is given by $n_2 + (n_3 - n_2) \cdot x_{aux\_2}$), lerp ($(1 - n_1) \cdot n_2 + n_1 \cdot n_3$), 3D Perlin Noise [47], and image warping operations (warping of the input n_1 using spline interpolation of order 3, where n_2 and n_3 dictate the transformation of the coordinates). Finally, the only quaternary function is given by the following conditional function: if $n_1 > n_2$, then n_3, otherwise n_4.

A new tree is generated by randomly assembling the possible nodes, following some rules. The function tree is assembled from the root to its terminals, such that a random interior node (non-terminal) is first chosen to be the root node (level 0), and its children (arguments) are then randomly chosen as well. After choosing each new node, its children are selected (their level is the level of the parent increased by 1), and so on, until there are no more functions with unspecified arguments - all the nodes with no children are leaf nodes (terminal). To force the process to finish, if the level of a node is 11, then it must be a terminal node.

The probabilities of selecting each type of node are specified by Figure 5.5, and the process for choosing a new `LocVar` and `TempVar` is schematized in Figure 5.6.

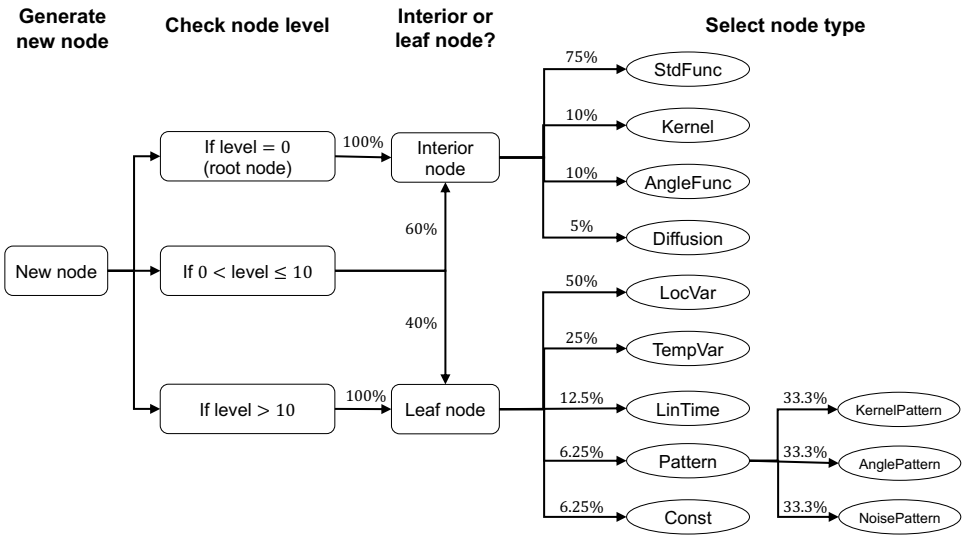


Figure 5.5: Diagram of the process for selecting the node type, with the indication of the conditions required and the probabilities for each node.

5.3 Animation Parsing

Each animation frame corresponds to a time window of $spf = 0.08$ seconds, so the extracted audio features, which also contain information referring to a time window of $spf = 0.08$ seconds, can be

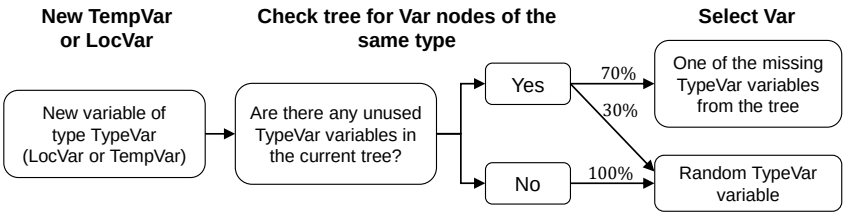


Figure 5.6: Diagram of the process for selecting a new variable, which is the same for a new node of type LocVar or TempVar, but the chosen variables can only be of the desired type.

directly fetched for each frame. This results in a frame rate of 12.5 frames per second.

The process starts by calling the `parse` method of the root node, which works by calling the `parse` method of its child nodes, and only then applying the node function to the arguments. This process continues until a leaf node is reached. The leaf nodes return the corresponding variable, constant, or pattern in matrix form, which are then used as the function arguments of their parent nodes.

In practice, the functions take matrices as arguments, and return a matrix as output, and the output of each node is passed along to its parent. This process continues until the output of the root node is computed, which is transformed into the [0,1] range using Equation (5.1).

For the time-dependent nodes of the function tree, the system fetches the required features (i.e., the Subband Intensities) that match the timestamp of the frame. A similar process is performed for the sinusoidal variables and the linear time variation variable. Figure 5.7 shows a few examples of time-dependent animations.

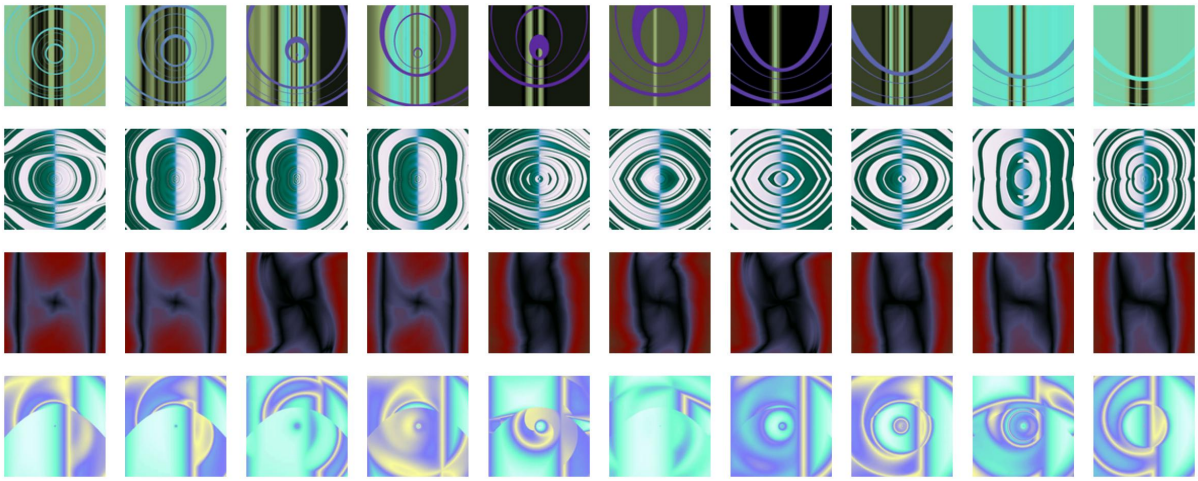


Figure 5.7: Examples of animations resulting from our system. Each row uses a different function and colormap, and each column represents a frame of the sequence, where time progresses from left to right. Every other frame of the animations was skipped to emphasize their time evolution.

5.4 Animation Features Extraction

For analyzing the animations, we based the features being extracted on the work of Li et al. [41], since the features they consider were shown to be useful to predict the user's aesthetic preference regarding images. We also extracted additional features to capture time-dependent aspects of the animation. For instance, we evaluate how much the colors change between frames, on average, following [24].

We transform information about frames into animation features by computing their average over the animation. All the features are computed on frames with 50×50 resolution, independently of what the final video resolution will be, to ensure consistency of results. An overview of the extracted features is shown in Table 5.1.

Table 5.1: Animation features (Feature) that are extracted by the system, and their definitions (Description). Count indicates the number of different features a description corresponds to. Includes information about where the features are used in our system: Function Filtering (FF), Mood Matching (MM), or Preference Model (PM). Some features are adapted from [41] or [24], as indicated by the last column.

Feature	Description	Count	FF	MM	PM	From
Color moments	Three central moments of hue, saturation and lightness	9			X	[41]
Lightness feature	Lightness channel based on Benford's Law	1			X	
Texture feature	Local binary patterns (LBP) in four ranges	8			X	
Image complexity	Information entropy of HSL, RGB and Y709	5			X	
Image order	Low complexity based on fractal compressor	1			X	
MC metric	The image complexity (IC) and processing complexity (PC) ratio	1			X	
Local Variation	Mean and standard deviation over the animation of loc_var, which is computed as $Prewitt(Grayscale(frame))$	2	X	X	X	-
Local Variation Percentage	Percentage of frames in the animation for which the condition $0.02 < loc\_var$ applies	1	X			-
Temporal Variation	Mean and standard deviation of $temp\_var$, which is the average distance in the RGB space between pixel colors of consecutive frames in the same positions	2	X	X	X	[24]
Temporal Variation Percentage	Percentage of frame transitions where $0.005 < temp\_var$	1	X			-
Movement Correlation	Quantification of correlation between $temp\_var$ and each Subband Intensity	7			X	-

As indicated by column MM for Mood Matching and column PM for the Preference Model in Table 5.1, some features are used by these methods to analyze animations. This process is further detailed in the following sections.

5.5 Function Filtering

A few constraints were enforced on the functions generated to consider them valid. Most constraints aim to increase the likelihood of producing interesting animations, i.e., to avoid a large number of frames having similar colors for all pixels in the canvas, or animations perceived as almost static. Inversely, we also attempt to rule out animations that look like pure noise, i.e., random colors in every pixel. We also avoid functions that take too long to parse.

Some of these are evaluated on the function tree alone, and some are dependent on the resulting animation (according to the colormap). Function Filtering is the reason why we need the colormap and section features to get a new function. After a function tree is assembled, it is tested for the tree constraints. If these are fulfilled, then the animation constraints are evaluated. If any of the constraints is not respected, then the function tree is rejected and a new one is assembled. This process continues until a valid function is acquired.

The function tree constraints are as follows:

- at least one time-dependent node (`TimeVar`, `LinTime` or `Diffusion`);
- at least two `LocVar` nodes, or at least one `Pattern node`;
- the time elapsed when parsing the function tree to compute one frame with 50×50 resolution (the first timestamp of the section is used) should be less than 0.1 seconds.

The constraints that depend on the animation features are:

- $0.05 \leq 3 \cdot mean\_Local\_Variation \leq 0.9$;
- $0.05 \leq 2 \cdot mean\_Temporal\_Variation \leq 0.9$;
- $0.5 \leq Local\_Variation\_Percentage$;
- $0.5 \leq Temporal\_Variation\_Percentage$.

Chapter 6

Animation Analysis

In this chapter, we consider the animation analysis methods individually, detailing how each one selects the colormaps and animation functions. Section 6.1 describes the Mood Matching method, Section 6.2 the Preference Model, and Section 6.3 explains Manual Selection.

6.1 Mood Matching

The Mood Matching approach was the first method we developed for animation selection. The idea behind it is to estimate which animation best matches the mood of the audio, given a set of animations. For this purpose, we take the arousal and valence of the audio estimated from the Mood Model, and try to choose a colormap related to this mood, and then an animation with an amount of detail and movement that also match this mood. We now describe in more detail these methods of colormap and animation selection.

6.1.1 Colormap Selection

To capture the arousal of the audio in the colormap, we associated the use of more colors to higher arousal, and less colors to lower arousal. Equation (6.1) denotes the rules used to decide how many distinct colors the colormap should use:

$$n_{colors}(Audio\_Arousal) = \begin{cases} 2, & \text{for } Audio\_Arousal < 1/3, \\ 3, & \text{for } 1/3 \leq Audio\_Arousal < 2/3, \\ 4, & \text{otherwise}, \end{cases} \tag{6.1}$$

where $Audio\_Arousal$ indicates the estimated arousal of the audio obtained from the Mood Model. The computed n_{colors} variable is provided to the colormap generator, which creates several colormaps to be evaluated based on valence in the next step.

In order to choose the best colormap from a set of randomly generated colormaps, we developed a function that associates a valence value with every color in the HSL space. Using this method, and by

representing each colormap with 50 color points, we can compute the average valence of the colormap according to this model.

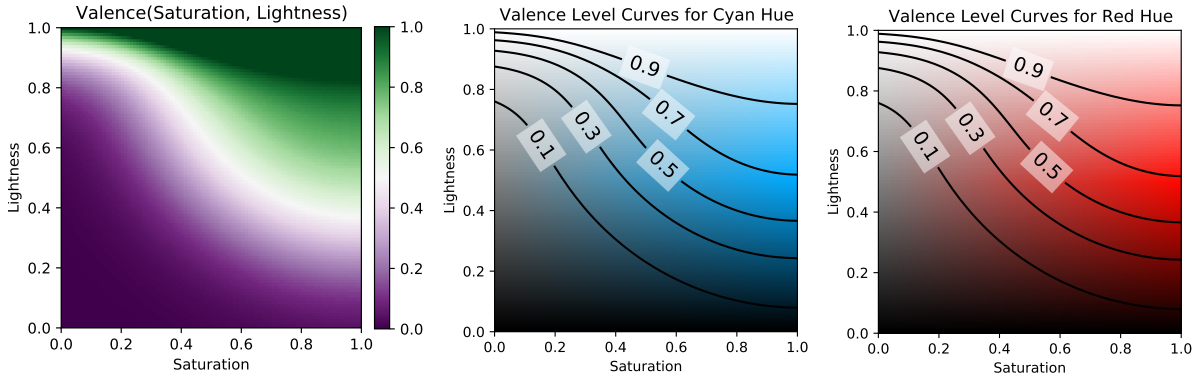
Although studies suggest that people associate certain Hues with different emotions, they also indicate that the meaning of a color is mostly associated with its Lightness and Saturation, and less influenced by the Hue [50, 51]. Therefore, to provide a more general mood estimation, our mood analysis of colormaps neglects the Hue.

To capture the valence of a color, we looked for a function that would provide higher valence for brighter colors and lower valence for darker colors, but at the same time provide a low valence for gray colors and higher for more saturated ones. At the same time, saturated colors should not have a valence as high as the color white. We tested if different functions captured this idea by visually examining the level curves that were formed, as exemplified in Figure 6.1. The function that we believe best captured this idea is:

$$Valence(Sat, Lig) = \min(\{1, Lig^{10} + 1 - sigm((1 - Sat)^2 + (1 - Lig)^2)\}), \tag{6.2}$$

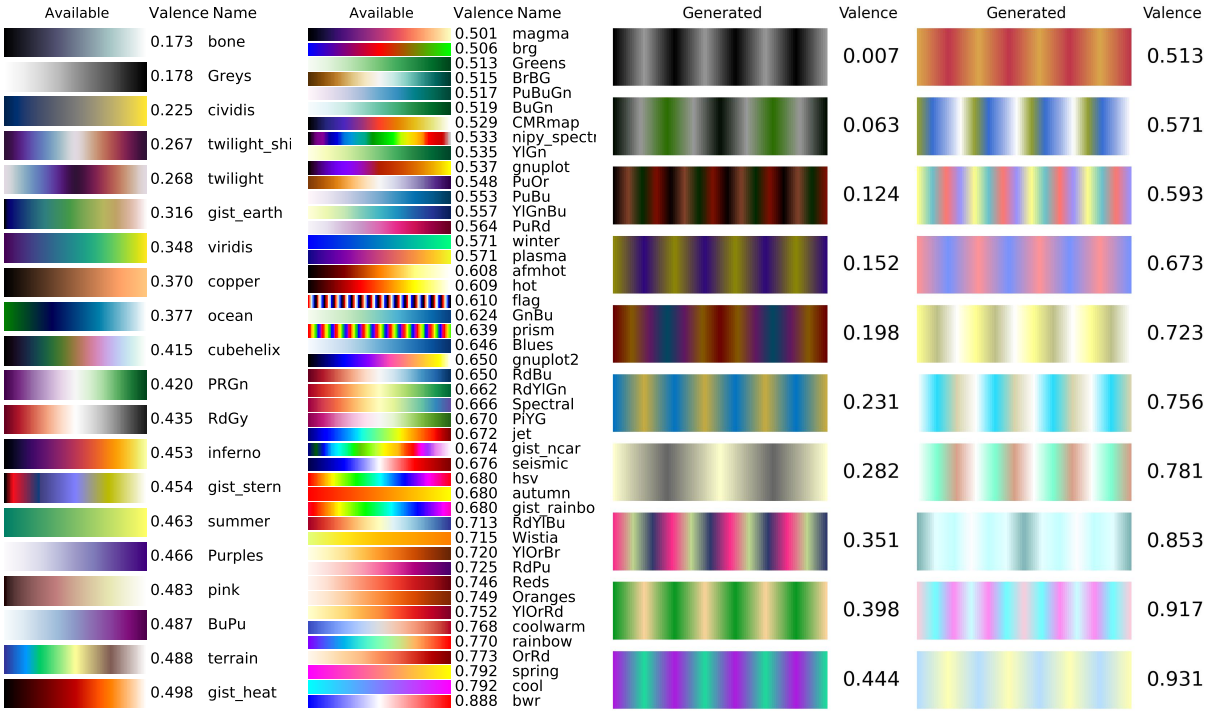
where $Sat, Lig \in [0, 1]$ represent the Saturation and Lightness values, respectively, and $sigm(\cdot)$ is the sigmoid function.

