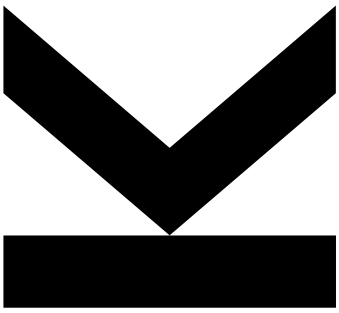
Eingereicht von
Anna Maria Piessenberger

Betreuer
Dr. Ismail Khalil

Web-Technologies for Artists

SoVia - Sound Visualization

05 2021



Bachelorarbeit

Institute of Telecooperation

1. Motivation and Problem Statement

Sound surrounds us almost always. It influences, how we perceive situations and environments. We use ambient sound to analyse situations and for being aware of our surroundings. Maintaining this awareness is difficult for people who are deaf or have hearing impairments. They might not be able to enjoy the many facets and emotions that audio signals can transport and generate. How about approaching audio from another perspective by translating audio content into visual content in order to make it accessible for the deaf and hard hearing? In this thesis, an approach named SoVia is introduced, to make audio accessible from another perspective was evaluated. Artistic artefacts are created from audio signals to picture the current ambient sounds, so that someone get a sense what is currently going on in his or her environment.

SoVia is a single page web application written in Javascript, with the sub goal of finding suitable web-technologies and Javascript-Libraries that can be used in an artistic way. Another goal of this thesis was to combine a technological aspect with an artistic one. First the audio signal is analysed and characteristics such as pitch, volume and 'change rate' are extracted and calculated. Those characteristics are used to categorize an emotion.
"Real" emotion detection would have gone beyond the scope of the Thesis and is not easy to implement in Javascript. With the Information extracted from the audio signal lines are drawn, varying in colour, shape and smoothness. The user can draw such lines by either interacting with the web application directly or by using the automatic mode that draws random lines considering the ambient sounds using their characteristics.

The challenge was to generate images someone can assign to a particular sound scenery. To evaluate how well that was working, a short survey was done in which people were asked to assign images to audio samples.

As a result of the stated problem above this thesis should answer the following questions:
How can audio streams be analysed so that emotions can be approximated?
Is pre-processing of the audio stream necessary?
Which algorithms can be used?
Which technologies can be used to analyse audio signals?
How can one visualize an audio signal?

2. Related Work

In the field of assistive technology research for the deaf and hard hearing community the focus is set on supporting verbal communication, such as automatic sign language recognition and speech therapy using technology to giving deaf and hard hearing people the possibility to practice verbal communication [2,3]. There is also a technique introduced, where speech therapists are trained to interpret waveforms and spectrograph visuals. [4] To use this method properly, experts in the field of audio processing are needed. Unlike these previous researches, this thesis focuses on a representation of non-verbal sounds that can be interpreted by non-expert users.

To notify deaf and hard hearing people of acoustic events like the fire alarm going off, the phone or doorbell ringing or the alarm clock going off in the morning, products have been developed that use flashing light or vibrations as an alternative to sound. The disadvantage of this approach at making audio signals accessible for people with hearing impairment is that for every specific acoustic event an extra tool has to be used. This might become expensive and confusing quickly. A sub goal of this thesis is, to visualise all acoustic alerts in one application.

“Audio accessibility has been neglected but is an important area of research”. [0] This is stated in the paper written by Collins and Taillon who have been investigating accessibility of sound for hearing impaired people. They introduced an approach, where they use, as they call it, SoundSigns to communicate sounds to people with hearing impairment while playing a video game. After asking people how they feel about the introduction of SoundSigns in Shooter Games, the feedback was consistently good and showed that there is a need for sound visualisation. [0]

A more artistic approach to visualizing sound was exhibited 2009 in Linz Austria at the “See This Sound” exhibition that took place in the art museum Lentos. The exhibition aimed to make several important aspects of the manifold and diverse relationship between image and sound accessible in an interdisciplinary way. One exibit for example was from Jörg Jewanski, dealing with the relationship between color and tone. [5]. SoVia should be also seen as an artistic approach to generate visual artefacts from ambient sounds.

In her paper “Visual Music - A Composition Of The Things Themselves” [6], Maura McDonnell gives some interesting examples of ways to visualize music. Javier Sanchez’s video Unorthogonality (2009) for example, is using simple lines to explore spatial and musical

movement. Another example is the work 1/3 (one over three) vol.1 (2006) of Chiaki Watanabe, she experiments with one-bit as an art expression based on one-bit technology in her computer images for live performance.

Recently another paper in this field has been published [11]. ViTune is a visualizer tool that enhances the musical experiences of the deaf and hard hearing through the use of an on-screen visualizer generating effects alongside music. The black screen of the web application is divided vertically into twelve rectangular columns, mapped to the colour spectrum (red to violet) from left to right. Each column contains smaller visual elements: two smaller rectangles and a star. These twelve columns represent the frequencies, falling in one of the twelve pitch classes (C to B) present in the music at the current time. ViTune was developed as a prototype by asking members of the deaf and hard hearing community how they felt while watching the visualization of the music.

3. Implementation

3.1. Technology Stack

3.1.1. Programming Language

Since one of the goals of this thesis is to provide the implementation as a Web Application, JavaScript has been chosen to be the programming language used. The Web Application is implemented as a Single Page Application, so that the page is not reloaded entirely every time something gets updated. Several Javascript libraries are used to implement SoVia in a clean and modern way. Web Audio API, ML5.js, Raphael.js, Meyda.js and P5.js are the main protagonists in this project. The following section describes them in more detail and how they are used.

3.1.2. Infrastructure

As a versioning software, Git was used. This is also where you can find the whole implementation and additional material – **annamiaz/DrawingWithEmotions**. Additionally, Git made it easier to try different approaches by using different branches. Also for sharing reasons it is more practical to keep the code on GitHub. For a better UI support while working with GitHub, Github Desktop was used, directly connected with Atom. Atom is a text editor, build with HTML, JavaScript, CSS and Node.js integration, running on Electron, a framework for building cross platform apps using web technologies. [12]

Additionally, the Atom package atom-live-server was used to launch a HTTP-Server, to make easier to test the developed code.
As the default browser while developing, the Google Chrome Browser was used. Therefore, the application should at least run on this Browser without any problems.
To keep the code clean and structured, the Atom Extension Atom Beautify was used. For a direct connection to the Git Repository in the Atom editor, the Github extension for Atom was also added.

3.1.3. Web Audio API

One of the first major achievement was to record a sound in real time via a microphone using the WebAPI AudioContext. This API is supported by most Browsers but unfortunately not by Safari which was used at the very beginning of this project and also not by Internet Explorer.
If you want to use the AudioContext in Safari, you have to use a specific prefix to make it compatible to this Browser. If you want to know which Browser, in which version supports Web Audio API you can query it from the CanIUse Database [29].
After noticing this restriction, Google Chrome was used as the default Browser version for this Web Application. Setting the right permissions for the audio input in the browser, the audio was recorded correctly.
As it says in the documentation of the Web Audio API [7]

> *"The Web Audio API provides a powerful and versatile system for controlling audio on the Web, allowing developers to choose audio sources, add effects to audio, create audio visualizations, apply spatial effects (such as panning) and much more."*

This can be done by using the so-called audio context. Within this context you can do audio operations and modify the signal by adding audio nodes into an audio routing graph. Every node is able to have multiply sources that are connected by their inputs and outputs. Sources are arrays with samples made in very small-time intervals leading to tens of thousands of entries per second. These samples can be computed or can be used as recordings from sound or video.
The documentation states as an example, that this library can be used to create advanced and interactive instruments. Therefore, it is suitable for developers as well as for musicians and other creatives.
Since ML5.js recommends the AudioContext from Web Audio API for the Pitch estimation module, Web Audio API is used for this implementation. Of Course,
there are some alternatives to mention: For example, howler.js which as it says on their Github page, is an audio library made for modern web applications. Default using Web Audio API and falls back to HTML5 Audio. This makes it easy to use and reliable across all platforms. [10].