The final valence of a colormap corresponds to its average color valence, computed through the discretization of the colormap into 50 colors. Valence estimation for different colormaps is exemplified in Figure 6.2. In this way, for any color represented in the HSL space, the valence will depend solely on the Lightness and Saturation values.



(a) Valence mapping **(b)** Cyan Hue **(c)** Red Hue

Figure 6.1: Representation of valence computation based on the Saturation and Lightness values, according to Equation (6.2) (this computation is independent of Hue): (a) visualization of the mapping between Saturation and Lightness and Valence; (b) valence level curves for the cyan Hue; (c) valence level curves for the red Hue.



(a) Available, $val < 0.5$ **(b)** Available, $val > 0.5$ **(c)** Generated, $val < 0.5$ **(d)** Generated, $val > 0.5$

Figure 6.2: Examples of colormaps and their valence values. Colormaps available from `matplotlib` with the indication of their names: (a) valence below 0.5; (b) valence above 0.5. Colormaps generated using the method developed in this work: (c) valence below 0.5; (d) valence above 0.5.

6.1.2 Animation Selection

The next step corresponds to selecting an animation from a set of random animations obtained using the previously selected colormaps. To compare each animation with the audio mood, we now only consider the arousal of the audio. It is our understanding that, for higher arousal, an animation should have more edges and abundant movement, while lower arousal is better represented by frames with fewer edges and slower movements.

For every sample animation, the Mean Local Variation and Mean Temporal Variation are compared to the audio arousal in the following manner:

$$\begin{aligned} Local\_Variation\_distance &= |Audio\_Arousal - 3 \cdot mean\_Local\_Variation|, \\ Temporal\_Variation\_distance &= |Audio\_Arousal - 2 \cdot mean\_Temporal\_Variation|. \end{aligned} \tag{6.3}$$

The constants 3 and 2 that rescale the mean variations were determined empirically, and were imposed to better capture what we considered a good correspondence between the arousal and these

values. Then, the arousal of the animation is considered to have a distance to the arousal of the audio, $arousal\_distance$, according to

$$arousal\_distance = max(\{Local\_Variation\_distance, Temporal\_Variation\_distance\}). \tag{6.4}$$

We opted for this metric to find the animation where both measures would match the arousal, and avoided using an average of the distances because this resulted in the selection of animations in which one of the distances was small, but the other could be large. By using the maximum distance, we could minimize both distances.

Using this comparison method, the generated animation with the lowest $arousal\_distance$ out of the sample set is considered the one which best matches the mood of the audio, and is selected to represent the current section.

Note that, although the average audio arousal of the function group is used to analyze the function, only the animation for the first section of the group is analyzed, for simplicity. In other words, the remaining animations that will use the final function of the group are not analyzed. This can result in less desirable animations for these sections, as they use different colormaps and can exhibit a different animation due to using different audio frames. An analysis that takes all the sections of the function group into account could be implemented to improve this method.

6.2 Preference Model

Although the Mood Matching approach provides interesting results in terms of colormap and animation choices, we looked for a way of including the aesthetic preference of the user when evaluating candidate animations. To do this, we avoided using hard-coded rules to keep the method user-independent, and instead developed a dataset we could use to get the preferential taste of the user, and also assembled a model selection method to train and select the best model based on the user's input. The determination of the animation features to be used by the Preference Model was based on the work of [41] and [24].

6.2.1 Features and Output

To build a model that would provide an animation choice that tries to estimate the user's preference, given an audio and a set of animations, we pondered on what would be the best features the model should receive and what output it should provide. Because it is important that the animations take into account the audio being produced, the model should receive both audio and animation features. The features used are indicated in Tables 4.1 and 5.1 by their respective *PM* columns. The estimated arousal

and valence of the audio are also considered. In this way, information regarding the audio and visual elements obtained by our system can be taken into account.

Regarding the audio features, because some of them have values for every frame, they need to be further processed to represent an entire audio section. To achieve this, the mean and standard deviation of the features that have a Span of Frame as indicated by Table 4.1 are used to represent the audio in question, while the features that have a Span of Entire Audio are used without further processing. Similarly, the arousal and valence of the audio obtained from the Mood Model are concatenated without changes being required.

As mentioned in Section 5.4, because the animation features are frame-dependent, they need to be averaged over all the frames to represent the entire animation.

Thus, an audio-animation pair is represented by: primary audio features, mood features (arousal and valence), and animation features.

In this work, we attempted to achieve a model that could return a confidence score for how much a user would enjoy a particular audio-animation pair. Therefore, the final model should return the fitness of every audio-animation pair, so the pair with the highest fitness can be selected, as this is interpreted as the pair with the highest confidence of being preferred by the user.

To achieve this goal, the model needs to be trained based on the user's preference. A dataset was assembled with various audio-animation pairs, for which the user provides feedback regarding which ones are preferred.

6.2.2 Audio Dataset

To train a model with this purpose, a variety of 10-second audios were obtained from the DEAM dataset [42], and their audio features were computed. The 10-second audio duration was chosen to assure the usability of the dataset, since the audios should be short to minimize user fatigue during training, but should also be long enough so the user can get a good representation of the audio and the generated animations.

To choose which audios to use, we analyzed the annotations provided by the DEAM dataset regarding the music genres each audio is associated with. For all the 1744 audios provided by the DEAM dataset, the fourteen considered genres and their occurrence count are indicated in Table 6.1. It should be noted that the genres are not mutually exclusive, and several audios were annotated with more than one genre simultaneously.

To choose a diverse enough set of audios, we randomly chose for each genre four different audios independently, and then kept track of the additional genres these audios were associated with. To exemplify this process, let's consider the Pop genre. The first step would be to randomly pick four tracks that have Pop annotated as one of their tags. Then, if two of the tracks are only associated with Pop,

Table 6.1: The occurrence count of every genre as indicated in the DEAM dataset, and the occurrence of the genres in the audio dataset used to train the Preference Model. Since an audio can have multiple genres annotated at the same time, the occurrence count sum does not equal the number of audios. The genres are: Blues, Classical, Country, Electronic, Experimental, Folk, Hip-Hop, Instrumental, International, Jazz, Pop, Rock, Soul/Rhythm&Blues, Spoken.

Dataset	Blues	Classical	Country	Electro.	Experim.	Folk	Hip-Hop	Instrum.	Intern.	Jazz	Pop	Rock	Soul/R&B	Spoken
DEAM	165	216	213	369	159	296	106	25	118	219	257	429	104	8
Preference	6	10	7	9	11	8	4	7	8	6	6	13	6	5

but one of them is also tagged International, and another one is also tagged Instrumental, then we end up with four audios, but an occurrence count of 4 Pop tags, 1 International tag, and 1 Instrumental tag. For the next genre, we would repeat the same process regardless of the current occurrence count. This means that we probably end up with an occurrence count of more than four for every genre, but with at least four audios for every tag. The final audio dataset for the preference model training corresponds to 55 distinct audio samples, and the genre occurrence count is represented in Table 6.1, along with the original count regarding the DEAM dataset.

6.2.3 Animation Dataset

The next step corresponds to generating sample animations for all the tracks. For each 10 second audio independently, 15 different animation samples were generated, each one with a randomly obtained colormap (using the method described in Section 5.1) and a randomly obtained function (using the method described in Section 5.2), each sample with a resolution of 100×100 pixels. The final animation dataset corresponds to 825 unique animations (55 audios $\times$ 15 animations). All the features are computed and normalized to have a mean of zero and standard deviation of 1, and their normalization coefficients are saved for use in future animations.

6.2.4 User Input

The audio-animation dataset should also contain the target fitness for each pair so the model can be trained according to the user's preference. Although the model will output a fitness value in floating-point format for every audio-animation pair, it is easier for the user to provide a classification instead. Therefore, we ask the user to classify each animation individually, which improves the usability of this approach.

To facilitate user input regarding the preference of each audio-animation pair, a user interface was developed, using the standard Python library `tkinter`[1]. For every unique audio from the audio dataset, its 15 corresponding animations are shown simultaneously, being animated synchronously with the audio. Then, the user indicates, for its 15 animations, which ones they prefer being used for that audio

[1] https://docs.python.org/3/library/tkinter.html

(class +1), which ones they feel neutral about (class 0), and which ones they do not like to see with that audio (class -1). The user should take into account how much they like an animation, but at the same time how much they think the audio and animation match.

After going through all the animations for all the audios, every audio-animation pair has a target fitness associated with it that can be used to train the Preference Model. Figure 6.3 illustrates the interface used to collect the user's classifications, developed using `tkinter`.

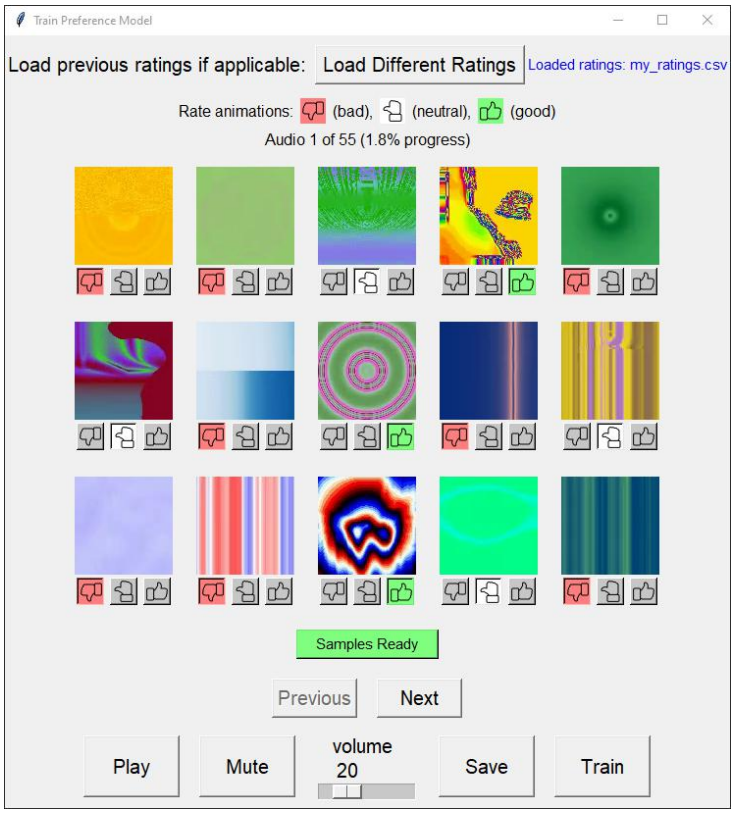


Figure 6.3: Example usage of the interface to classify each audio-animation pair depending on the user's preference for each audio. The user can analyze all the animations for a single audio at a time, and has access to play/pause and audio controls. All classes are initially set to the neutral class, and the *Samples Ready* button is in red, and it says *Samples Not Ready*. After the user changes to the desired classifications and changes the *Samples Ready* button by clicking it, they can press the *Next* button and analyze the animations related to the next audio. The user can load previous classifications and save their classification progress at any stage. When pleased with the final results, they can train the Preference Model with their classifications and save the model in the desired location.

6.2.5 Training and Model Selection

After computing the audio-animation features and with access to the classifications provided by the user, we now have all the required information to train the Preference Model. A systematic study was performed to determine which model and parameters best capture our preference regarding audio-animation pairs. This process can be repeated to train and select a model that best captures another

set of classifications, so that a new model can be obtained for a different user. We now describe the aspects contemplated in model selection.

We considered whether the fitness estimation should be obtained directly from the output of a regression model, or if we could get better results by training a classifier and using as fitness the confidence of the classifier on the class being +1. The idea of using a classifier stems from the fact that our dataset is composed of classifications, and not continuous target values.

All the following models were evaluated using their classifier and regression versions separately, using the `sklearn` Python library with the default arguments unless otherwise stated: K-Nearest Neighbors (KNN) (considering 5 neighbors), Decision Tree (using an entropy criterion and a minimum of 2 samples per leaf), Support Vector Machines (using an rbf kernel with coefficients scaled by $1/(n_{features} \cdot var_{feats})$, where var_{feats} is the variance over all the features), and Multilayer Perceptron (with one hidden layer containing 100 neurons). An overview of each model can be found in Appendix A.3.

Another thing that we tested was the usage of binary classes (+1 for a good match and 0 for both neutral and bad matches) instead of using the three original classes. The idea behind this variation is to test whether more granularity during training improves the final results or not. Note that, when the model in question is a classification one and the three classes are considered for training, the fitness output is still only dependent on the confidence of the class being +1, and this process stays the same when using binary classes.

One of the aspects that could affect the model's ability to learn is the number of features considered. Since it is likely that not all features play key roles in predicting fitness, feature selection was considered to different extents. Either no feature selection was performed, or only the features deemed more influential were used for training. Different amounts of features were tested: 30, 50, 70, 90, or all the features. The top features are chosen by checking which ones have the highest correlation with the outputs, in absolute terms.

Because the dataset uses the same audio for several audio-animation pairs, we considered whether the split between training, validation, and test sets should be performed on an individual animation level, where each audio-animation pair is assigned independently, or at the level of the audio, in which case the audios are assigned to either training, validation, or test, and this decides the assignments of the audio-animation pairs. The dataset split was always performed in the proportions 75%, 15%, and 10% for training, validation, and test sets, respectively.

We considered the option of oversampling the lower frequency classes. In fact, when we trained the model using our preference, the animations were not likely to be classified as +1, as most of the time only between 1 and 3 animations out of 15 were classified as so for each audio. After performing a systematic analysis by keeping all parameters constant except for the use of oversampling, the difference was only visible after the fourth decimal place of the RMSE and, for this reason, we decided to not resort to

Table 6.2: The parameters that provided the best results according to different metrics, and their average metrics from 30 runs. The final parameters correspond to the ones that provided the highest average precision, and are highlighted in gray. The best performance in terms of each metric is highlighted in **bold**.

model	Type	Model	N Classes	Feat. Sel.	Split By	RMSE	Acc	Prec	Rec	F1
min RMSE, max Acc	Class	SVM	3	All	Audio	**0.380**	**0.817**	0.750	0.450	0.563
max Prec	Reg	SVM	3	All	Audio	0.682	0.765	**0.833**	0.125	0.217
max Rec	Reg	KNN	2	70	Audio	0.496	0.694	0.670	**0.767**	0.715
max F1	Class	SVM	2	50	Audio	0.419	0.762	0.753	0.744	**0.748**

oversampling. The analysis presented henceforth ignores this possible variation.

In total, the variations considered are: type of model (classification or regression), model name (KNN, DT, SVM or MLP), number of classes (three or two, by changing the -1 classifications into the 0 class), feature selection (use 30, 50, 70, 90, or all features), and the method used to split the dataset (by animation or by audio). Table 6.2 shows the model variations that obtained the highest average validation metrics, regarding 30 different runs for each of the 320 combinations of parameters considered. Notice that, regarding the metric RMSE, the values tend to be larger when considering 3 classes instead of 2, due to the domain difference.

Although we are interested in using a model that provides as output a continuous fitness value for every audio-animation pair, we analyze all models in terms of both regression and classification metrics, with the purpose of getting more insight regarding the performance of the models. For regression, the metric RMSE is used, while the classification outputs are analyzed using accuracy (Acc), precision (Prec), recall (Rec), and F1. To do this, we need to transform the regression outputs into classification outputs, and vice versa. A regression output is transformed into a classification by checking if the value is below 0.5, in which case the predicted class is considered to be 0, and otherwise it is considered +1. On the other hand, a classification model can be transformed into a regression one by taking the confidence of the model that the class is +1 as the regression value. In this way, all the metrics can be computed. When using the model for computing fitness, only the regression version of the output is used. An overview of the metric definitions can be found in Appendix A.2.

Table 6.2 reveals some insights regarding the performance of different parameters. One thing that is clear is that the train-valid-test split of the dataset by audio instead of audio-animation pair provides better validation results, as all the best models use this parameter. Then, we see that the use of an SVM model, whether with regression or classification, results in better performance in almost every metric, except for recall, for which the best model is the regression version of KNN. In addition, the fewer features are considered in the model in terms of feature selection, the better the F1 score, which means a better balance between precision and recall. Notice also how the lowest RMSE parameters also correspond to the highest accuracy. This makes sense, as both are metrics of the overall performance of the models per sample. The usage of two classes seems to lead to higher F1, compared to the usage

of three classes.

The classification version of the SVM model seems to lead to the best results in terms of metrics that provide an overall analysis of the performance of the model, such as RMSE, accuracy, and F1 score. On the other hand, the precision is improved using a regression version of the SVM model, and the best recall is achieved using a regression version of KNN.

We opted for using the parameters that resulted in the highest average precision. This is due to the purpose of the model. Since we want to estimate a fitness value for every audio-animation pair, and decide which animation to use based on the one with the highest fitness, then we want to make sure that the animations that the model considers more likely to be of class +1 belong in fact to this class. In other words, we want to minimize the number of false positives in terms of preferred animations, and the precision is the measure that represents this goal. It is not a problem if a lot of preferred animations get considered as class 0 or -1, because we only want to use only one animation per audio. Therefore, a low recall is not problematic. Similarly, a high confidence in the +1 classes is preferred over a low RMSE or high accuracy, which are metrics of the overall performance, and will tend to increase the performance of the model in terms of the more frequent class (which is not useful in this case, because class +1 is the least frequent). Finally, we do not wish to choose the parameters that lead to the highest F1 score because, in this case, we wish to prioritize the precision results instead of achieving a balance between precision and recall.

Based on the results from Table 6.2, the best model parameters when training on our personal classification of the dataset correspond to a regression version of SVM, using 3 classes and all the features, splitting the dataset by audio and without performing resampling. This analysis provided insight into the best model and model parameters, and now we need to acquire the final model that will be used to estimate the fitness of audio-animation pairs. The final model is acquired by gathering a set of 30 models that use these parameters but different seeds, and the best one is considered to be the one with the highest precision metric on the validation set. The metrics of the final Preference Model used in this work are presented in Table 6.3. It appears that the final model only provides regressions that lead to $+1$ classifications in rare cases, with 100% precision in the test set, but recall of 29.4%. However, it is very unlikely that this precision is representative of the performance of the model in our system, as this appears to be a case of overfitting to this particular dataset. This could be the case if the audios and animations selected for the test set have similar features to the ones from the training set by chance. Nevertheless, it appears that this approach successfully learns the preference taste of the user, indicating that the considered features and models are useful to emulate human aesthetic preference.

We now have the final Preference Model. Over the next sections, we will explain how the model is used.

Table 6.3: Results of the final Preference Model on the train, validation, and test sets.

data	RMSE	Acc	Prec	Rec	F1
Train	0.659	0.835	0.891	0.466	1.117
Valid	0.642	0.775	0.846	0.306	0.596
Test	0.675	0.840	1.000	0.294	0.761

6.2.6 Colormap Selection

In the context of the Preference Model approach, selecting a colormap means choosing a colormap that increases the likelihood of generating animations with higher fitness. To measure this likelihood, several steps need to be taken. Recall that a distinct colormap is used for every colormap group, which corresponds to sections clustered by mood.

Before any colormap analysis is performed, the process begins with the generation of three random animation functions, which will be used to evaluate all the colormaps for all the colormap groups. Then, for each group, the process of getting its final colormap is captured by the following steps.

The first step involves generating a random set of colormaps. In our approach, this set is constituted by the top 5 colormaps out of 250 according to the Mood Matching method of colormap selection (Section 6.1.1). This set can be seen as a filtered selection of the colormaps generated, such that the colormaps are more likely to match the mood of the audio according to the Mood Matching approach. This initial selection allows a considerable reduction of the number of colormaps that need to be analyzed, and was deployed to reduce the required processing time.

Next, for each colormap group, if the group has more than three sections, the process randomly selects the three sections that will be analyzed out of all the sections of the group. If the colormap has three sections or less, all of them are considered.

To determine which colormap is more likely to provide animations with higher fitness, we use each one to create several sample animations - one animation per colormap-function-section combination. The fitness of every animation is computed using the user preference model, and the fitness of each colormap is given by the average fitness of its produced animations. The same animation functions are used to create the sample animations for every colormap analysis to allow a direct comparison between colormaps. The computation of colormap fitness, $colormap\_fitness$, is therefore given by:

$$colormap\_fitness(cmap) = \frac{\sum_{s \in G} \sum_{f \in F} fitness(s, f, cmap)}{\sum_{s \in G} \sum_{f \in F} 1}, \tag{6.5}$$

where $cmap$ is the colormap for which we want to compute the fitness, $s \in G$ is a section s that is part of the set of sections from the colormap group G, $f \in F$ is an animation function that is part of the set

F of functions that were selected to analyze colormaps, and $fitness(\cdot)$ is the regression obtained from the model trained on user classifications of audio-animation pairs.

Because the colormap with the highest fitness produced animations with higher fitness on average using different functions, it is estimated to be the colormap that will provide animations that are more likely to be preferred by the user. Therefore, for each colormap group, the colormap with the highest fitness is selected.

6.2.7 Animation Selection

Similarly to the Mood Matching approach, the Preference Model is used to compute the fitness of every animation from a randomly acquired set generated using the same audio, in order to choose the one estimated to be enjoyed by the user with a higher confidence rate. The process is as follows: a random set of animation functions is assembled and animated, the audio-animation features are extracted, their fitness is computed by the preference estimation model, and the animation with the highest fitness is selected.

Once again, only the first section of the function group is considered to analyze the best function. This means that the method does not take into account the fitness values that would be obtained from the remaining sections of the function group. This allows for a faster analysis of the functions, but has the downside of neglecting the remaining sections. This can result in less ideal results for these sections, as they use different colormaps and can exhibit a different animation due to using different audio frames. An analysis that takes all the sections of the function group into account could be implemented to improve this method.

6.3 Manual Selection

The previous colormap and function selection methods, Mood Matching and Preference Model, allow a completely autonomous generation of new videos. In the case of the Preference Model, the user can choose one of the already trained models, or even choose to train a new one. After the user classifications are acquired for the Preference dataset, the models are trained and the best one is chosen, and the user does not need to provide new classifications to generate new videos automatically.

Besides the autonomous methods available in the system, we developed a method that allows the user to choose the samples that will be used in the final video. This approach uses the previous methods to filter some of the samples so that only the top ones are shown to the user, which can then choose their favorite. This approach can be seen as an example of co-creation.

6.3.1 Colormap Selection

In the case of colormap selection, Mood Matching is used to filter the colormaps before showing them to the user for selection. As before, one colormap is used for each colormap group of sections. The process begins with the generation of 250 colormap samples, from which the top 5 according to Mood Matching is selected to be shown to the user.

Because the user can have a specific idea of what the colors the animation should use and, therefore, be displeased with the examples provided, the user has the possibility of asking for 5 more colormaps selected from a new set of 250 examples, until they are pleased with one of them. It takes about two seconds for the system to show 5 new colormaps to the user using this mechanism. Once the user has found a colormap they are pleased with, they indicate which one they want, and then the process is repeated for the next colormap group.

We opted to use Mood Matching to filter the colormaps due to its speed of execution when compared with the Preference Model, as it easily allows the user to get more examples until they are satisfied with one of them.

Although this selection method is practical, the user lacks the knowledge of which audio sections will use these colormaps, which could greatly influence the colormaps they would choose. The audios for each colormap group could be presented to the user to improve the system.

6.3.2 Animation Selection

For animation selection, the Preference Model is used to filter the animations. Given a random set of 100 functions, each fitness is computed, and the top 15 is shown to the user through a user interface created using the `tkinter` library, ordered from highest fitness to lowest, so the user can indicate which one should be used in the final video.

Because the computation of each animation's fitness is time-consuming, the program first computes all the top 15 animations per function group, saves them in the system, and the program is terminated. This process takes approximately 3 hours, although this period depends on the number of function groups, the length of the sections in terms of frames, and the time complexity of the functions that need to be parsed. The user then comes back to the program when it is convenient, and chooses which of the animations they prefer for each function group. An example usage of the interface is displayed in Figure 6.4.

Unlike the manual colormap selection method, the user can not choose to see more samples in case none of the ones displayed seem to be a good fit. This could be improved in the future to increase the probability of finding a satisfactory match.

Another issue that we noticed was that very noisy animations were often found in the top fitness

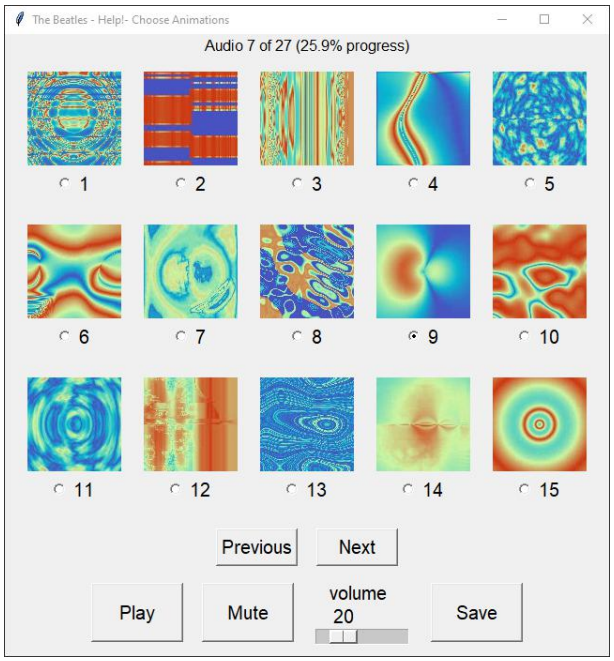


Figure 6.4: Example usage of the interface to choose the animations. Fifteen samples are shown for each function group, but the user only sees the first audio section of this group being animated, using the colormaps for this section as well. Play/pause and audio controls are available. The animations are ordered from the highest fitness (animation 1) to the lowest one (animation 15), and the default choice is the first animation. The user can then change the selected animation, and move on to the next section. Once all the preferred animations are selected, the user can press the *Save* button and, after the desired location for the final video is inserted, the interface closes and the generation of the final video starts.

selection. This can be improved by adjusting constraints that determine which functions are valid or not.

In addition, the evaluation of functions is based solely on the first section that will use that function, ignoring the remaining sections in the function group. The first section serves as a representation of its use, and a complete analysis would imply seeing the function animating different parts of the audio with the different colormaps it would use in each section, which would be time-consuming and tiring. Nevertheless, this visualization could be added in the future as an optional feature of the system.

Once the favorite animations are chosen, the final video is ready to be computed.

Chapter 7

Evaluation

We now detail the evaluation process deployed to analyze our approach, and the obtained results. We begin with the description of the produced artifacts that were used in its evaluation, then describe the online forms that served as the means to gather feedback regarding the system's creations, and finally analyze and discuss the obtained results.

7.1 Evaluation Method

To evaluate our system, we selected five songs, and each one was used to generate one video per animation analysis method: Mood Matching, Preference Model, and Manual Selection. The videos were generated using a frame rate of 12.5 frames per second, and a resolution of 720×720 pixels. An additional video was generated per song to serve as control in our study.

The songs were selected such that a variety of genres and moods were considered, to evaluate the system's versatility. The genres include electronic, jazz, classical, chant, and rock, while the moods vary both in arousal (level of energy) and in valence (how pleasant the mood is/how low is its level of stress).

Five forms were developed such that, for each form, only one song and the videos associated with it were shown. The song dataset is presented in Table 7.1, along with the indication of form-song correspondence. Table 7.2 provides links to the videos according to the associated audio and method. The videos can also be consulted in `https://vimeo.com/frommusic2animations`. The videos were presented accompanied by the audio, since we wished to evaluate our system's ability to create animations that were meant to be seen alongside the audio.

Research has shown that, in music videos and other multimedia, the perception of one of the mediums (such as audio) can bias the interpretation of the other (such as animation), so they are more likely to be perceived to match [2]. The Control method was developed so we could more accurately evaluate the relationships that study participants described. This method refers to the usage of a video that was created based on a different song, for which the audio was switched to that of the desired song, adjusting the length of the video accordingly.

The control videos were hand-picked by us from a collection of outputs from the other methods

Table 7.1: Songs used to evaluate the videos generated by the system using different methods, and their corresponding forms. Each song was used to create a unique form, for which the song was presented, along with the four associated videos, each corresponding to a different method of generation.

Form	Music
A	Autechre - “plyPhon”
B	Chet Baker - “It Could Happen To You”
C	Chopin - “Prelude In E Minor (opus 28, nº 4)”
D	Coro Madrigale Slovenico - “Sanctus”
E	The Beatles - “Help!”

Table 7.2: Links to the videos used in each form, with the indication of the method used to generate them. The form is indicated instead of the audio for simplicity. The corresponding songs can be consulted in Table 7.1.

Form	Control	Mood Matching	Preference Model	Manual Selection
A	https://vimeo.com/613975628	https://vimeo.com/606007552	https://vimeo.com/599897289	https://vimeo.com/606009454
B	https://vimeo.com/613971643	https://vimeo.com/606011721	https://vimeo.com/606025855	https://vimeo.com/606030509
C	https://vimeo.com/613970953	https://vimeo.com/606034680	https://vimeo.com/606036083	https://vimeo.com/606037203
D	https://vimeo.com/613971339	https://vimeo.com/606039308	https://vimeo.com/606041229	https://vimeo.com/606042817
E	https://vimeo.com/613975313	https://vimeo.com/606643520	https://vimeo.com/606642456	https://vimeo.com/606644254

generated on previous occasions. These were chosen such that the mood, style, and rhythm of the animations did not appear to match those of the audio. For instance, instead of “plyPhon” by Autechre, “Prelude In E Minor (opus 28, nº 4)” by Chopin was used, and instead of “It Could Happen To You” by Chet Baker, the shown video was created based on the song “Help!” by The Beatles. The other mappings are: “Prelude In E Minor (opus 28, nº 4)” by Chopin used “Muriel” by Bobby Richards (which is a dance/electronic track); “Sanctus” by Coro Madrigale Slovenico used “Sharp” by Jeremy Korpas (a song of the rock genre with an angry mood); and “Help!” by The Beatles used “The Awakening” by Patrick Patrikios (which has a cinematic and dark tone).

No information was provided to the participants regarding how the animations were created, and the videos were referred to using a version number. To reduce bias relating to the order of the videos shown to the participants, their order was shuffled in each form. Table 7.3 shows the mapping between the form/song, the version indicated in the form, and the method associated. The videos were shown to the participants according to the version order, i.e., from version 1 to version 4.

Table 7.3: Mapping between the versions indicated in the forms, the songs, and the methods used to produce the videos.

Form	Music	Version 1	Version 2	Version 3	Version 4
A	Autechre - “plyPhon”	Preference	Manual	Control	Mood
B	Chet Baker - “It Could Happen To You”	Manual	Preference	Mood	Control
C	Chopin - “Prelude In E Minor (opus 28, nº 4)”	Mood	Control	Preference	Manual
D	Coro Madrigale Slovenico - “Sanctus”	Control	Mood	Manual	Preference
E	The Beatles - “Help!”	Manual	Control	Preference	Mood

7.1.1 Form Description

The exemplary interface for Form A, corresponding to the song “plyPhon” by Autechre, can be consulted in Appendix B. All the forms had a similar structure, varying only in terms of audio and videos presented. The forms started by asking for consent for the gathering of demographic information, followed by a question to know it was the first form that the participant was answering out of the five. After the demographic information was gathered, the artifact characterization took place. First, the audio was shown to the participant, which was asked to provide from one to three adjectives/expressions that described it. Then, each video version was displayed in turn, and only after the participant had characterized it (using the same method as before) was the next video shown. This was identical from version 1 to version 4.

We made sure to first show every audio or video in turn, asking the participant to describe them freely, and only then asking for opinions regarding particular aspects and displaying lists of adjectives. If the participant had already answered a form previously, then they might feel inclined to provide descriptions using the terms already seen in the questions.

This could only have been avoided if a person could only answer one form in total, which would reduce the number of gathered answers. Alternatively, we could have a unique form instead of five, for which the participant would first describe freely all the songs/videos, and only then answer questions. Since these options were not optimal, we opted for this approach instead, yet future work should take these limitations into account when designing the evaluation forms. Although the percentage of answers for each form from first submitting participants was considerable (Table 7.4), this is an aspect to keep in mind.

After the participant had provided all the unrestrained descriptions, another round of questions about the audio and video versions were presented. As before, all the audio questions were asked first, then all the version 1 questions, then version 2, and so on.

We collected some opinions resorting to the Likert scale method [52]. In our evaluation process, this method was deployed by presenting a statement to the participant, and then asking them to indicate how much they agree or disagree using the following scale: 1 - Strongly Disagree, 2 - Disagree, 3 - Undecided, 4 - Agree, 5 - Strongly Agree. This method allows different degrees of opinion, including a neutral one, while still keeping the options relatively constrained.

The Likert scale questions were different for audios and videos. In the case of the audio, the statements were “I like it a lot” and “I consider it very creative”. Then, for each video, we asked for feedback regarding the statements “I consider it a music video”, “I like it a lot”, “I consider it very creative”, “I consider it abstract”, and “I think the visuals go well with the audio”.