3.1.4. ML5.js

Ml5.js is an open source project that enables machine learning for the web in the browser. It is developed and maintained by NYU's Interactive Telecommunications/Interactive Media Arts program and also by artists, designers, students, technologists and developers from all over the world. [10]

The main goal of ml5.js is to provide Machine Learning for a broad audience in an approachable manner, so that artists, creative coders and students can use them easily. It can be used to access pre-trained models in the browser that are for instance able to detect human poses, generate text, transfer styles, compose music or as used in SoVia to detect the pitch in audio signals. It uses TensorFlow.js in the background with no other dependencies. It could be compared with Processing or P5.js – which is also part of this project. The community around ml5.js emphasis ethical computing, which is reflected in the provided code examples, tutorials and sample datasets. Bias in data, stereotypical harms, and responsible crowdsourcing are part of the documentation around data collection and usage. The community also has a common statement, which is similar to the p5.js community statement:

> *"We are a community of, and in solidarity with, people from every gender identity and expression, sexual orientation, race, ethnicity, language, neuro-type, size, ability, class, religion, culture, subculture, political opinion, age, skill level, occupation, and background. We acknowledge that not everyone has the time, financial means, or capacity to actively participate, but we recognize and encourage involvement of all kinds. We facilitate and foster access and empowerment. We are all learners."*

Since they aim for projects in the area of artistic web applications and because they provide exactly what is necessary for this project, the pitch estimation is done via a model from ml5.js. ML5.js is integrated via CDN (Content Delivery Network) link in the project. How the pitch estimation is done in detail is discussed the "Pitch estimation" section.

3.1.5. P5.js

Similar to ml5.js, p5.js is an Open Source JavaScript library, developed to make JavaScript code more accessible and can therefore be used by more different people, also including artists, designers and beginners. [14] P5.js comes with a full set of drawing functionality. But it is not only about drawing, there are many other functions to work with, like images, text, sound and video. The community around p5.js have the same mind-set as the community around ml5.js. Starting with p5.js is quick and easy, there are many tutorials where the basics are explained. P5.js comes with a separate Editor which can be directly used in the browser. But that was not an option for this project.

3.1.6. Raphael.js

Since it is not possible to draw SVG graphics with p5.js yet, Raphael is used for the implementation of drawing lines. Raphael is a lightweight JavaScript library that focuses on working with vector graphics on the web. As a base for creating graphics Raphael uses the SVG W3C Recommendation and Vector Markup Language. Every graphical object created, is also a DOM element, so it is possible to attach JavaScript event handlers or modify them later on. With Raphael, graphics are compatible with most browsers. [13]

3.1.7. Meyda.js

Meyda is a Javascript library aiming for Audio feature extraction. Meyda implements a selection of standardized audio features that can be used for audio signal computing. It can listen to audio and give a selection of statistics that describe the input signal. For this project, the time domain feature energy is used, which is described as the infinite integral of the squared signal. [27, 28]

This library is provided, similar to ML5.js via a CDN (Content Delivery Network) link in this project. Meyda could be very useful for future improvements for this implementation since it provides many computation methods for interesting sound characteristics that could be meaningful for this area of application.

3.1.8. Architecture

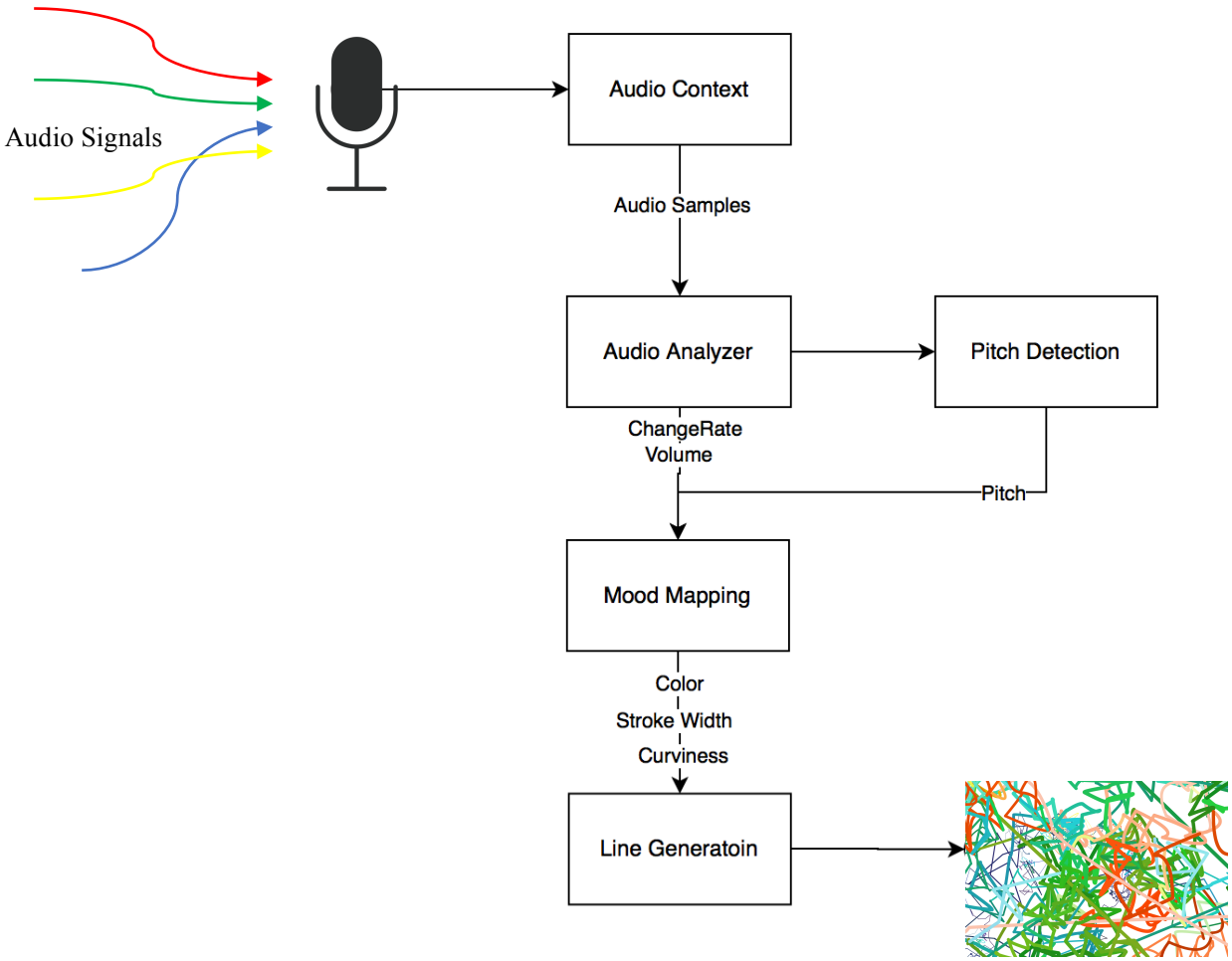


Figure 1: Architecture Data Flow

As you can see in Figure 1 above, SoVIa is build from on five different components. Each component is part of the transformation from an audio signal to a line-based drawing. The start of this system is the built in microphone of your laptop, or whatever microphone is used in your browser. The audio signals are recorded and collected in the Audio Context node provided from the Web Audio API. This Audio Context is then propagated to the Pitch Detection model, from ML5.js, provided in a CDN. At the same stage, volume and change rate are calculated from the Audio Context in the Audio Analyzer Node. With pitch, change rate and volume as input parameters, the mood is calculated in the Mood Mapping Node. The mood should represent the parameters from which the line can be generated in the next steps – color, curviness and stroke width are propagated to the final module – the line generation. In this step, the concrete lines are drawn with Raphael.js as SVGs. In the next section, the components are explained in detail.