For each audio and video, after the Likert scale questions were asked, the participant was asked to choose all the adjectives that apply out of a list. These adjectives were: exciting, calm, happy, enjoyable, boring, surprising, sad, aggressive, funny, interesting, disgusting, fearful, confusing, numb, and tender.

Table 7.4: Submissions

Form	Music	Submissions	First Submitting
A	Autechre - “plyPhon”	34	33
B	Chet Baker - “It Could Happen To You”	35	13
C	Chopin - “Prelude In E Minor (opus 28, nº 4)”	32	9
D	Coro Madrigale Slovenico - “Sanctus”	32	14
E	The Beatles - “Help!”	34	15
Total		167	84

We looked for mood, emotional, and feeling adjectives across the spectrum, that would also make sense to use when describing a song or video, while trying to keep the list short.

The last question asked the participant to order the video versions from most preferred to least preferred.

7.2 Results and Discussion

The number of submitted answers for each form can be consulted in Table 7.4, showing the total submission count for each one, and the number of answers that correspond to first-time participants, that is, people who did not answer any form before that one. In total, there were 167 form submissions, obtained from 84 different people. Each form had a similar answer count, between 32 and 35, while the answers for form A were mostly from first-time participants, with a proportion of 97% of all answers, and for the rest of the forms this percentage was considerably lower, ranging from 28% to 44%.

Table 7.5 presents the distributions of several demographic variables: age, gender, level of expertise in Audiovisual Analysis or related fields, and how often the participant watches music videos. Specific distributions for each form can be consulted in Appendix C (Figure C.1 and Figure C.2). We can see that most participants belong to the 18-29 age group, and there is a slight imbalance in terms of gender, with more male participants than other groups. Therefore, our results are more representative of these demographics than the others. The reported frequency of watching music videos appears to be somewhat distributed across “at least once a month” and “everyday”, with the options “at least once a month/week” being predominantly chosen. Most participants indicated no expertise in the field, with some participants indicating almost no experience or self-learning.

7.2.1 Video Descriptions

Table 7.6 shows the most frequent descriptors provided for the different audios and videos, discarding the control video. Some word processing was performed to translate Portuguese terms into English. Some videos elicit descriptions related to feelings and states, such as calm, relaxing, energetic, and sad.

(a) Age Group

Age	
< 18	0
18 - 29	58
30 - 39	5
40 - 49	6
50 - 60	12
> 60	3
No Answer	0

(b) Gender

Gender	
Male	44
Female	37
Other	1
No Answer	2

(c) Frequency of watching music videos

Music Videos Freq.	
Never	3
Less than once a month	19
At least once a month	23
At least once a week	22
Everyday	17
No Answer	0

(d) Expertise Level

Expertise Level	
No expertise	44
Almost no experience	21
Self-learning	15
Pursuing a degree	1
Has a degree	3
No Answer	0

Table 7.5: Distribution of participants according to several demographics: age group, gender, frequency of watching music videos, and expertise level.

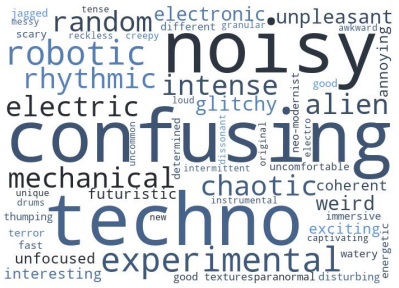
These correspond to the videos that have descriptions more evidently related to their audio counterparts, such as “Prelude In E Minor (opus 28, nº 4)” and “Help!”. Descriptions more explicitly relating to the visual textures are also provided, and include colorful, psychedelic, trippy, and hypnotizing.

Table 7.6: Top descriptions provided by users for the audios and the videos in general discarding the control (Videos w/o Control), when asked to ”Describe the previous audio/video with at least one adjective/expression”.

Source	Pos	Form				
		A	B	C	D	E
Audio	1º	robotic	relaxing	sad	religious	happy
	2º	confusing	calm	calm	calm	cheerful
	3º	noisy	smooth	melancholic	relaxing	uplifting
Videos w/o Control	1º	psychedelic	trippy	sad	confusing	colorful
	2º	colorful	colorful	relaxing	interesting	interesting
	3º	hypnotizing	psychedelic	calm	weird	energetic

Since the answers from Form A were almost entirely from participants who were answering one of these forms for the first time, we will take their descriptions as representative of the video descriptors. Figure 7.1 represents the most common adjectives and expressions used to describe the audio for “plyPhon” by Autechre (Audio), and to describe the corresponding videos, with the exception of the control one (Videos w/o Control). The most common terms for each audio or video are also indicated.

From analyzing the expressions, it appears that our approach creates visuals that create an impression similar to that caused by this audio, with common words being: techno, robotic, electronic, and alien. We also notice that words used to describe the textures of the videos are: psychedelic, colorful as well as dark, hypnotizing, trippy, and strange. These appear to capture the words used to describe the animations generated by our approach, as they are also used for videos from the other audios. Word clouds based on the descriptions provided for the remaining forms are available in Appendix C (Figures C.3-C.6).



(a) Audio

(b) Videos w/o Control

Source	Top 3 Descriptors
Audio	robotic, confusing, noisy
Videos w/o Control	psychedelic, colorful, hypnotizing
Manual	dark, hypnotizing, colorful
Preference	colorful, psychedelic, trippy
Mood	psychedelic, trippy, weird
Control	colorful, boring, abstract

(c) Top 3 Descriptors

Figure 7.1: Word Clouds of the descriptions provided for Form A: Autechre - “plyPhon”

7.2.2 Adjective Selection

To analyze the adjectives that the participants selected from the lists, we used the Bhattacharyya distance, D_{BC}, a measure of similarity between two probability distributions (Equation (A.8)). Concerning the adjectives that participants selected out of a list for the audio and each video, Figure 7.2 shows the Bhattacharyya distance for the adjective distribution relating to the audio, compared with the distribution considering all the videos except the control, and also compared with the distributions of each video individually.

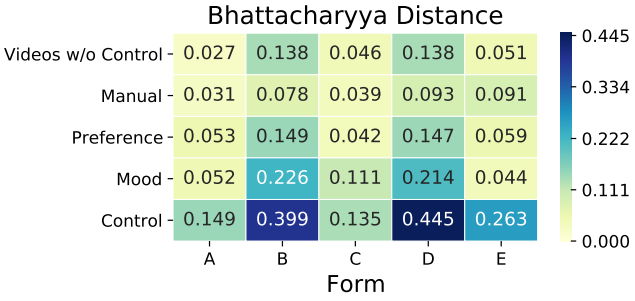


Figure 7.2: D_{BC} between the adjective selection distributions of audio and the animation methods for each form.

The values for the control method can be regarded as the baseline performances of each form. A few patterns emerge from this analysis. With a few exceptions, the manual method distributions obtained better matches with the audio, followed by the preference method, the mood method, and finally the control method. In addition, some songs generated smaller distance metrics than others. In particular, the songs “It Could Happen To You” and “Sanctus” had videos for which the selected adjectives were less related to the audio when compared to the other songs.

Overall, the Bhattacharyya distances for the videos without control analysis were always less than 35% of their control counterparts, which can be interpreted as a correlation between audio perception and the visual perception that the animation methods produce. The adjective distributions for all the

songs and videos can be found in Appendix C in Figure C.7, along with the top adjective list for each one in Table C.1.

7.2.3 Analysis of Opinion Statements

We now move on to the analysis of the answers provided for the Likert scale questions. To refer to the statements used in these questions, we will use the following short-hands: Audio Creative - "I consider [the audio] very creative", Audio Liked - "I like [the audio] a lot." Video Abstract - "I consider [the video] abstract", Video Fits Audio - "I think the visuals go well with the audio", Video Creative - "I consider [the video] very creative", Video Liked - "I like [the video] a lot", "Is Music Video?" - "I consider [the video] a music video". We recall that the meaning of the answers provided by the participants goes from 1 - Strongly Disagree, to 5 - Strongly Agree.

The following metrics were selected to capture the overall responses for each of the video questions: mode (most frequent value), median (the value for which the count of samples that are equal to it or lower is equal to the amount of samples that are equal to it or higher), average (sum of all samples divided by their total count) and standard deviation (square root of the mean squared differences between the samples and the average, a metric related to the dispersion of the samples).

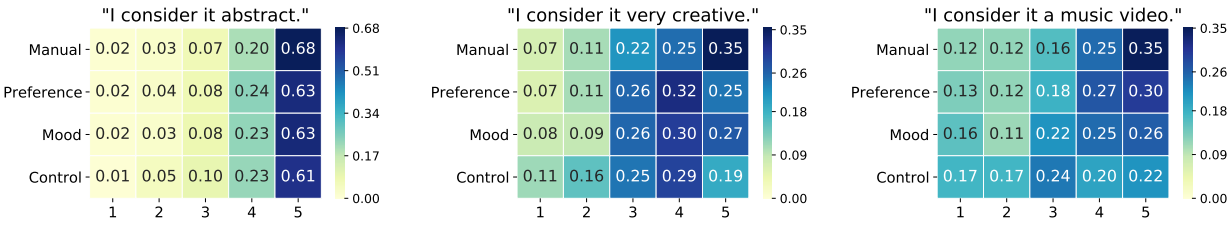
The average and standard deviation metrics should be analyzed with care, as the intervals between the answer values should not be assumed equal, since they correspond to ordinal data [53], e.g., the distance between the interpretation of Strongly Disagree (1) and Disagree (2) may be different from Undecided (3) and Agree (4). Although these metrics still provide some information regarding the distributions of the answers, they should be analyzed with this in mind.

Table 7.7 shows representative metrics regarding the video questions, with respect to the distributions of each animation method. These metrics complement the explicit answer distributions that will be presented regarding each question, in Figures 7.3-7.5.

Table 7.7: Metrics for each of the video questions, for each method. The metrics are: mode (Mod), median (Med), average (Avg), and standard deviation (Std).

	Video Abstract				Video Creative				Is Music Video?				Video Fits Audio				Video Liked			
	Mod	Med	Avg	Std	Mod	Med	Avg	Std	Mod	Med	Avg	Std	Mod	Med	Avg	Std	Mod	Med	Avg	Std
Manual	5	5	4.49	0.89	5	4	3.71	1.25	5	4	3.59	1.38	5	4	3.73	1.31	5	4	3.38	1.36
Preference	5	5	4.43	0.91	4	4	3.57	1.16	5	4	3.49	1.37	4	4	3.47	1.22	4	3	3.17	1.28
Mood	5	5	4.44	0.90	4	4	3.58	1.21	5	4	3.35	1.39	4	3	3.20	1.31	4	3	3.07	1.33
Control	5	5	4.37	0.95	4	3	3.29	1.26	3	3	3.11	1.39	1	2	2.54	1.33	3	3	2.65	1.26

Figure 7.3 presents several metrics, averaged over all the forms, for the answer distributions relating to questions on whether the videos were abstract, creative, and music videos.



(a) Video Abstract **(b)** Video Creative **(c)** Is Music Video?

Figure 7.3: Answer distributions regarding the Likert scale questions: (a) "I consider it abstract"; (b) "I consider it very creative"; (c) "I consider it a music video". The answers go from 1 - Strongly Disagree, to 5 - Strongly Agree.

Video Abstract

Information regarding whether the participants considered the visuals abstract is presented in column Video Abstract of Table 7.7, and in Figure 7.3(a). We can see that, overall, the videos from all the methods were considered abstract, with most participants classifying all methods as 5, and a few using 4, with rare occurrences of the values 3 and below. The varied animation methods don't lead to significantly different answers to this question. Analyzing the distributions song by song, we have noted that the distributions are very similar as well (Figure C.8 in Appendix C presents the answer distributions per song and per method).

Video Creative

The answer metrics for the question "I consider it very creative", referring to each video, can be consulted in the column Video Creative of Table 7.7, and in Figure 7.3(b) (results for each song can be consulted in Figure C.9 in Appendix C).

The fact that the control video had a considerable performance in this regard, although opinions diverged, is important to note. Its distribution, even considering its overall lower values, is comparable to the distributions of the automatic animation methods. This means that, despite the disagreement between audio and animations, a considerable percentage of participants still classified the videos as creative to some extent. We believe this adds to the legitimacy of our approach in terms of the ability to generate creative animations, since the control animations also correspond to animations produced using our system.

All the other methods present classifications mostly ranging from 3 to 5, with 4 being the median and mode in all of them, except for the manual method, which is significantly skewed towards 5, for which the mode is 5. The preference and mood methods don't present significant differences, while the manual method seems to be the most effective at producing videos deemed creative. Since the participants had

no information regarding the methods used to generate each method, we can speculate on what led to this slight inclination of results. The correlation analysis that we present later provides some relevant insights.

Is Music Video

The results for the agreement distributions regarding the statement “I consider it a music video” are in column “Is Music Video” from Table 7.7, and in Figure 7.3(c).

We can see that the mode was 5 and the median was 4 for all the versions except the control one, although the distributions are considerably dispersed across the options. The control video produced opinions almost evenly distributed, then the distributions get more skewed to agreeing with the statement as the methods go from mood, to preference, to manual. This is consistent with the expected ability for selecting the best animations to go with an audio: a person has more expertise than the program that simulates the person’s preference, and the preference model performs better than the mood model, since it was specifically trained on audio-animation pairs.

The distributions for each song showed that some songs lead to distributions more skewed towards strongly agreeing, such as “plyPhon”, “Prelude In E Minor (opus 28, nº 4)”, and “Help!”, while others lead to distributions that are more spread out across the options, such as “It Could Happen To You” and “Sanctus”. These graphs can be consulted in Appendix C, in Figure C.10. This is evidence that our approach is more appropriate for some music genres than others.

Video Fits Audio

The participants’ answers regarding the question “I think the visuals go well with the audio” can be analyzed through column Video Fits Audio from Table 7.7, and in Figure 7.4. The overall analysis of the methods (bottom right corner of Figure 7.4) shows that, while most answers regarding the control method were 3 or bellow, with a mode of 1 and a median of 2, most answers for the other methods were 3 or higher.

In particular, the manual videos tend to align better with the audio, which is no surprise, since there was a human user deciding what animations to use. The preference method is also strongly skewed towards an agreement with the statement, which is a sign that this method is capable of selecting adequate animations, even if its performance is worse than for manual selection. Although the mood method presents a distribution more spread out across all the options, its more frequent answer is 4, with 3 being a close second. We can then conclude that the preference method is the automatic method that can produce more reliable results in terms of selecting visuals that go well with the audio.

Analyzing the distributions per song, the songs “plyPhon”, “Prelude In E Minor (opus 28, nº 4)”, and “Help!” are more skewed to an agreement with the statement, while “It Could Happen To You” and

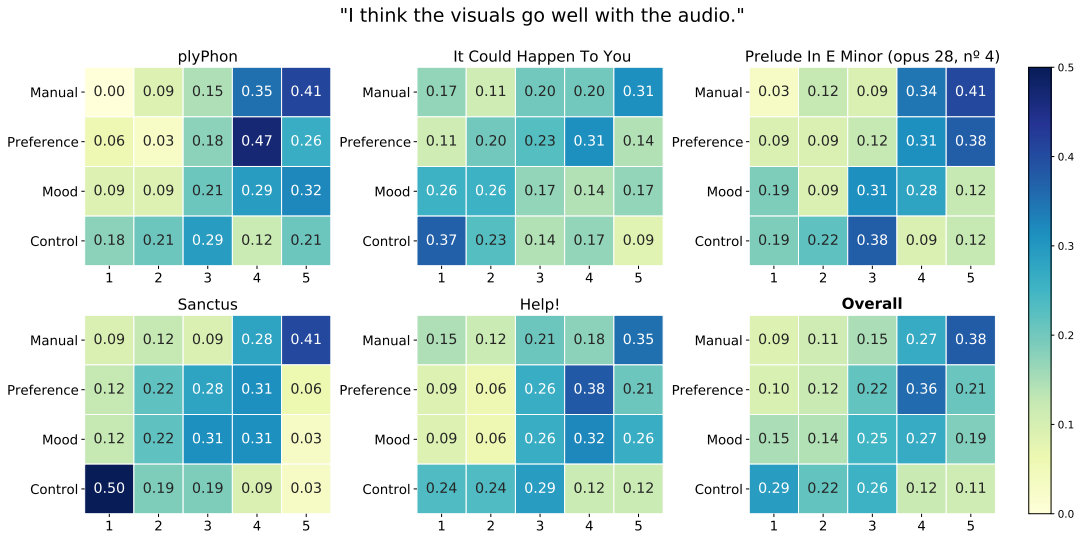


Figure 7.4: Answer distributions regarding the Likert scale question "I think the visuals go well with the audio". The answers go from 1 - Strongly Disagree, to 5 - Strongly Agree.

"Sanctus" are more spread out (in the case of "It Could Happen To You") or centered (in the case of "Sanctus"). This coincides with the comparison between the most frequent descriptors matching the audio and the videos, as discussed before.

This contributes to the assessment that the type of animation developed in this work fits some types of music better than others. On the other hand, the manual versions exhibit positive metrics in this regard, which seems to imply that the fitness of the generated animations can be improved by the manual selection of the samples.

Video Liked

The values obtained from the answers to the level of agreement with the statement "I like it a lot", referring to the videos, can be consulted in column Video Liked of Table 7.7, and in Figure 7.5. When analyzing song by song, it appears that the opinions are specific to each song-method pair, and an overall pattern between graphs is not obvious. Another remarkable feature is that opinions greatly diverged in this matter for some videos, which is coherent with the notion that aesthetic tastes are based on subjective personal criteria by each individual, at least to some extent. Nevertheless, through the analysis of the graph referring to the overall performance of the metrics (bottom right corner of Figure 7.5) and looking at the mode, median and average values of answers, we can see a tendency. In general, the control videos were not strongly liked, and the mood method videos originated diverging opinions with a mode of 4 and a median of 3. Although presenting the same mode and median, the preference videos were considered slightly more likable. The manual videos, on the other hand, resulted in a mode of 5 and a median of 4, being the most likable of all. Since we selected each animation to be shown in the manual

videos, this confirms that a person is more reliable to select animations that are more probable of being liked, compared with the automatic variations of our system.

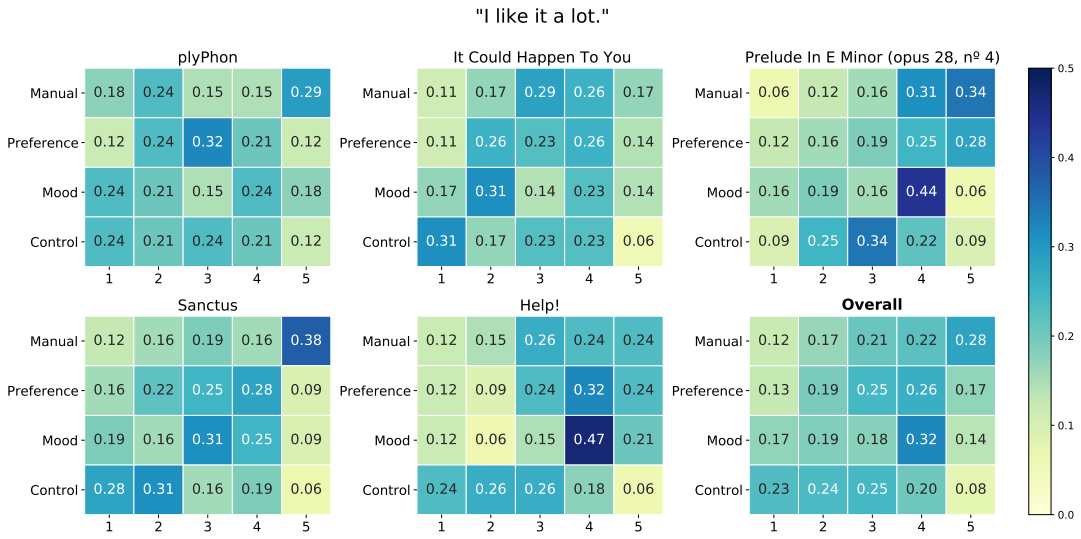


Figure 7.5: Answer distributions regarding the Likert scale question "I like it a lot", regarding each video. The answers go from 1 - Strongly Disagree, to 5 - Strongly Agree.

7.2.4 Version Preference

Through Figure 7.6, the overall preference order for each video and song can be analyzed. Comparing the song graphs between each other, we can see that the order of preference often depends on the particular videos in question, and it is not always the same. Even so, an overall tendency is noticeable by consulting the bottom right corner of Figure 7.6, presenting the overall method preference order.

The manual videos are often the top-ranked ones, followed by the preference videos, with the mood videos being placed in third place (although with a distribution similar to the preference videos), and the control videos are frequently placed last. This is consistent with the analysis regarding the themes of whether the videos are considered music videos, whether the animations go well with the audio, and if the videos are liked.

7.2.5 Correlation Analysis

As a means to get more insight into the different aspects that may have influenced the participants' answers, Figure 7.7 presents the correlation analysis between different concepts considered in the preceding analysis. To compute such metrics, Spearman's correlation coefficient was deployed (Equation (A.7)). We shall now look into notable relations between these concepts.

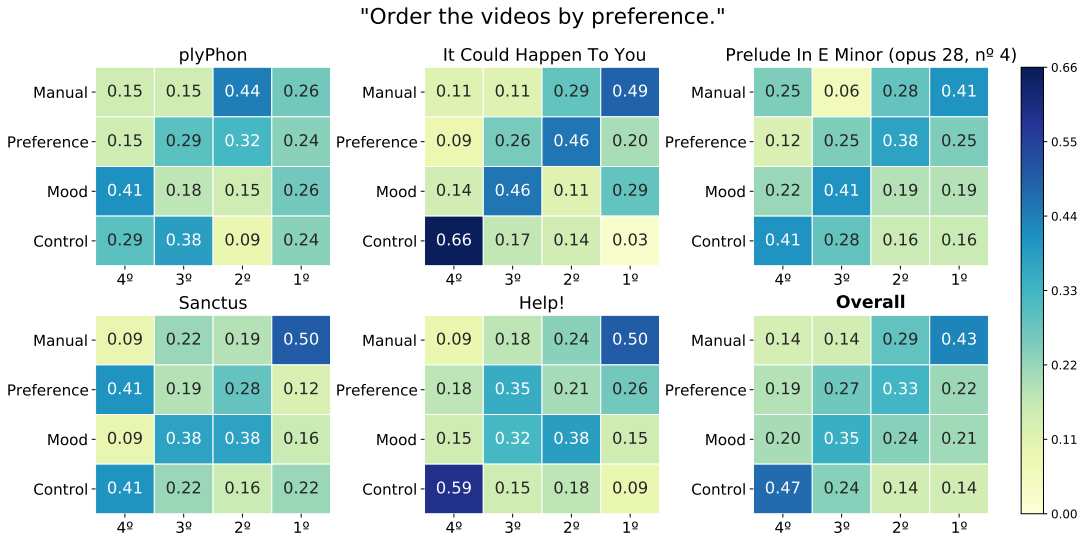


Figure 7.6: Answer distributions regarding the average preference order of each video, from least preferred by the participant (4º) to most preferred (1º).

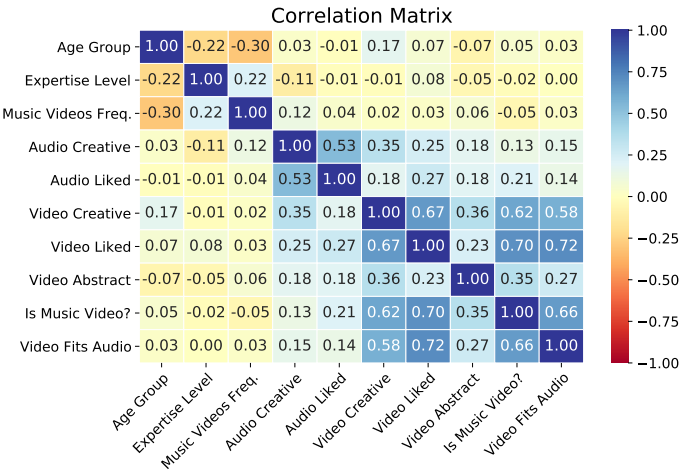


Figure 7.7: Spearman's correlation coefficient between different concepts analyzed in this work.

In terms of demographic groups, we can see that age is somewhat negatively correlated with the expertise level and the frequency of watching music videos. The relations between the demographic groups and the way the participants classified the content do not appear to be critical, with perhaps the exception that older participants considered the videos creative slightly more often than younger participants.

On the other hand, several noticeable relations between audio and video classifications emerged from this analysis. The classifications provided regarding whether the participant liked the audio or considered it creative appear to be positively correlated with whether they liked the videos and considered them creative, abstract, a music video, and a good fit for the audio, to some extent.

Since this analysis considers all submissions, it could be the case that some participants tend to answer more positively overall, and others more negatively, which may result in these small correlations. Out of these values, it seems that how much an audio is considered creative is positively associated with whether the videos are considered creative as well. Similarly, how much a participant likes a song appears to be slightly correlated with whether they like the videos.

Other correlations appear to be more prominent. For instance, while the answers regarding the statement that the video is abstract are somewhat related to the other answers, the other video classifications appear to be strongly correlated. In other words, the answers for the video statements regarding if the video is creative, liked, a music video, and a good fit, all present Spearman's correlations between 0.58 and 0.72 among each other.

Although causality relations between these concepts are not deducible from this information, it is apparent that they are linked, and more so than all the other analyzed concepts. This reflects the link between the notion of creativity and what is considered valuable, as we described in Section 2.1.

7.2.6 Overview

We now provide a brief overview of the results obtained from the evaluation phase. Common words used to describe our system's outputs were colorful, psychedelic, trippy, and hypnotizing.

Participants considered the videos abstract, and tended to agree with the statement that the videos were creative, even the control versions. Opinions concerning the creativity of the videos, if they were music videos, if the visuals suited the song, and if the participants liked the videos, all appear to be correlated to a substantial degree. Regarding whether the videos were music videos, the answers were less unanimous, but more predominantly skewed towards an agreement with the statement as the methods went from control, to mood, to preference, to manual. Opinions were more consistent for whether the visuals went well with the audio, and followed the same tendency. Regarding the likeability of the videos produced by the methods, the opinions obtained were dispersed among the options, appearing to be highly particular to each video in question, and the taste of each person.

By comparing the rankings provided by the participants on which videos they preferred, the control version was mostly the least favorite, followed by the mood version, then the preference method version, and the manual version was the preferred one in general. This tendency was apparent across metrics.

Another common pattern throughout the analysis was that some songs lead to better metrics for the automatic methods than others. From this, we extrapolate that our approach fits better with certain genres, particularly electronic, dance, classical, and psychedelic rock, while not adapting as well to others, like jazz and chant.

Nevertheless, the manual method exhibited potential in terms of creating visuals that go well with the audio, due to its flexibility as a co-creation tool.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this work, we have presented a system that creates original abstract videos with animations that interpret a song. The driving idea behind our approach was the generation of animations by drawing time-dependent heatmaps frame by frame, which should move in accordance with the audio. The videos that our system generated can be consulted in https://vimeo.com/frommusic2animations.

It should be noted that the visualizations created are not meant to be interpreted as a direct mapping of audio data. Because we aimed to present surprising animations, we created visualizations in the sense that the visuals were influenced by the audio, but the element of chance was a key component of our work.

To better represent the structure of the song, it is segmented according to audio novelty and the average tempo, such that each audio section generates its own animation. The sections are analyzed by mood - arousal and valence, predicted using the Mood Model, an SVR model trained on the DEAM dataset. Audio sections are clustered by mood to create colormap groups, which indicate the sections that will use the same colors. The audio sections are also independently grouped according to their audio frequency distributions to create function groups, which dictate the sections that use the same animation function.

New colormaps are generated by randomly assembling sequences of RGB values that are then linearly interpolated to create a function that maps a value in the range [0,1] to an RGB value. The animation functions are obtained by randomly arranging mathematical operations, constants, pixel location variables, and time-dependent variables. To create movement that can be associated with the audio, we use the frequency intensities extracted from the audio as the temporal variables, along with sinusoidal functions that align with the average tempo of the audio.

To select the colormaps and functions to be used in each group, first a collection of samples is generated, then the final ones are selected using one of the animation analysis methods.

The first developed method, Mood Matching, is focused on the estimated audio mood for each section. It computes the fitness of audio-colormap pairs by evaluating color Lightness and Saturation, and

estimates the arousal of animations as being proportional to the average amount of detail of the frames, as well as the average amount of color change between frames.

To create the Preference Model, a dataset was generated, composed of audio-animation pairs, and an interface was developed to collect user classifications regarding their fitness. The features used result from joining the features of each medium into a unique feature vector. The audio features were mostly features used to estimate the audio mood, and the animation features were ones considered relevant to predict the user's aesthetic taste. A variety of models and parameters were tested (KNN, DT, and MLP), and the combination with the highest average precision on the validation set was an SVR model, obtaining a final validation set precision of 84.6%. Different user classifications can be provided to create a new model of preference. This study revealed the ability of the considered features and models to reasonably simulate human preference modeling.

A final animation analysis method relies on the user to select the best colormaps out of a collection previously filtered according to Mood Matching. The top fitness animation functions, which were filtered using the Preference Model, are shown to the user through an interface that displays the animations along with the audio, which then selects their favorite for each section.

Five songs from varied genres were considered for evaluating our system, and online forms were deployed to measure different metrics, producing positive results. Specifically, the videos were considered creative and, in terms of likeability and audio-visual alignment, the results indicated that the Preference Model performed better than Mood Matching, and the Manual Selection method showed promising results for its use as a co-creation tool to generate music visualizations that fit the audio.

We hope our work inspires further discussion regarding creative artifacts, the creative process, and the possibility of creative computational systems. We believe the results obtained in this work further motivate the study of human preference modeling, with applications not only in the evaluation of aesthetic measures, but also regarding the development and improvement of co-creation tools.

8.2 Future Work

Within our approach, several improvements could be developed. The analysis of colormap groups and function groups should ideally consider all the sections that belong to it. In terms of colormap selection, color theory could be used to evaluate the colormaps, and the co-creation tool could include the option of manually creating colormaps through an interface. In terms of function analysis, the function filter should rule out noisy or visually overwhelming animations, and different features, models, parameters, and training datasets could be explored regarding animation analysis. The presented system could serve as the basis for further study of human preference modeling in the field of music visualization.

Another interesting approach to music visualization could use a common theme throughout the video,

connecting consecutive animations as consecutive animation mutations, for instance.

A considerable limitation of our system was its processing time, which could be improved by time optimization of our algorithm, and by taking advantage of parallel processing techniques enabled by GPUs. Significantly reducing the processing time would make viable the deployment of a Genetic Algorithm to iteratively increase the fitness of generated animations before selecting them.

An app for video creation based on this work's approach could be developed, as an automatic music visualization tool, with the possibility of co-creation. It would be desirable to measure the performance of the Preference Model and the co-creation method, which would require the deployment of the app and its subjective evaluation by a considerable number of users.

It would be valuable to explore other ways of animating to music, as we have merely explored one possible approach of many, such as fractals, reaction-diffusion systems, CPPNs, 3D animation approaches, among others.

Bibliography

[1] T. Z. Tardif and R. J. Sternberg, "What do we know about creativity?" *The nature of creativity: Contemporary psychological perspectives*, pp. 429–440, 1988.

[2] M. Boltz, "Music videos and visual influences on music perception and appreciation: Should you want your mtv?" *The psychology of music in multimedia*, pp. 217–234, 2013.

[3] J. C. Yoon, I. K. Lee, and S. Byun, "Automated music video generation using multi-level feature-based segmentation," *Multimedia Tools and Applications*, vol. 41, no. 2, pp. 197–214, jan 2009.

[4] M. A. Boden, *The Creative Mind: Myths and mechanisms*. Routledge, 2004.

[5] Merriam-Webster, "Valuable," in *Merriam-Webster.com dictionary*, accessed: October 14, 2021. [Online]. Available: https://www.merriam-webster.com/dictionary/valuable

[6] A. Koestler, *The Act of Creation*. Penguin Books, 1964.

[7] R. K. Sawyer, *Explaining Creativity: The Science of Human Innovation*. Oxford university press, 2011.

[8] Acoustical Society of America, "Acoustical Terminology," *ANSI/ASA S1.1-2013*, 2013.

[9] X.-S. Hua, L. Lu, and H.-J. Zhang, "Automatic music video generation based on temporal pattern analysis," in *Proc. of the 12th ACM international conference on Multimedia*, 2004, pp. 472–475.

[10] T. Nakano, S. Murofushi, M. Goto, and S. Morishima, "Dancereproducer: An automatic mashup music video generation system by reusing dance video clips on the web," in *Proc. of the 8th Sound and Music Computing Conference*, 2011, pp. 183–189.

[11] J. C. Wang, Y. H. Yang, I. H. Jhuo, Y. Y. Lin, and H. M. Wang, "The acousticvisual emotion guassians model for automatic generation of music video," in *Proc. of the 20th ACM International Conference on Multimedia*, 2012, pp. 1379–1380.

[12] R. Cai, L. Zhang, F. Jing, W. Lai, and W. Y. Ma, “Automated music video generation using web image resource,” in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, 2007, pp. 737–740.