3.2. Graphical User Interface

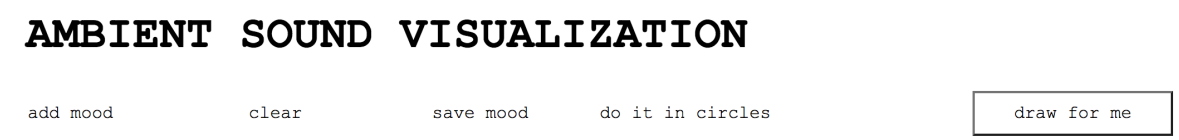


Figure 2: Web Page GUI

The GUI, as shown in Figure 2 is kept simple. The button “add mood” triggers the emotion detection. The “draw for me”-Button starts the algorithm to automatically generate paths. Clearing the canvas or paper as it is called in Raphael context, is triggered with the “clear”-Button. With the “save mood” button, a SVG file is generated and downloaded. The “do it in circles”-Button generates circles instead of lines – but is yet only available for the manual drawing mode also available in a dark mode. Just a little extra gimmick.
For styling the GUI as wanted, an extra CSS file was used.

3.3. Components

3.3.1. Audio Context

To work with Web Audio API, first of all you need an Audio Context Node. The source of the Audio Context Node in this case is the microphone configured in the browser. It provides an array of sound samples which are used later on for computing the necessary audio characteristics.

```
audioContext = new AudioContext();
stream = await navigator.mediaDevices.getUserMedia({
    audio: true,
    video: false
  });
```

In the code sample above, a AudioContext is created and used to set up an AnalyserNode. The AudioContext and the stream are needed for the Pitch Detection, the analyzer is used for calculating other characteristics like energy of the signal or change rates. The audioArray holds the values from the Fast Fourier transformation. In the context of this project, the AudioContext is just used as a recorder, there is no output node necessary.

3.3.2. Audio Analyzer

The Audio Analyzer Component calculates audio characteristics like energy, change rate of the energy and change rate of the pitch. Energy and pitch are stored in queues of 20 samples each. This is done because it flattens the audio signal – so it is more harmonic and outliers do not influence the evaluation of the mood.

Energy

The energy is calculated via Meyda.js with the time domain feature Energy which is calculated with the infinite integral of the squared signal according to Lathi [27, 28]

The energy E_g of a signal $g(t)$ is defined as the area under $|g(t)|^2$. The signal energy can also be determined from its Fourier transform $G(\omega)$ through the Parceval's theorem:

$$E_g = \int_{-\infty}^{\infty} g(t)g^*(t)\,dt = \int_{-\infty}^{\infty} g(t)\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} G^*(\omega)e^{-j\omega t}\,d\omega\right]dt$$

Further the energy has to be scaled to fit it to the stroke width of the line.

Harmony

The harmony parameter is set to true if the pitch stays the same for a while, so the line should not change directions or do something crazy, instead the length of the paths that are drawn is reduces steadily until it settles at a point.

Change Rate

If the energy of the sound varies a lot, the sound is more furious, and therefore it should be take into account in the mood mapping. The change rate is calculated for the pitch and the energy of the signal as follows:

$$\sum_{i=0}^{18} |q_i - q_{i+1}|$$

Where q is a sample stored in the energy queue respectively in the pitch queue.

Only the change of two consecutive energy values are considered in the calculation.

3.3.3. Pitch Detection

Before we start to explain the used methodology of the pitch detection, it is necessary to discuss briefly the terms fundamental frequency and pitch. The fundamental frequency is a physical property of an audio signal. It is defined as the lowest frequency of a periodic waveform. Meanwhile, pitch is defined as a subjective quality of perceived sounds and does not precisely correlate to fundamental frequency. However, apart from a few rare exceptions, pitch can be quantified using fundamental frequency, and thus both terms are used for the same purpose outside psychoacoustical studies. [15] For convenience, these two terms are used interchangeably in this thesis.

The pitch detection algorithm estimates the pitch present in the audio signal.
In this project, the pitch detection is done by the pre-trained ml5.js Pitch Detection Model which is provided in an CDN (Content Delivery Network) as an external resource. A connection to the internet is necessary for executing this web application properly. It needs the Audio Context and the stream of the microphone mentioned above in the Audio Context section.
For the Pitch estimation, the ml5.js model uses CREPE: a convolutional representation for pitch estimation. [15] CREPE is a data-driven pitch tracking algorithm, which is based on a deep convolutional neural network that operates directly on the time-domain waveform. A 1024-sample extract from the time-domain audio signal with 16 kHz sampling rate is used as the input of the deep CNN. There are six convolutional layers that result in a 2048-dimensional latent representation, connected closely with the output layer with sigmoid activations corresponding to a 360-dimensional output vector. This result vector is used to calculate the pitch estimation deterministically.
The Algorithm has some restrictions, but works better in general than similar algorithms like pYIN [16] and SWIPE [17]. One of the problems while detecting the fundamental frequency of an audio signal is noise. CREPE works best if there is white noise present, pink noise works almost equally good for pYIN and CREPE, only brown noise is more challenging for CREPE then for the other algorithms. Because of that reason, SoVia does not perform in the expected way for audio samples containing pink or brown noise. Situations with heavy rain or the sound of the ocean are harder to work with for this application.

3.3.4. Mood Mapping

In SoVia, the focus was on detecting and visualizing the emotions sad, neutral, angry and happy. For the mood mapping the calculated parameters pitch, change rates and pitch energy are available. Fortunately it is possible to get emotions present in a signal by taking the pitch into account. The following table (Table 1) should give an overview and explain how the pitch can categorize audio signals in the 3 emotions of anger, happiness and sadness. As you can see, if a high pitch is detected and it changes more often (change rate) anger is more likely to be present in the signal. Meanwhile, if a high pitch is present but the pitch is not changing that much and abruptly, it is probably a happy ambient. But if a lower a pitch is estimated the atmosphere is more likely to be sad.

	Anger	Happiness	Sadness
Pitch Statistics	Higher in mean, median, range and variability	Higher in mean, median, range and variability	Lower in mean and range
Pitch Changes	Abrupt on stressed	Smooth, upward inflections	Downward inflections
Pitch Contour	Angular frequency curve, stressed syllables ascend, frequently and rhythmically, irregular up and down inflection, level average pitch, expect for jumps of about of about a fourth or a fifth on stressed syllables	Descending line, melody ascending, frequently and at irregular intervals	Downward inflections

Table 1: Relations between F_0 and emotions. Neutral speech is considered as a reference for the presented relative description [20] [23]

In previous research the F_0 patterns in different segmental and linguistic levels like phoneme, word or part of speech have been analysed. In these studies [21] [24] [25] it has been observed that the fundamental frequency significantly differs for angry, happy, sad and the neutral speech. Working with the features of F_0, the detection rate of this four-class problem was between 50.9 % and 55.7 %. [22]

The problem to which most of the errors can be referred in these approaches, is that there is uncertainty and confusion between happiness and anger, and sadness and neutral state. This was shown in a study from Serdar Yildirim and his colleagues where they investigated acoustic properties of speech associated with the four different emotions sadness, anger, happiness,

and neutral intentionally expressed in speech by an actress. As you can see in Figure 4 the fundamental frequencies on phoneme level are very similar for anger and happiness, and for neutral and sadness. Another perspective of these 4 emotional states and their relations to the fundamental frequency is shown in Figure 3. The “distance” between happy and angry, and sad and neutral is significant – but happy and angry are very close to each other, as well as sad and neutral.

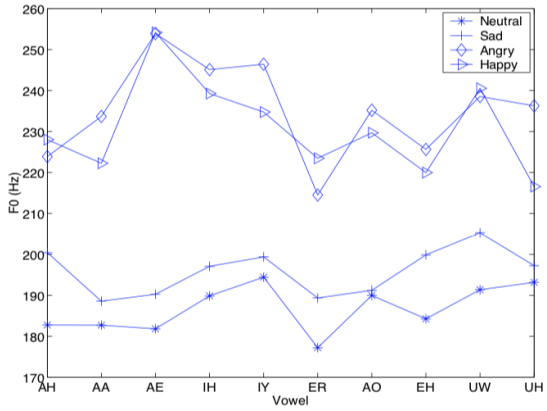


Figure 4: Mean Vowel fundamental frequencies. [22]

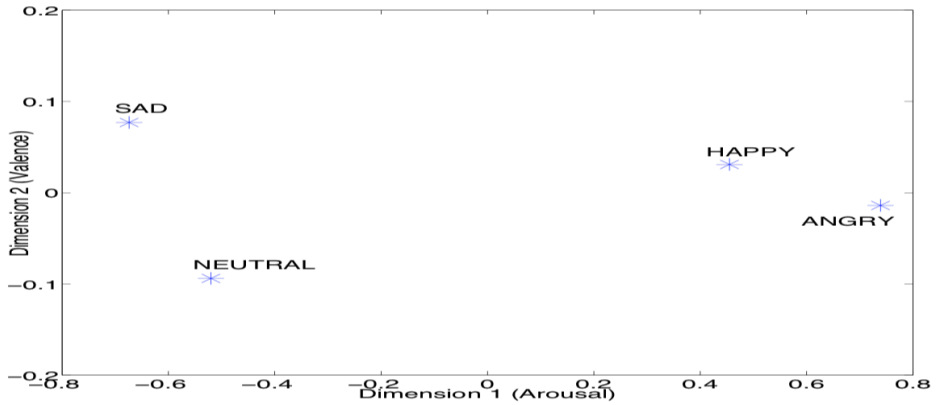


Figure 3: Multidimensional Scaling corresponding to F0 for emotion categories. [22]

This knowledge of the relationships between the four emotions (sad, happy, neutral and angry) and the fundamental frequency is used in this thesis to estimate the mood in an audio signal. So, the first step in this component is to put the audio sample into the right mood category. This is done by taking the pitch, the energy, the energy change rate and the emotion change rate into account. Pitch, energy, and the change rates are always computed from the last 20 sound samples to flatten out peaks and get a better understanding of the current mood in the signal.

Emotion	Pitch (average)	Energy (average)	Change rate
Sad	< 60	<= 30	-
Neutral	60 – 74	-	< 1.0
	< 65	> 20	-
Happy	> 74	-	< 2.5
Angry	> 74	-	> 2.5

Table 2: Mood Classification

The values presented in Table 2 are a result of experimenting around with the findings from above in mind and evaluating which value composition leads to a satisfying visualisation.

The second step deals with the mapping of this detected emotion to a representative colour. So that the audience can sense the present emotion in the visual as well as in the audio signal. There are many studies dealing with the topic of associating colours with mood. One of these studies, done in 1961 by Klaus Schaie [19, 20], suggests the following colour mapping:

Color	**Strong association**
Red	Protective, defending, powerful, strong, masterful (exiting, stimulating)*
Orange	Exciting, stimulating
Yellow	Exciting, stimulating, cheerful, jovial, joyful, pleasant
Green	
Blue	Pleasant, secure, comfortable, tender, soothing; (calm, peaceful, serene, exciting, stimulating
Purple	Dignified, stately, (despondent, dejected melancholy, unhappy)
* Parenthesis indicate moderate associations or mood-tones whose scalar order varied between the two groups of judges.	

Table 3: Descriptive Scheme for the Association between Colors and Mood-Tones [19]

The method he used for his evaluation is called constant stimulus, where the sensory threshold is determined by randomly presenting multiple stimuli known to be close to the threshold. It is also known as the constant stimulus method. He asked two groups of people, one with 24 and one with 21 people, to take part in the experiment.

For SoVia, four colours are used: Red for excitement and anger, Yellow for joy and happiness, Blue for sadness and calmness and Green for a neutral mood, because no particular mood is strongly associated to this colour according to Schaie [19, 20]

The implementation of the colour mapping also considers if the emotion is present for a while and varies around the chosen colour.

For this project, the Hue-saturation-lightness model (HSL) is used. The HSL model is perfect for this area of application because colour, saturation and lightness can be set separately by simply changing their values – that would be much more effort with the RGB model.

Every emotion moves around a specified Hue and saturation area as listed in the following table (Table 4)

Emotion	Min – Max Hue	Min – Max Saturation	
Sad	190 – 280	20 – 45	
Neutral	80 – 190	45 – 70	
Happy	30 – 60	80 – 100	
Angry	0 – 20	80 – 100	

Table 4: Emotion-Colour-Mapping

The saturation is, similar to the Hue, calculated by interpolating between the maximum value of the saturation for a specific mood and the actual saturation value. To make it smoother, the value is also reduced by the factor of 0.7 (this was also determinded by just experimenting and see what looks best in the visualization).

Lightness is also an important factor when associating colours with emotions. Suk and Irtel stated in their research that greyish colours are more connected with emotions like sad, calm and submissive. [9] Therefore the lightness of the color is depending on the pitch that is detected. Because of the relation between the pitch and the mood it is easy to compute the lightness by taking the value of the pitch into account. The higher the pitch the lighter the colour should appear.It is set accordingly to the current pitch but with regard to a separate “pitch reducer”. If the estimated mood is sad, the lightness should be more reduced then with other moods. A “pitch-reducer” is simply subtracted from the pitch to get the desired result in the visualisation.

Unfortunately, using the HSL colour scheme for Scalable Vector Graphics does not work as expected when it comes to saving the SVGs – therefor HSL colour is converted in the RGB colour scheme to can properly save the SVGs. If the saving of the SVGs is not necessary – HSL can be used without any problems.

Additionally, a “curve factor” is generated if the mood is estimated. It is a randomly generated number which specifies how the drawn line should look like.
If the mood is sad, the line should be straight. If the mood is happy or angry the line should be more lively and energetic. Therefore, the line should rather be drawn as a curve when these moods are detected. Neutral lies between curvy and straight. A detailed overview is provided in the following table (Table 5)

Emotion	Curve Factor
Sad	0
Neutral	random(-10, 10)
Happy	random(-70, 70)
Angry	random(-100, 100)

Table 5: Curve factor mapping

3.3.5. Line Generation

As already mention above in the technology stack section, Raphael.js is used to generate Scalable Vector Graphics in this project. With the SVG standard, shapes and images are described in a XML-based format. The graphics are not depicting as a grid of pixels, they are defined by a series of points and curves in space, which can also be called Vectors.
SVG can also be seen as a graphic language, that describes individual object and their properties like colour or gradient fills, thus they can be addressed and manipulated separately, similar to DOM objects in HTML code.
SVGs are well-supported by modern web browsers. Libraries like Raphael make it relatively straightforward to use SVG graphics across multiple browsers.
Because of the advances of SVGs and the compatibility across browsers, SVGs was chosen as underlying technology to generate visuals.

Before creating lines, a Raphael.js paper has to be created as the base for drawing SVGs. That is done at the very beginning with:

```
Raphael("paper", WIDTH, HEIGHT);
```

All elements generated in this application are drawn on top of this paper.