[13] X. Wu, Y. Qiao, X. Wang, and X. Tang, “Cross matching of music and image,” in *Proc. of the 20th ACM International Conference on Multimedia*, 2012, pp. 837–840.

[14] J. Foote, M. Cooper, and A. Girgensohn, “Creating music videos using automatic media analysis,” in *Proc. of the ACM International Multimedia Conference and Exhibition*, 2002, pp. 553–560.

[15] L. Lu, D. Liu, and H. J. Zhang, “Automatic mood detection and tracking of music audio signals,” in *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 1, jan 2006, pp. 5–18.

[16] R. E. Thayer, *The biopsychology of mood and arousal*. Oxford University Press, 1990.

[17] H. Cohen, “Aaron, colorist: from expert system to expert,” 2006, accessed: October 18, 2021. [Online]. Available: http://www.aaronshome.com/aaron/publications/urbana-final.doc

[18] S. Colton, “The painting fool: Stories from building an automated painter,” in *Computers and Creativity*, J. McCormack and M. D’Inverno, Eds. Springer, 2012, pp. 3–38.

[19] R. Dawkins, *The Blind Watchmaker*. Norton & Company, Inc, 1986.

[20] S. Proud, “William Latham,” 2013, accessed: October 18, 2021. [Online]. Available: http://latham-mutator.com/

[21] K. Sims, “Artificial evolution for computer graphics,” in *Proc. of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, jul 1991, pp. 319–328.

[22] P. Machado and A. Cardoso, “All the truth about NEvAr,” *Applied Intelligence*, vol. 16, no. 2, pp. 101–118, feb 2002.

[23] D. A. Hart, “Toward greater artistic control for interactive evolution of images and animation,” in *Workshops on Applications of Evolutionary Computation*, 2007, pp. 527–536.

[24] T. Unemi, “Automated daily production of evolutionary audio visual art - an experimental practice,” in *Proc. of the 5th International Conference on Computational Creativity*, 2014.

[25] K. Sims, “Evolving virtual creatures,” in *Proc. of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. Association for Computing Machinery, Inc, jul 1994, pp. 15–22.

[26] Sims, Karl, “Seven experiments in procedural animation,” 2018, accessed: October 21, 2021. [Online]. Available: https://karlsims.com/seven.html

[27] S. Draves, “The electric sheep screen-saver: A case study in aesthetic evolution,” in *Proc. of the 3rd European conference on Applications of Evolutionary Computing*, 2005, pp. 458–467.

[28] L. Berov and K. U. Kühnberger, “Visual hallucination for computational creation,” in *Proc. of the 7th International Conference on Computational Creativity*, 2016, pp. 107–114.

[29] A. Mordvintsev, C. Olah, and M. Tyka, “Inceptionism: Going deeper into neural networks,” 2015, accessed: October 21, 2021. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[30] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” *arXiv preprint arXiv:1508.06576*, 2015.

[31] A. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone, “Can: Creative adversarial networks generating “art” by learning about styles and deviating from style norms,” *arXiv preprint arXiv:1706.07068*, 2017.

[32] V. Zorić and B. Gambäck, “The image artist: computer generated art based on musical input,” *Proc. of the 9th International Conference on Computational Creativity*, pp. 296–303, 2018.

[33] L. Aleixo, H. S. Pinto, and N. Correia, “From music to image: a computational creativity approach,” in *Artificial Intelligence in Music, Sound, Art and Design*, J. Romero, T. Martins, and N. Rodríguez-Fernández, Eds. Springer International Publishing, April 2021, pp. 379–395.

[34] N. Markuš, “Using cppns to generate abstract visualizations from audio data,” accessed: October 10, 2021. [Online]. Available: https://nenadmarkus.com/p/visualizing-audio-with-cppns/

[35] K. O. Stanley, “Compositional pattern producing networks: A novel abstraction of development,” *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[36] P. Galanter, “Computational aesthetic evaluation: Past and future,” in *Computers and Creativity*, J. McCormack and M. D’Inverno, Eds. Springer, 2012, pp. 255–293.

[37] E. den Heijer and A. Eiben, “Investigating aesthetic measures for unsupervised evolutionary art,” *Swarm and Evolutionary Computation*, vol. 16, pp. 52–68, 2014.

[38] B. J. Ross, W. Ralph, and H. Zong, “Evolutionary image synthesis using a model of aesthetics,” in *Proc. of the 2006 IEEE International Conference on Evolutionary Computation*, 2006, pp. 1087–1094.

[39] D. Norton, D. Heath, and D. Ventura, “Autonomously creating quality images,” *Proc. of the 2nd International Conference on Computational Creativity*, pp. 10–15, 2011.

[40] J. Correia, P. Machado, J. Romero, and A. Carballal, “Evolving figurative images using expression-based evolutionary art,” in *Proc. of the 4th International Conference on Computational Creativity*, 2013, pp. 24–31.

[41] Y. Li, C. Hu, L. L. Minku, and H. Zuo, “Learning aesthetic judgements in evolutionary art systems,” *Genetic Programming and Evolvable Machines*, vol. 14, no. 3, pp. 315–337, 2013.

[42] A. Aljanaki, Y.-H. Yang, and M. Soleymani, “Developing a benchmark for emotional analysis of music,” *PLoS ONE*, vol. 12, no. 3, 2017.

[43] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[44] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[45] D. A. Reynolds, “Gaussian mixture models,” *Encyclopedia of biometrics*, vol. 741, pp. 659–663, 2009.

[46] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “Optics: Ordering points to identify the clustering structure,” *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[47] K. Perlin, “Improving Noise,” *ACM Transactions on Graphics*, pp. 681–682, 2002.

[48] J. M. Prewitt, “Object enhancement and extraction,” *Picture processing and Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.

[49] J. E. Pearson, “Complex patterns in a simple system,” *Science*, vol. 261, no. 5118, pp. 189–192, 1993.

[50] R. Bresin, “What is the color of that music performance?” in *Proc. of the 2005 International Computer Music Conference*, 2005, pp. 367–370.

[51] X. P. Gao, J. H. Xin, T. Sato, A. Hansuebsai, M. Scalzo, K. Kajiwara, S. S. Guan, J. Valldeperas, M. J. Lis, and M. Billger, “Analysis of cross-cultural color emotion,” *Color Research and Application*, vol. 32, no. 3, pp. 223–229, 2007.

[52] R. Likert, “A technique for the measurement of attitudes,” *Archives of Psychology*, 1932.

[53] S. Jamieson, “Likert scales: How to (ab)use them?” *Medical education*, vol. 38, no. 12, pp. 1217–1218, 2004.

[54] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson Education, 2016.

[55] J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson correlation coefficient,” in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[56] C. Spearman, “The proof and measurement of association between two things,” *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[57] A. Bhattacharyya, “On a measure of divergence between two multinomial populations,” *Sankhyā: the indian journal of statistics*, pp. 401–406, 1946.

[58] H. H. Aghdam and E. J. Heravi, “Convolutional neural networks,” in *Guide to convolutional neural networks*. Springer, 2017, pp. 85–130.

Appendix A

Definitions and Techniques

Here we provide some short descriptions of several concepts, metrics, and models mentioned throughout this work, with the intent of clarifying their usage and simplifying the main text in which they are inserted. Further details can be consulted in the indicated sources.

A.1 Audio Analysis

A.1.1 Fourier Analysis

Fourier analysis is the general term used to describe the process by which a function is decomposed into simpler oscillatory terms. This analysis is extremely useful in the study of sound, as we can approximate it by determining the main frequencies that make up the signal, and their intensities. The process that maps the original function into its representation in terms of frequencies is called a Fourier transform.

Since digital audio corresponds to a sequence of samples of a continuous function, the Discrete-time Fourier Transform (DTFT) is a useful method for analyzing its frequencies. The transform takes uniformly spaced samples, and computes a continuous function of frequency, which represents a decomposition of the original continuous function into oscillatory components. The discrete representation of the frequency distribution over time is obtained using the Discrete Fourier Transform (DFT), which samples from the DTFT at the same sample rate of the original signal. An algorithm that is frequently used to compute the DFT is the FFT.

An STFT, in the discrete-time case, applies a Fourier transform separately to equal length frames of the original signal, usually overlapping. In most implementations, the FFT is used to analyze each frame.

A.1.2 Mel-frequency Cepstrum

The Mel-frequency Cepstrum is a representation of the frequency spectrum in terms of the mel scale. The mel scale is a different form of frequency representation, using mels instead of hertz, which was

developed to better represent how humans perceive frequency intervals. The coefficients of the Mel-frequency Cepstrum are the MFCC.

The computation of the MFCC of a digital signal is performed as follows. The DFT of each short window (frame) of the signal is first computed, and the frequency domain is transformed into the mel scale. The logarithm of the powers is computed, and the DFT for each frame is computed. The amplitudes of the final spectrum are the MFCC.

A.2 Metrics

A.2.1 Regression and Classification Metrics

In prediction models, the ground-truth value (target), t_i, of the sample data point i, is the real-world value that ideally the system should predict, while the output value, o_i, is the value predicted by the system for that sample. In binary classification problems, if a class is present, the class is said to be positive (value 1), otherwise the class is negative (value 0). It is useful to distinguish the different relations that can exist between target and prediction: true positive $(t_i = 1, o_i = 1)$; true negative $(t_i = 0, o_i = 0)$; false positive $(t_i = 0, o_i = 1)$; false negative $(t_i = 1, o_i = 0)$. The count of all true positives is denoted as TP, and similarly for true negatives (TN), false positives (FP), and false negatives (FN). We now define some frequently used metrics that compare the relations between the values of t_i and o_i.

The Root Mean Square Error (RMSE) is a metric used in regression to evaluate the general proximity of the predicted values to their target counterparts [44]. Better predictions have smaller values, zero meaning a perfect prediction model with regards to the considered samples, but the size of the RMSE can vary with the scale of the numbers.It is computed as:

$$RMSE = \sqrt{\frac{\sum_i (t_i - o_i)^2}{N}}, \tag{A.1}$$

where N is the sample size.

There are several metrics that evaluate classifiers based on the relation between target and output values [54]. For instance, the accuracy is the percentage of cases the model predicted correctly, according to:

$$accuracy = \frac{TP + TN}{FP + FN}. \tag{A.2}$$

A metric that measures the percentage of correct predictions in terms of the set that was considered positive by the model is the precision, which is defined as:

$$precision = \frac{TP}{TP + FP}. \tag{A.3}$$

Furthermore, the recall measures the percentage of the ground-truth positive samples that were in fact considered positive by the model:

$$recall = \frac{TP}{TP + FN}. \tag{A.4}$$

An additional metric that serves to evaluate the overall performance of the model is the F1-score, which is defined as the harmonic mean between precision and recall:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{A.5}$$

A.2.2 Distribution Analysis

A.2.2.A Pearson Correlation Coefficient

Pearson's correlation coefficient, ρ, is a metric of the linear correlation between two sets of data [55], and can be used to compare the ground-truth samples and their output counterparts according to:

$$\rho(\mathbf{t}, \mathbf{o}) = \frac{cov(\mathbf{t}, \mathbf{o})}{\sigma_{\mathbf{t}} \sigma_{\mathbf{o}}}, \tag{A.6}$$

given the target vector $\mathbf{t}$ which represents all the target values, $\mathbf{o}$ all the output values, σ is the standard deviation of a set of values, $cov(\mathbf{t}, \mathbf{o})$ is the covariance between the target and output values, and can be computed based on the average values, μ, according to $cov(\mathbf{t}, \mathbf{o}) = \mu_{\mathbf{t} \cdot \mathbf{o}} - \mu_{\mathbf{t}} \mu_{\mathbf{o}}$. The variables $\mathbf{t}$ and $\mathbf{o}$ can be replaced by any two variables to study their linear correlation.

A.2.2.B Spearman's rank correlation coefficient

The Spearman's correlation coefficient [56] is often used to evaluate to which degree the relationship between two variables corresponds to a monotonic function. To compute the coefficient, r_s, the ranks of the variables are used. For instance, the rank of the sample X_i of the variable X, $R(X_i)$, indicates the position in which the sample appears if all the unique X samples were ordered from lowest to highest. Thus, r_s is computed as the Pearson correlation coefficient between the ranks regarding each variable, $R(X_i)$ and $R(Y_i)$, according to:

$$r_s = \rho(R(X), R(Y)) = \frac{cov(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}}, \tag{A.7}$$

where ρ is computed using Equation (A.6), cov is the covariance between the ranks, and σ denotes the standard deviations of the variables.

A.2.2.C Bhattacharyya distance

The Bhattacharyya distance [57], D_{BC}, is a measure of similarity between two probability distributions. In the case of two discrete probability distributions, p and q, defined over the same domain X, the Bhattcharyya distance between the two, $D_{BC}(p, q)$, is given by:

$$D_{BC}(p, q) = -\ln\left(\sum_{x \in X} \sqrt{p(x)q(x)}\right). \tag{A.8}$$

A.3 Models

A.3.1 Decision Tree

A DT is a model that determines an output based on a series of tests that can be thought of as nodes from a tree [54]. Each test evaluates one of the features of the sample at hand, and the result of the analysis of the feature dictates the next test in the decision process. This reasoning goes on until a leaf node is reached, which must be a decision regarding what output to provide.

A.3.2 Linear Regression

If it is estimated that there is a linear relationship between the features of a sample, $\mathbf{X}$, and the target value, then a Linear Regression model is appropriate to represent such a relationship. In a linear model, the outputs are computed based on a set of weights, $\mathbf{w}$, and a bias, b, according to $o = \mathbf{x} \cdot \mathbf{w} + b$, and the task of computing the weights $\mathbf{w}$ and b is known as Linear Regression (LR) [54].

A.3.3 Support Vector Machines

SVM can be used as classification models, as they look for the maximum margin separators (hyperplanes) that separate the different classes [54]. The models can solve non-linear problems by first transforming the feature space. The same family of models can be used for regression in the form of SVR models, where the output is dictated by the hyperplanes themselves.

A.3.4 Gaussian Mixture Models

A Gaussian Mixture Model (GMM) is formed by a linear combination of Gaussians [45]. In this model, a single Gaussian distribution is considered a mixture component, and the weight of each Gaussian is

called the mixing coefficient. By associating each Gaussian component with a class, these models can be used as classification models, by evaluating to which component the sample has a higher likelihood of association.

A.3.5 Neural Networks and Multilayer Perceptron

A neuron is a computing unit inspired by the way neurons process information in our brains [54]. It represents a mapping from real numbers taken as input to a real number output, and produces an output based on a linear combination of the inputs defined by a vector of weights, $\mathbf{w}$, a bias, b, and an activation function, $f(\cdot)$. An output o is computed from the input $\mathbf{x}$ according to $o = f(\mathbf{x} \cdot \mathbf{w} + b)$.

Several neurons can be combined to produce an artificial neural network, where the output of some neurons is passed as the input to others. An MLP is a network composed of perceptrons with at least one hidden layer of neurons (a layer located between the input and output nodes).

A CNN is a neural network specialized in the identification of translational spatial characteristics in images [58]. It deploys mathematical operations called convolutions, which are a specialized kind of linear operation, in at least one of its layers. The parameters of a convolutional layer are a set of learnable filters (also called kernels). A convolution is performed by passing the kernel across the input of the layer, while computing the dot product between the input and the kernel entries. As a result, the network learns filters that activate when it detects a learned image feature at some spatial position in the input.

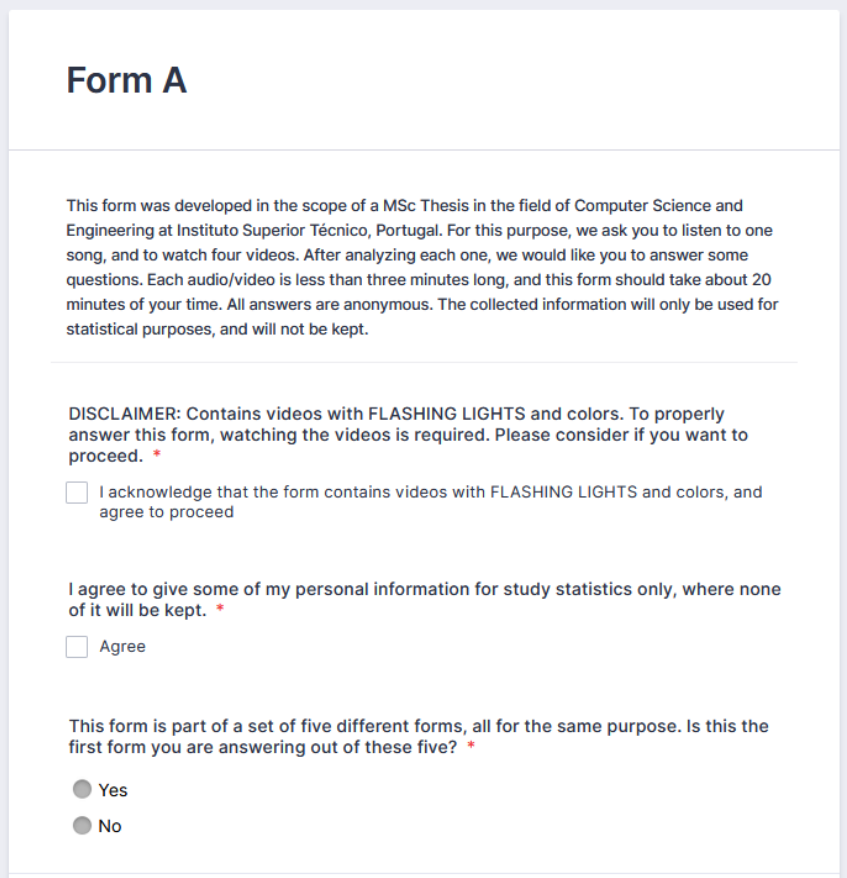
A.4 Data Preprocessing

A.4.1 Principal Component Analysis

PCA is often used for dimensionality reduction, such that the features are transformed into a new space with fewer dimensions before being used by a model (giving rise to the possibility of information loss) [44]. When PCA is performed, the features undergo an orthogonal projection onto the principal subspace, which is a linear space with fewer dimensions than the original feature space, and is determined by maximizing the variance of the projected data.