There are two modes for drawing with the detected moods. The first one is the manual drawing mode, where only the colour and stroke width are set through the emotion estimation. This is default mode when the application starts. The second mode available is the “Draw for me”-Mode, triggered by clicking the “Draw for me”-Button. This mode draws automatically considering the ambient sounds.

Manually

In the manual mode, the user can start drawing by simply clicking into the paper and moving the mouse around and stop drawing by clicking the mouse again. A line is drawn if a specific movement offset has been reached, so that not every little movement has to be processed. If the movement is big enough the emotion is calculated for that specific line. Finally, the line is generated with the Raphael.js path-method.

```
path = paper.path("M{0} {1}Q{0} {1} {2} {3}", start.x, start.y,
coordys.x, coordys.y)
    .attr({ "stroke-width":
            energyAvg * 0.3 < 1 ? 1 : energyAvg * 0.3,
        "stroke-linejoin": "round",
        "stroke-linecap": "round",
        stroke: getColorStr()
});
```

The line is drawn from the starting position to the current mouse position. The current mouse position becomes the start position for the next line. The energy average, calculated from the last 20 samples (as already mentioned in the mood mapping area above), is used for the stroke width attribute of the SVG path. The stroke is set to the colour mapped from the estimated mood from the acoustic environment. The edges of the paths are set to “round” with strike-linejoin and stroke-linecap, to get smooth transitions between the paths.

Automatically “Draw for me”

The „Draw for me“-mode is more sophisticated as the manuel mode. Lines are generated fully automatically by computing direction, length and curves of a line. For this purpose, a class called Line has been implemented, to hold the local attributes of a specific line: startpos (x, y), endpos(x,y), direction(x,y), and a history of all previous lines. There are two methods implemented for a line – update() and show(). The update method generates a new line and adds it to the history. The endpoint of the previous line becomes the startpoint of the current line and a new endpoint has to be calculated. This is done by generating a new direction

vector, multiplied by the length which is computed by summarizing the energy change rate the emotion change rate and multiply them by a slowdown factor. The slow down factor is reduced if there is a harmony detected (the detected pitch stays the same for the 20 samples that are considered). Additionally, if a harmony is detected the direction does not change and is taken from the previous line. So, the end position of the new line is the start vector plus the calculated vector. If the new endpoint lays outside the window, the vector is rotated until it is inside the window again or a specific count is reached. (36 as a full rotation – because the angle is adjusted with 10 every time the loop is executed). All the currently available information of the line is saved in an extra object and added to the line history – Start of the path, end of the path, the colour string, energy and the curve factor are saved in this object to use it for creating the path in the show()-method.

A line this context is actually a set of paths, to have a visual example have a look at Figure 5. It is not necessary to save all the generated line information as a history. But it could be interesting for further development of the application, so this was not refactored.

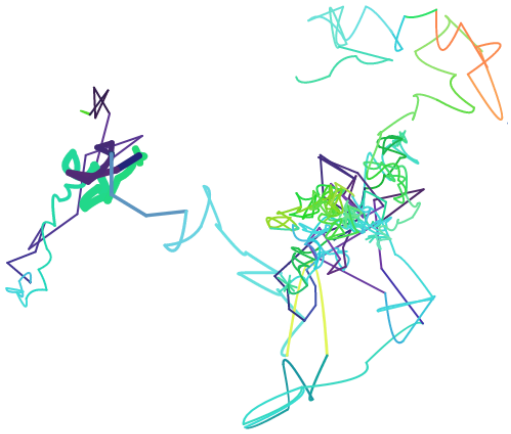


Figure 5: Example of a Line

After updating the Line object, that latest path added to the line is generated in the show()-method of the Line. For drawing the line, the cubic Bezier-Curve is used. This can simply be achieved by using the Curve Command Q and an extra control point, calculated with the curve factor and the endpoint:

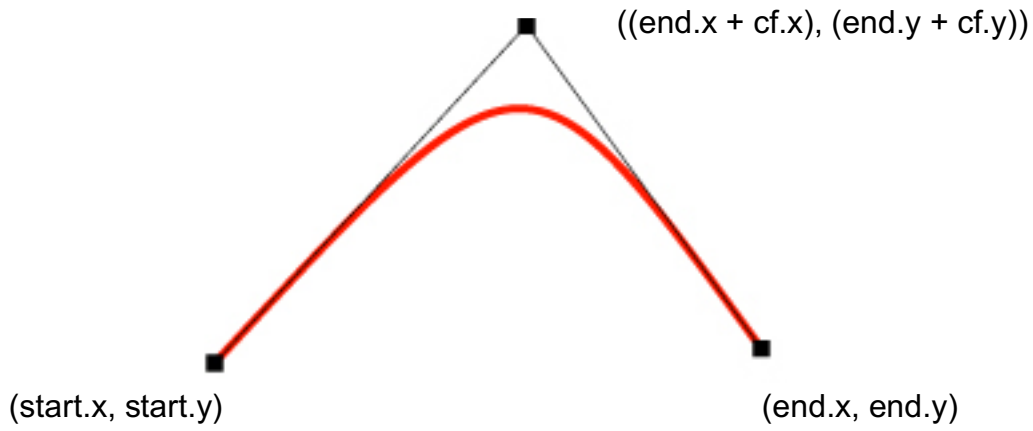


Figure 6: Cubic Bezier Curve

```
M end.x, end.y Q (end.x + cf.x) (end.y + cf.y) start.x start.y)
```

The attributes of a line are set exactly like it is done in the manual drawing mode. Further, event listeners are added to every path, so that a switch to the manual mode is possible without problems. Because of the fact that SVGs are handled similar to DOM elements it is not possible to interact with the paper that lies under the SVG object. Therefore it is necessary to add a mouse listener to draw over an existing SVG path.

If the “Draw for me”-Button is clicked five Line objects are created. They start with a random 2D vector and are updated separately every time the p5.js draw method is called. In this project the framerate is set to 200 frames per second.

3.3.6. Additional Features

As an addition, a saving functionality was added to this project. This makes it possible to save visualized moods as a SVG file directly to the computer. This was implemented with the help of the Raphael.js extension Raphael.export.js. With this extension, it is easy to convert the paper into a SVG:

```
svg = paper.toSVG();
```

Then a Blob object with the SVG file is created and download via an object URL:

```
a = document.createElement('a');
a.download = 'mood.svg';
a.type = 'image/svg';
blob = new Blob([svg], {"type": "image/svg"});
a.href = (window.URL || webkitURL).createObjectURL(blob);
a.click();
```

Another Gimmick, as already mentioned in the Section 3.2, is the circle-drawing-mode. This feature is only available in the manual drawing mode and simply switches from drawing paths to drawing circles with the radius of the energy and the current mood colour. You can also switch into a dark mode when drawing with circles, where the mapped mood colour is used as the stroke colour and the fill colour is set to black.

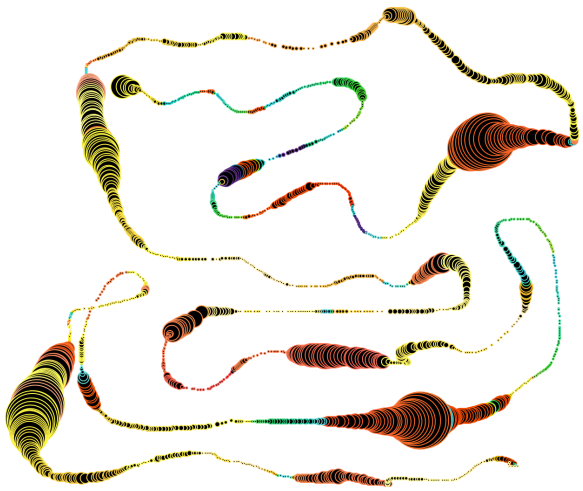


Figure 7: Dark Circle Mode

4. Generated Artefacts

4.1. Analysing Artefacts

For a better understanding of how the generated artefacts look like or have to be interpreted, let's have a look at some examples. The Audio samples used for the following images can be found in the Git Repository.