Appendix B

Form Example



Form A

This form was developed in the scope of a MSc Thesis in the field of Computer Science and Engineering at Instituto Superior Técnico, Portugal. For this purpose, we ask you to listen to one song, and to watch four videos. After analyzing each one, we would like you to answer some questions. Each audio/video is less than three minutes long, and this form should take about 20 minutes of your time. All answers are anonymous. The collected information will only be used for statistical purposes, and will not be kept.

DISCLAIMER: Contains videos with FLASHING LIGHTS and colors. To properly answer this form, watching the videos is required. Please consider if you want to proceed. *

☐ I acknowledge that the form contains videos with FLASHING LIGHTS and colors, and agree to proceed

I agree to give some of my personal information for study statistics only, where none of it will be kept. *

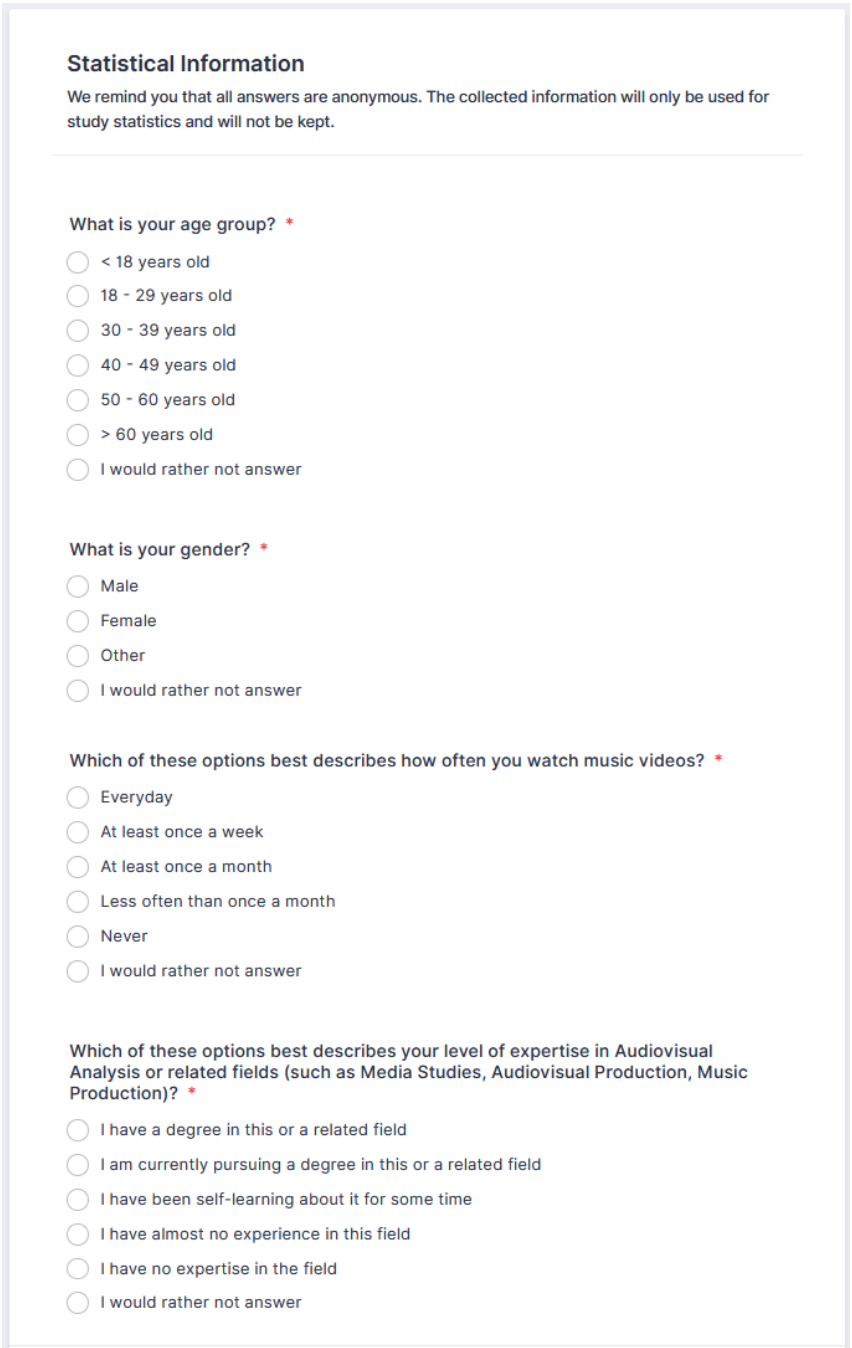
☐ Agree

This form is part of a set of five different forms, all for the same purpose. Is this the first form you are answering out of these five? *

○ Yes

○ No

Figure B.1: Front Page.



Statistical Information

We remind you that all answers are anonymous. The collected information will only be used for study statistics and will not be kept.

What is your age group? *

- < 18 years old
- 18 - 29 years old
- 30 - 39 years old
- 40 - 49 years old
- 50 - 60 years old
- > 60 years old
- I would rather not answer

What is your gender? *

- Male
- Female
- Other
- I would rather not answer

Which of these options best describes how often you watch music videos? *

- Everyday
- At least once a week
- At least once a month
- Less often than once a month
- Never
- I would rather not answer

Which of these options best describes your level of expertise in Audiovisual Analysis or related fields (such as Media Studies, Audiovisual Production, Music Production)? *

- I have a degree in this or a related field
- I am currently pursuing a degree in this or a related field
- I have been self-learning about it for some time
- I have almost no experience in this field
- I have no expertise in the field
- I would rather not answer

Figure B.2: Participant characterization.

Please listen to the following audio: Autechre - plyPhon.

Audio characterization

Describe the previous audio with at least one adjective/expression (use each field to provide one adjective/expression).

1. *

2.

3.

Figure B.3: Audio characterization through unrestrained descriptions.

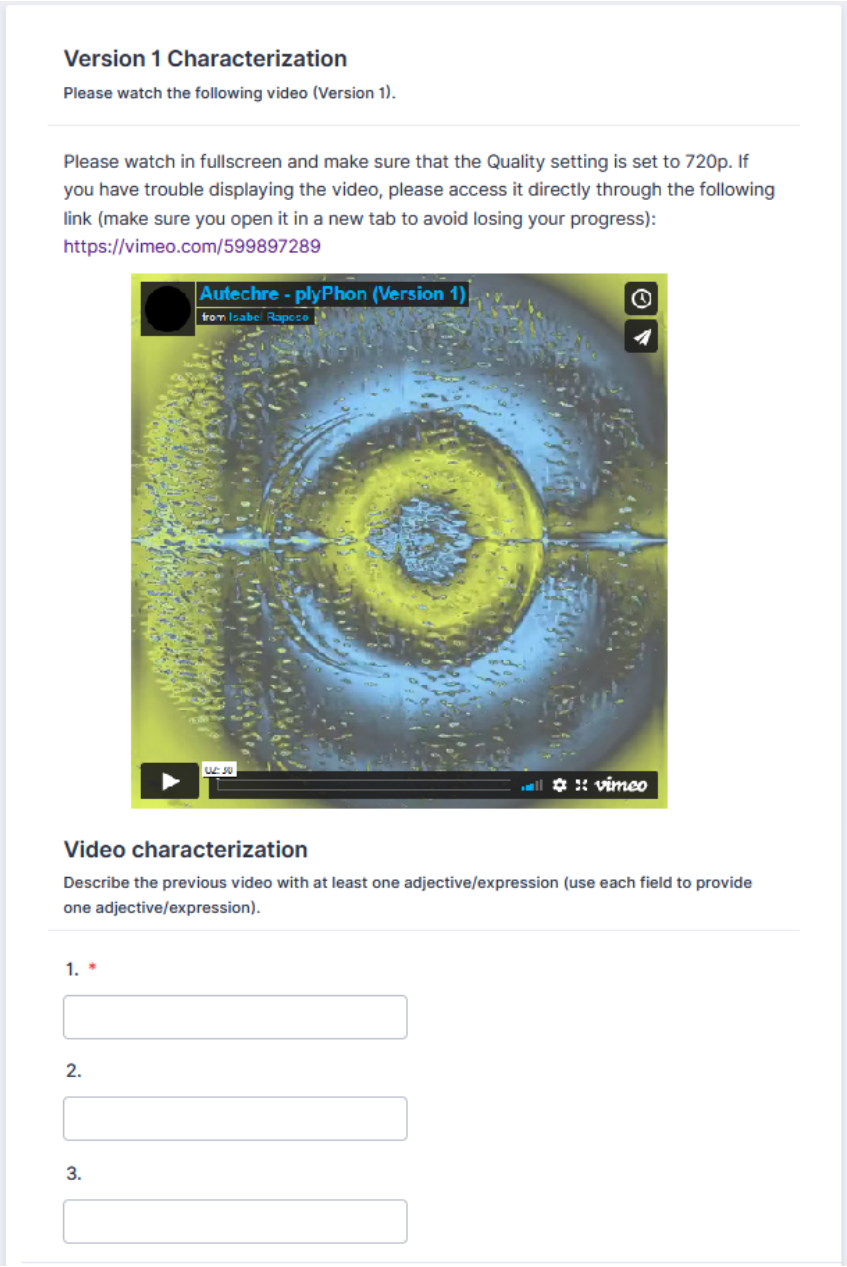


Figure B.4: Example of video characterization through unrestrained descriptions, for Version 1.

Regarding the Audio:

Here is the audio again (Autechre - plyPhon) for reference.

Audio characterization

Please indicate how much you agree or disagree with each of the following statements regarding the audio using the following scale: 1 - Strongly Disagree, 2 - Disagree, 3 - Undecided, 4 - Agree, 5 - Strongly Agree.

"I like it a lot." *

1 2 3 4 5

Strongly Disagree — Strongly Agree

"I consider it very creative." *

1 2 3 4 5

Strongly Disagree — Strongly Agree

Figure B.5: Audio characterization through Likert scale questions.

From the following adjectives, choose all that you think apply to the audio. *

- [ ] Exciting
- [ ] Calm
- [ ] Happy
- [ ] Enjoyable
- [ ] Boring
- [ ] Surprising
- [ ] Sad
- [ ] Aggressive
- [ ] Funny
- [ ] Interesting
- [ ] Disgusting
- [ ] Fearful
- [ ] Confusing
- [ ] Numb
- [ ] Tender

Figure B.6: Audio characterization through adjective selection.

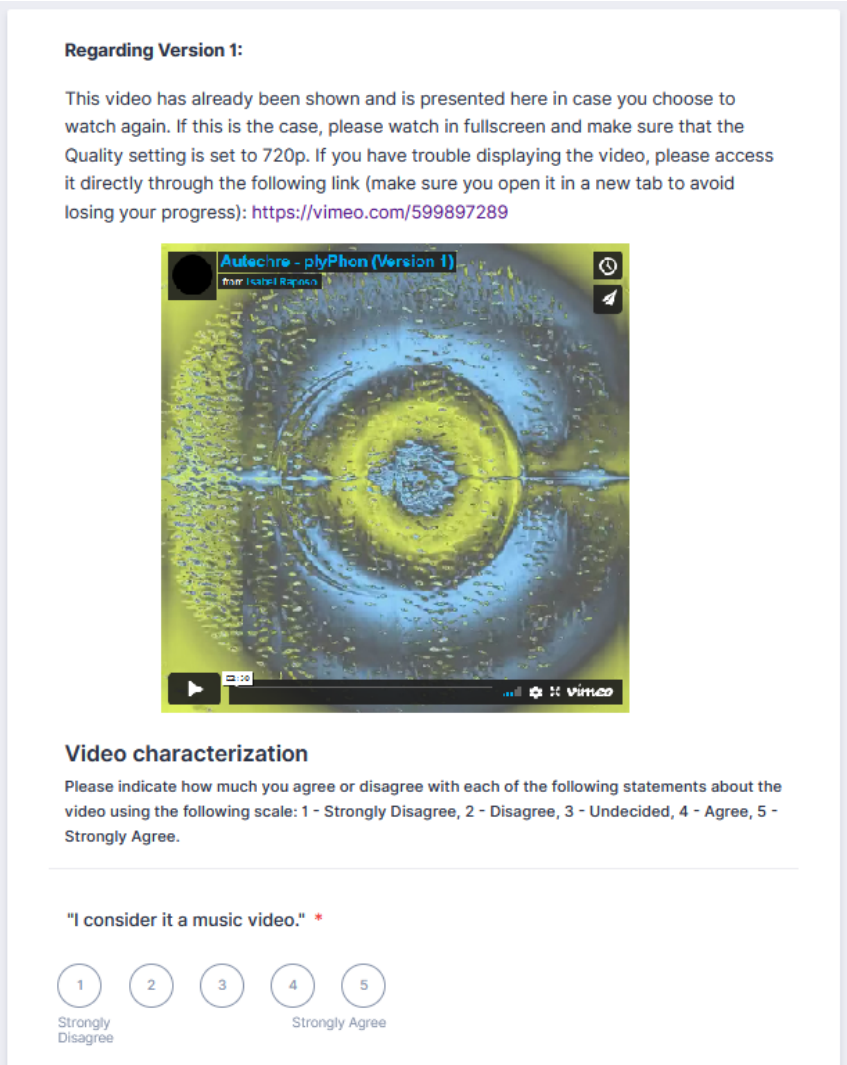


Figure B.7: Video characterization through Likert scale questions. The statements presented were all in this format, and were the following: “I consider it a music video”, “I like it a lot”, “I consider it very creative”, “I consider it abstract”, and “I think the visuals go well with the audio”. After these questions, a question regarding video characterization through adjective selection is also presented, using the same interface as in Figure B.6.

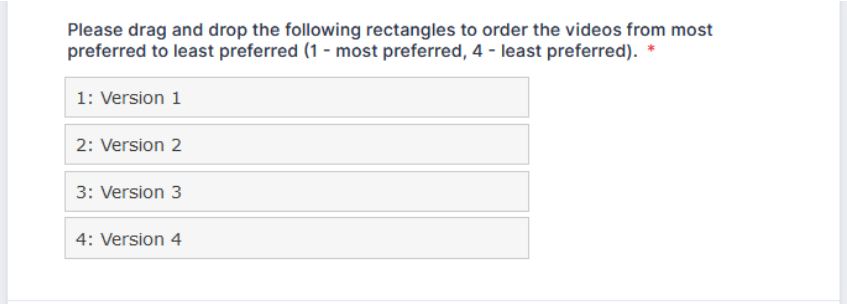


Figure B.8: Video preference ordering interface. All videos versions were available above this question, for reference.

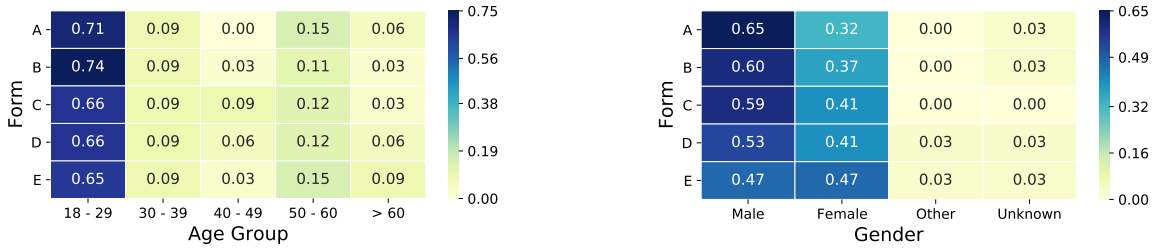
Appendix C

Additional Evaluation Statistics

In this Appendix, we present a few additional evaluation statistics obtained from the evaluation process of the videos produced by the system, that complement the analysis presented in Chapter 7.