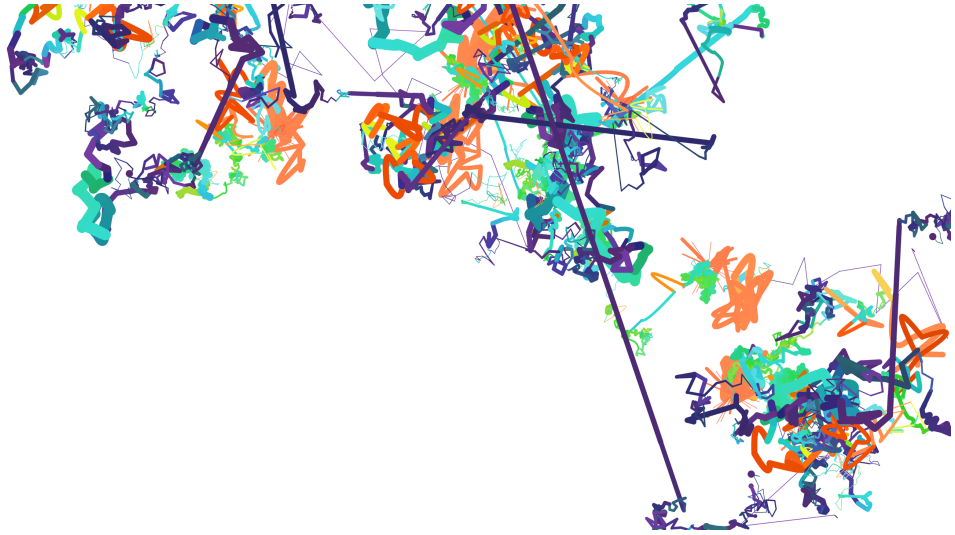


Figure 8: Charles Mingus - Goodbye pork pie hat (Sample_2_Charles_Mingus)

This song is a harmonic one, with more sad, calm and slow parts. The longer lines are a result of an abrupt change in pitch or energy level. Where the stroke of the lines are bolder, the part was louder and more energetic. In this particular song, the red parts of the drawing are a result of the higher pitch of the trumpets.

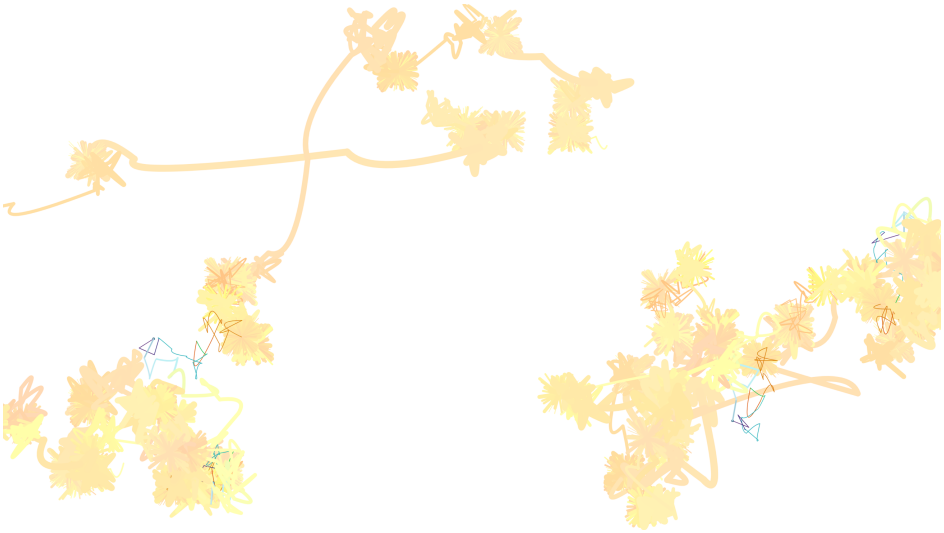


Figure 9: Birds (Sample_3_Birds)

In Figure 7, the underlying sound sample is birdsong. The mood that is detected is always bright and happy. The star-shaped parts of the drawing, resulting from consistent pitches.

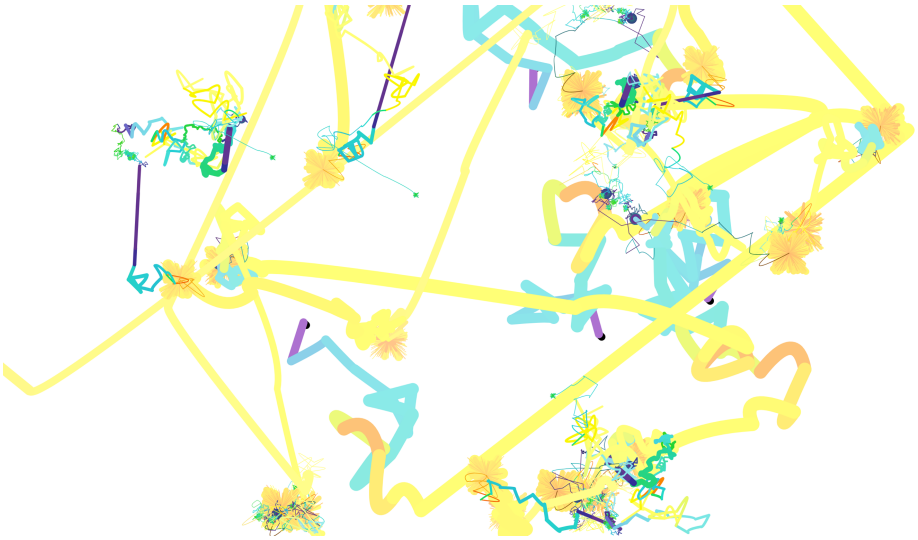


Figure 10: Street Noises (Sample_4_Streets)

In Figure 8, Street Noises are visualized. Unfortunately, the colour yellow is a little bit misleading, but if we think about passing cars the sound is not very aggressive or in our context angry (depending on the type of car passing of course). The other problem with this background noise is the that the pitch detection algorithm has problems with brown and pink noise present in this scenery. The mood which is detected is not always the one we would expect. Nevertheless, you can see that there are some very quiet parts combined with very turbulent parts and jumps from calm to more excited. In this picture, the red colour is not present at all, because the pitch is not changing that much and the street sound is in general not very exciting and thrilling.

4.2. Black-Box Evaluation

4.2.1. Procedure

To evaluate the outcome of this project, a short survey was carried out. Six pictures were generated from six different background sounds and pieces of music and presented to people which has no detailed information about the algorithm that generated the images, the only thing mentioned in the introduction was the purpose of this project and the parameters of the audio signal taken into account. For the survey, a free online survey tool was used (easy-feedback.de).

It started with a short introduction:

„As part of my bachelor thesis, I developed a program to translate audio signals into images. The idea was to visualize noises, conversations, music, etc. and the resulting mood. The pitch, volume and other characteristics of the audio samples were determined and mapped to the color, thickness and smoothness of a line.
I implemented this idea as a prototype and am now interested in your opinion."

The survey was originally held in German, so the introduction text is translated.
In the following, the generated pictures and the sound samples are presented. While scrolling down, the audio is provided, so the participant could listen to the sample directly. Scrolling further, the pictures were displayed like shown in Figure 9. Following a multiple-choice question "Which image was generated from this audio?" with the six numbers of the pictures as possible answers. So, the participant listens to the audio sample and assigns it to the visual he or she thought it is the visual generated from this given audio sample. This was done in the same way for all six sound samples and visuals – presented one after another.