Form	Pos.	Audio	Videos w/o Control	Manual	Preference	Mood	Control
A	1º	Aggressive	Interesting	Interesting	Interesting	Interesting	Boring
	2º	Interesting	Aggressive	Exciting	Confusing	Aggressive	Confusing
	3º	Surprising	Exciting	Surprising	Surprising	Exciting	Enjoyable
B	1º	Calm	Calm	Enjoyable	Calm	Enjoyable	Aggressive
	2º	Enjoyable	Enjoyable	Calm	Enjoyable	Interesting	Confusing
	3º	Tender	Interesting	Interesting	Interesting	Aggressive	Interesting
C	1º	Calm	Calm	Calm	Calm	Sad	Calm
	2º	Sad	Interesting	Interesting	Sad	Confusing	Happy
	3º	Enjoyable	Sad	Enjoyable	Enjoyable	Calm	Enjoyable
D	1º	Calm	Calm	Calm	Calm	Calm	Confusing
	2º	Sad	Interesting	Interesting	Interesting	Interesting	Boring
	3º	Interesting	Enjoyable	Enjoyable	Confusing	Enjoyable	Surprising
E	1º	Exciting	Exciting	Exciting	Enjoyable	Exciting	Boring
	2º	Happy	Interesting	Enjoyable	Interesting	Interesting	Calm
	3º	Enjoyable	Enjoyable	Interesting	Exciting	Enjoyable	Enjoyable

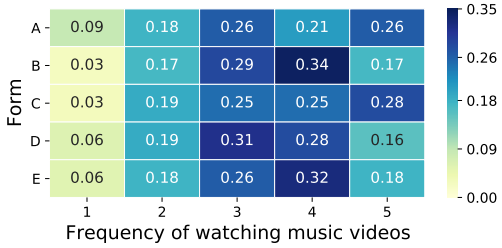
Table C.1: Top 3 chosen feelings to characterize the audio and video animations per form.



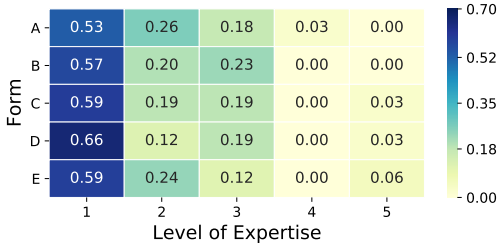
(a) Age group distribution per form

(b) Gender distribution per form

Figure C.1: Distributions per form of the answers provided to the questions: fig. C.1(a) "What is your age group?"; fig. C.1(b) "What is your gender?".



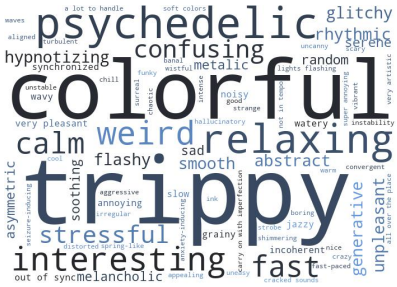
(a) Frequency of watching music videos distribution per form. **1** - Never, **2** - Less often than once a month, **3** - At least once a month, **4** - At least once a week, **5** - Everyday.



(b) Level of expertise distribution per form.**1** - No expertise, **2** - Almost no experience, **3** - Currently self-learning, **4** - Currently pursuing a degree, **5** - Has a degree.

Figure C.2: Distributions per form of the answers provided to the questions: fig. C.2(a) "Which of these options better describes how often you watch music videos?"; fig. C.2(b) "Which of these options better describes your level of expertise in Audiovisual Analysis or related fields (such as Media Studies, Audiovisual Production, Music Production)?".

(a) Audio



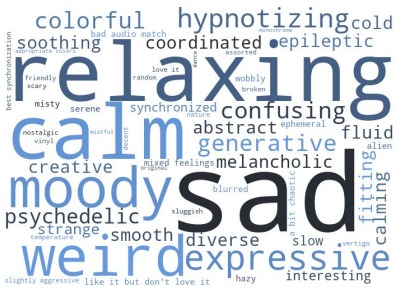
(b) Videos w/o Control

Source	Top 3 Descriptors
Audio	relaxing, calm, smooth
Videos w/o Control	trippy, colorful, psychedelic
Manual	trippy, interesting, relaxing
Preference	trippy, rhythmic, relaxing
Mood	psychedelic, flashy, colorful
Control	hypnotizing, dark, aggressive

(c) Top 3 Descriptors

Figure C.3: Word Clouds of the descriptions provided for Form B: *Chet Baker - "It Could Happen To You"*

(a) Audio



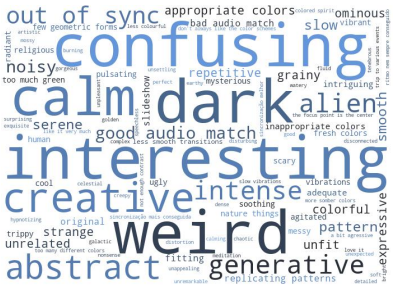
(b) Videos w/o Control

Source	Top 3 Descriptors
Audio	sad, calm, melancholic
Videos w/o Control	sad, relaxing, calm
Manual	relaxing, calm, hypnotizing
Preference	relaxing, sad, weird
Mood	sad, relaxing, calm
Control	colorful, happy, energetic

(c) Top 3 Descriptors

Figure C.4: Word Clouds of the descriptions provided for Form C: *Chopin - "Prelude In E Minor (opus 28, nº 4)"*

(a) Audio



(b) Videos w/o Control

Source	Top 3 Descriptors
Audio	religious, calm, relaxing
Videos w/o Control	confusing, interesting, weird
Manual	interesting, good audio match, calm
Preference	noisy, out of sync, creative
Mood	weird, confusing, dark
Control	colorful, bright, hypnotizing

(c) Top 3 Descriptors

Figure C.5: Word Clouds of the descriptions provided for Form D: *Coro Madrigale Slovenico - "Sanctus"*

(a) Audio

(b) Videos w/o Control

Source	Top 3 Descriptors
Audio	happy, cheerful, uplifting
Videos w/o Control	colorful, interesting, energetic
Manual	colorful, interesting, funny
Preference	colorful, flashy, psychedelic
Mood	bright, confusing, energetic
Control	boring, slow, out of sync

(c) Top 3 Descriptors

Figure C.6: Word Clouds of the descriptions provided for Form E: *The Beatles - "Help!"*

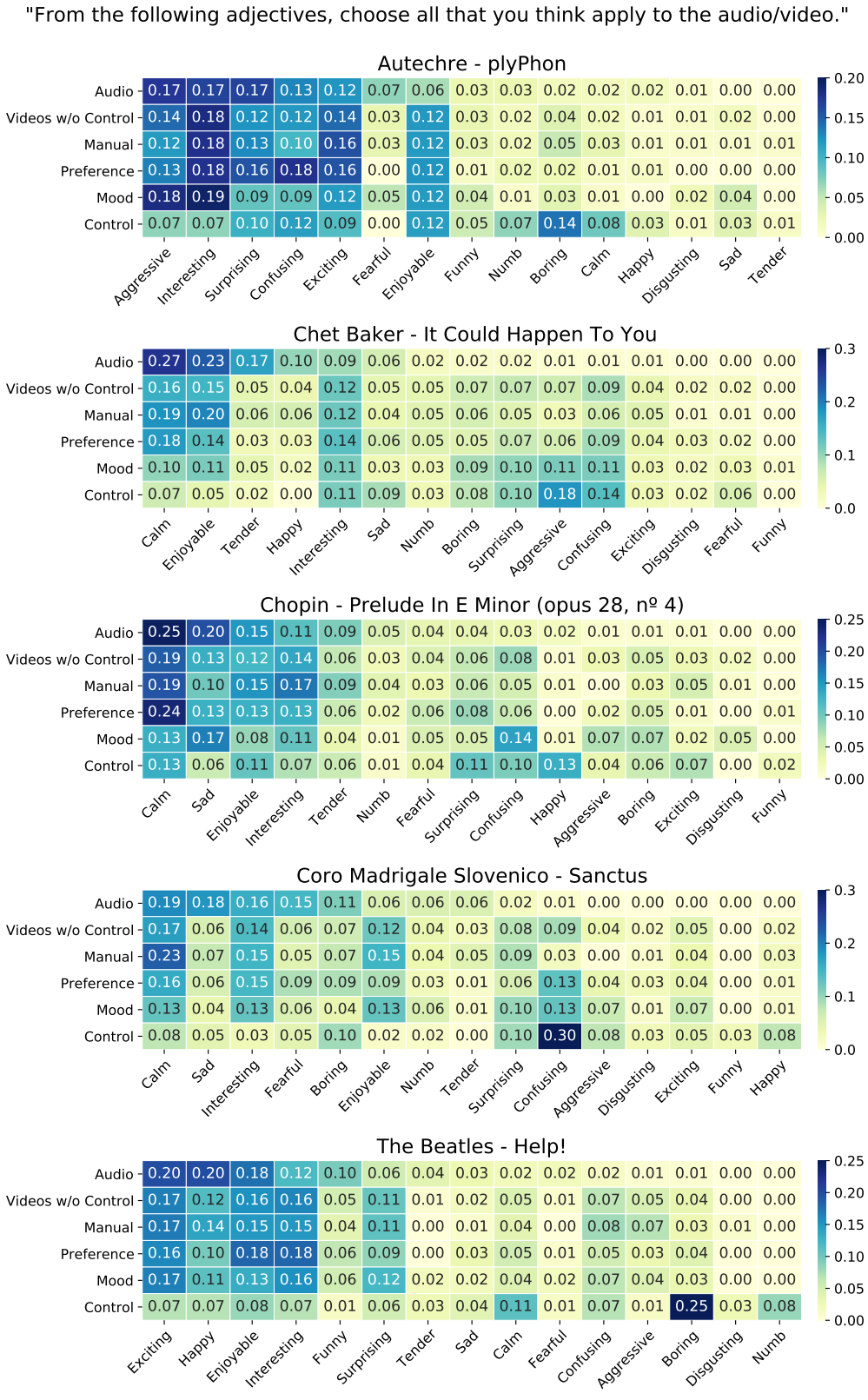


Figure C.7: Adjective distribution for each audio, the videos without considering the control one (Videos w/o Control), and each video in particular. Each graph has the adjectives sorted from left to right according to their frequency in the audio distribution.

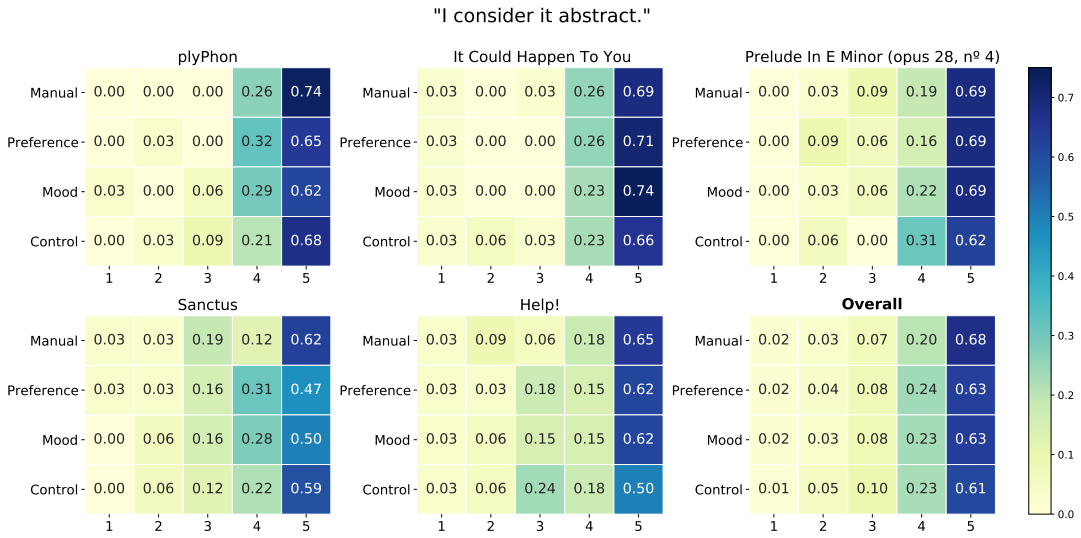


Figure C.8: Answer distributions regarding the Likert scale question “I consider [the video] abstract”. The answers go from 1 - Strongly Disagree, to 5 - Strongly Agree.

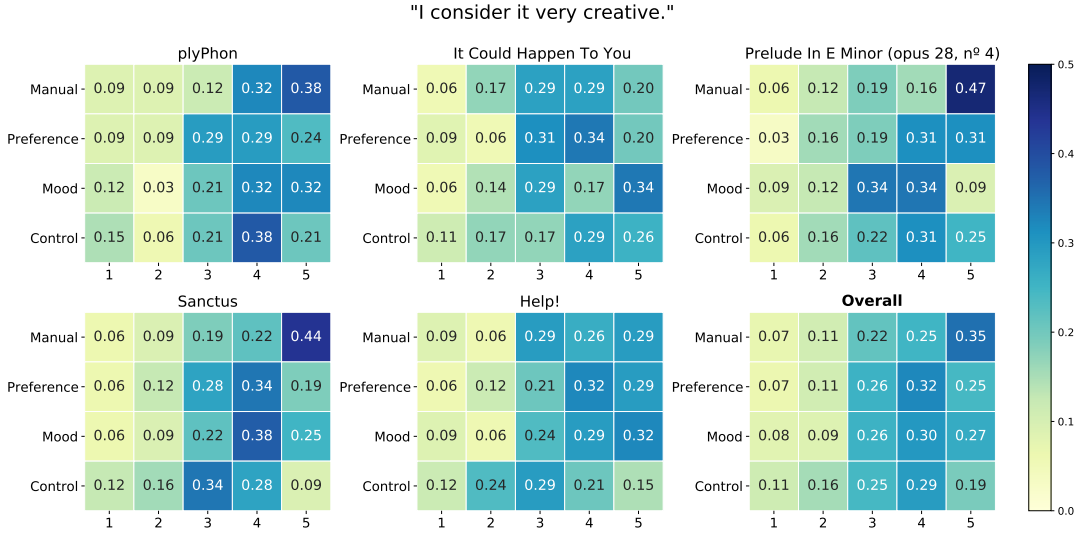


Figure C.9: Answer distributions regarding the Likert scale question “I consider [the video] very creative”. The answers go from 1 - Strongly Disagree, to 5 - Strongly Agree.

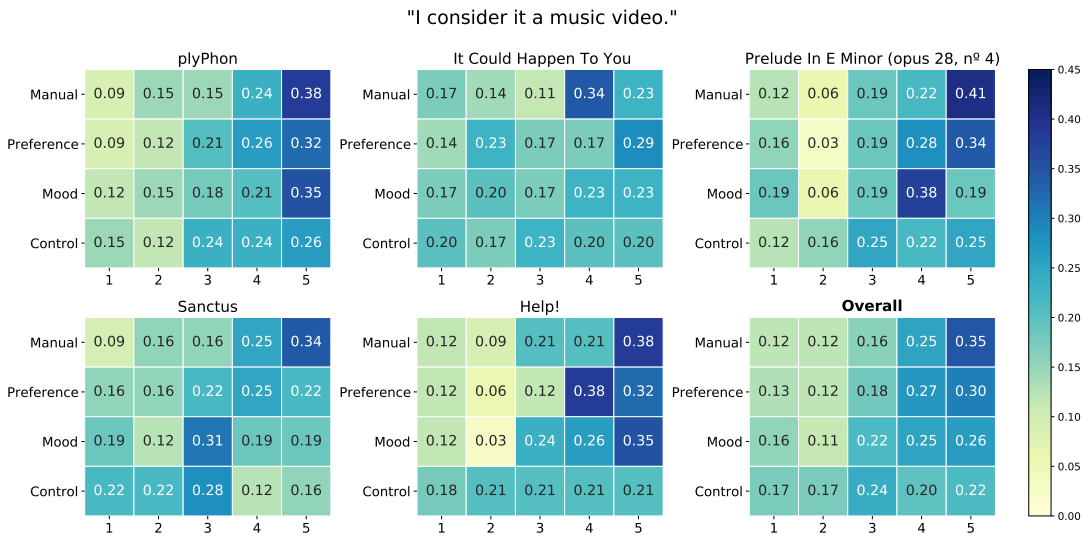


Figure C.10: Answer distributions regarding the Likert scale question “I consider [the video] a music video”. The answers go from 1 - Strongly Disagree, to 5 - Strongly Agree.