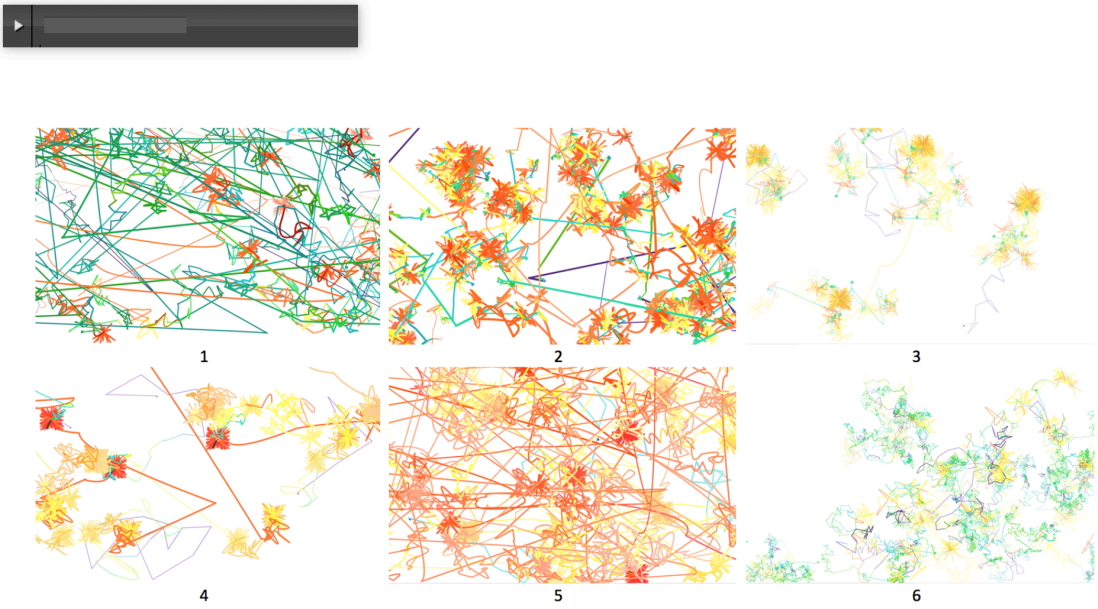


Figure 11: One part of the survey. Including the six generated pictures available

As you can see the participant has very little information about the generation of the visuals. Furthermore, some people listen to one sample after another, and so the decision how to assign the pictures was more difficult.

The survey was provided as a link and sent to friends and relatives and was shared on social media platforms, so that a broad group of people was able to attend the survey. At the end 42 people took part in the survey and provided interesting feedback on the visual artefacts that were generated from sound samples.

4.2.2. Results

The right mapping of audio samples and generated pictures is shown in Table 6.
The used audio samples and the pictures can be also found in the GitHub Repository in the folder Samples.

Audio	Picture
1: Classic Music – Claude Debussy	6
2: Construction work noises	4
3: A TED Talk – Joseph Gordon Levitt	1
4: Pop Song – Fleetwood Mac – Dreams	2
5: Birdsong	3
6: Techno – Charlotte de Witte	5

Table 6: Correct mapping of sound samples and pictures

Detailed results – used for the following matrix (Table 7) – can be found in the appendix in section “Survey Results”.

1.1.1.1. Result Matrix

	A1/P6 Classic	A2/P4 Noise	A3/P1 Talk	A4/P2 Pop	A5/P3 Birdsong	A6/P5 Techno	TP/FP
A1/P6 Classic	**16** 38.10 %	**2** 4.78 %	**2** 4.78 %	**14** 33.33 %	**4** 9.52	**4** 9.52 %	**38.10 %** **61.90 %**
A2/P4 Noise	**1** 2.38 %	**5** 11.9 %	**7** 16.67 %	**4** 9.52 %	**1** 2.38 %	**24** 57.14 %	**11.90 %** **88.10 %**
A3/P1 Talk	**2** 4.76 %	**20** 47.62 %	**1** 2.38 %	**1** 2.38 %	**18** 42.86 %	**0** 0.00 %	**2.38 %** **97.62 %**
A4/P2 Pop	**13** 30.95 %	**3** 7.14 %	**8** 19.05 %	**14** 33.33 %	**3** 7.14 %	**1** 2.38 %	**33.33 %** **66.67 %**
A5/P3 Birdsong	**11** 26.19 %	**8** 19.05 %	**4** 9.52 %	**3** 7.14 %	**14** 33.33 %	**2** 4.76 %	**33.33 %** **66.67 %**
A6/P5 Techno	**2** 4.76 %	**1** 2.38 %	**22** 52.38 %	**5** 11.9 %	**0** 0.00 %	**12** 28.57 %	**28.57 %** **71,43 %**

Table 7: Result Matrix of the survey results

The outcome of the survey was quite surprising. Although the participants did only know so little about the generation process of the pictures, what meanings the colours have, or what the length of path should express, half of the sound samples were mapped correctly to the generated pictures by the majority of the attendees. If you compare the True Positives with the False Positives, you can see that 4 Sound Samples around a third chosen the right picture to the sound. Of course, there is still room for improvement, but for the first attempt, this result is surprisingly good. There are some interesting findings, having a closer look at the feedback.

Audio Sample 1 – Classic Music

The first audio sample was classic music. 38.10 % assigned this sample to the right picture, 33.33 % of the participants assigned it to the picture generated from the pop song from audio

sample 4. So, at least, music can be identified in the pictures by around 70 % of the participants.

Audio Sample 2 – Construction Work Noise

Unfortunately, the construction work noise sound sample was mistakenly confused with the visual created from the Techno sample from a majority of 57.14 %. This result is not surprising – first of all the participants did not know which sound samples were following, so it totally makes sense to assign the picture which seems to be the most "annoying" to the sound of construction work. Second, in the sound sample of the construction work, it seems that there is a lot going on, but actually the algorithm is not smart enough yet to detect noises as we would expect it. The pitch detection does not work properly for noisy sound as mentioned in the sections above in more detail.

Audio Sample 3 – A TED Talk

The audio sample that should be referred to picture number 1, was correctly assigned by 1 person. This could probably be the same problem as mention already for the contruction work sample. The long green zick zack lines could be a bit misleading, almost a half of the participants thought that this could be assigned to the construction work picture, the other half thought that this could be the birdsong picture. This picture was probably the most difficult to assign although or because it differed the most.

Audio Sample 4 – Pop Song

Audio sample 4 was easier to assign, similar to audio sample 1 a third of the participants assigned the audio sample to the right image – image number 2. Almost another third assigned the pop song to the classic music image. So, as already mentioned, the music samples are detected from the majority.

Audio Sample 5 – Birdsong

For the birdsong audio sample, a third assigned the right image, another third was confusing the birdsong with the classic music image, which was also predictable. Both images radiate calm and without any knowledge about the meaning of the star-shaped components in the pictures this is hard to distinguish between.

Audio Sample 6 – Techno

The last audio sample that was presented in the survey was Techno music. Surprisingly, this was confused with the generated visual from the TED talk audio sample by a half of the attendees. But 28.57 % get it right and assigned the audio sample to the right image.

Taking into account, that the attendees, did not have detailed knowledge of the system and how it exactly operates and did not have any experiences with this system, the numbers of right assignments were surprisingly good. Some of the participants gave individual feedback, they said it was quite difficult to assign the right image to a sound sample with the knowledgebase they had about the system. After talking to some participants in more detail they also mentioned that they associate blue with a more aggressive and excited situation. Another interesting aspect that was pointed out by the participants was, that they sometimes do not associate some ambient situations with the same emotions, hence that they are expecting another way of visualisation of such ambient sounds.
Further, this project is not necessarily made for generating artefacts that are interpreted after they were created. It should be more a live visualization of a current noise scene and convey to the viewer what's going on at the moment of watching the application draw.

5. Conclusion

In first place the implementation of SoVia should be a proof of concept and demonstrates the capabilities of this approach of visualizing ambient sounds. With further improvements suggested in Section 5.1, SoVia could lead to an interesting sound visualisation tool for different fields of applications. For example, the Deaf and Hard Hearing community could profit from a tool like this to visualize their acoustic environment and are then able to participate in a better way on their environment.

While working on this approach and especially on the evaluation part, it turned out that people not always see the same emotions in specific colour. Talking to participants of the evaluation has shown that some people see blue as the more aggressive colour and red as the calm one just to name an example. It is also interesting that for some people ambient sounds are calming and for others the same ambient sound is annoying and stressing and therefore they expect another visualization than others. Another important aspect was, that there are cultural differences in perceiving colour. For example, the colour red in the European area is associated with courage, anger, aggression, passion, danger and also sexuality whereas in the Chinese culture the colour red stands for happiness, beauty, vitality, good luck, success and good fortune. These are all aspects that have to be considered in future work if this approach should be accessible and meaningful for a broad audience.

5.1. Future Work

There are restrictions, already mentioned throughout this thesis, like proper visualization of noisy sound samples or drawing images from conversations.

Localisation of the audio signal

Instead of randomly drawing paths, the position and direction of a path could be calculated from the direction of origin of the sound signal. But, to achieve a position-dependent visualization, special technical equipment is necessary – at least a stereo microphone is needed to distinguish between right and left. With iOS 14 for example, it is possible to record audio in stereo manner, to create the illusion of multidirectional sound, adding greater depth and dimension to the audio. [26]

Visualize audio channels separately

Especially for generating visuals from music audio samples it could be interesting to visualize every dimension of a song separately. For example, the visualization of the drums is done in a different way, as it is done for the guitars or the vocals.

Take more sound characteristics

In this implementation, the focus was on working with the pitch, the energy and how the signal changes over time regarding the pitch and energy. There are many other characteristics that could be also interesting for this approach. Interesting additional characteristics for example could be the Spectral Kurtosis as mentioned in the Meyda [27]. For detecting pointedness in a spectrum – this could lead to a better visualisation of noisy sounds. Additionally, the sharpness of a signal could be taken into account. To improve music and conversation visualisation, the Mel Frequency Cepstral Coefficients could be added to get information about the timbral charactaristics of an audio. There are many more characteristics that could be interesting for extending the experience of this implementation, the examples above are just to name a few.

More detailed evaluations

To get more feedback on how audiences perceive the visualisation of sound more evaluations are necessary. So that parameters of the drawing mechanism could be adjusted and a broader audience still gets the same impression when looking at a generated image.

At the end, SoVia is an interesting combination of psychological, technological and also artistic aspects. Although the result is not completely satisfying yet, it is a start for an approach that might develop into a more sophisticated tool for visualizing ambient sounds.

6. Literature

[0] Collins, Kc, Taillon, P., 2012, Visualized sound effect icons for improved multimedia accessibility: A pilot study. Entertainment Computing.

[1] Edwards, A.D.N., 199, Progress in Sign Language Recognition. In Proc. of Gesture and Sign-Language in HCI '97

[2] Auditory Visual Articulation Therapy Software, Sonido Inc., www.sonidoinc.com.

[3] Elssmann, S.F., Maki, J.E., 1987 Speech spectrographic display: use of visual feedback by hearing-impaired adults during independent articulation practice. American Annals of the Deaf

[4] Ellis, D.P.W., 1997, The Weft: A representation for periodic sounds. In Proc. of ICASSP '97, 1307-1310.

[5] Jewanski J., 2009, Color-Tone Analogies, See This Sound Compendium (2009)

[6] McDonnell M., 2010, Visual Music - A Composition Of The Things Themselves, Sounding Out, Bournemouth University, UK

[7] MDN Web Docs; Web Audio API; https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API, 2021

[8] F. Adams, C. Osgood. 1973 A Cross-Cultural Study of the Affective Meanings of Color. Journal of Cross-Cultural Psychology. (2):135-156.

[9] H. Suk, H. Irtel, 2010, Emotional Response to Color Across Media. Color Research and Application, 35, 64-77.

[10] Wenqi Li, Lee J., Shiffman D.. ML5.js, 2018, GitHub, https://ml5js.org/

[11] Aiko Deja J., Dela Torre A., Lee H. J., Ciriaco J. F., Eroles C. M., 2020, ViTune: A Visualizer Tool to Allow the Deaf and Hard of Hearing to See Music With Their Eyes. In Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems (CHI EA '20). Association for Computing Machinery, New York, NY, USA

[12] Github, Atom, OpenSource, 2021

[13] Baranovskiy D., Raphael.js, 2008, MIT License

[14] Lee McCarthy L., P5.js, 2021, LGPL-2.1 License,

[15] Kim J. W., Justin Salamon J., Li P., Bello J. P., 2018, CREPE: A Convolutional Representation for Pitch Estimation, IEEE International Conference on Acoustics, Speech and Signal Processing

[16] Mauch M., Dixon S., 2014, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

[17] Camacho A., Harris J. G., 2008, “A sawtooth waveform inspired pitch estimator for speech and music,” The Journal of the Acoustical Society of America, vol. 124, no. 3

[18] Manav, Banu, 2007, Color-emotion associations and color preferences: A case study for residences. Color Research & Application. 32. 144 - 150. 10.1002/col.20294.

[19] Schaie, K., 1961, Scaling the Association between Colors and Mood-Tones. The American Journal of Psychology

[20] Schaie, K, 1961, A Q-sort Study of Color-Mood Association. Journal of projective techniques.

[21] Cowie, R, Douglas-Cowie, E., Tsapatsoulis, N., Votsis, G., Kollias, S., Fellenz, W., Taylor, J. G., 2001, Emotion recognition in human-computer interaction. IEEE Signal Processing Magazine, 18/1,32–80

[22] Yildirim, S., Bulut, M., Lee, C. M., Kazemzadeh, A., Busso, C., Deng, Z., Lee, S., Narayanan, S. S. 2004. An acoustic study of emotions expressed in speech. In 8th International Conference on Spoken Language Processing (ICSLP 04), pages 2193–2196, Jeju Island, Korea

[23] Busso, C., Bulut, M., Lee, S., Narayanan, S., 2008. Fundamental frequency analysis for speech emotion processing.

[24] Bulut, M., Lee, S., Narayanan, S. S, 2007, Analysis of emotional speech prosody in terms of part of speech tags. In Interspeech 2007 - Eurospeech, pages 626–629, Antwerp, Belgium

[25] Bulut, M., Lee, S., Narayanan, S. S., 2008, Analysis of emotionally salient aspects of fundamental frequency for emotion detection. IEEE Transactions on Audio, Speech and Language Processing, In press 2008.

[26] Apple Developer, Capturing Stereo Audio from Buildt-In Microphones. https://developer.apple.com/documentation/avfaudio/avaudiosession/capturing_stereo_audio_from_built-in_microphones, 05/2021

[27] Rawlinson h., Segal N., Fiala j., 2015, Meyda: an audio feature extraction library for the Web Audio API, 1st Web Audio Conference ’15, Paris, France.

[28] B. P. Lathi, 1998, Modern Digital and Analog Communication Systems 3e Osece. Oxford University Press, 3rd ed.

[29] A. Deveria, L. Schoors, https://caniuse.com, 2021

7. Appendix

7.1. Survey Results

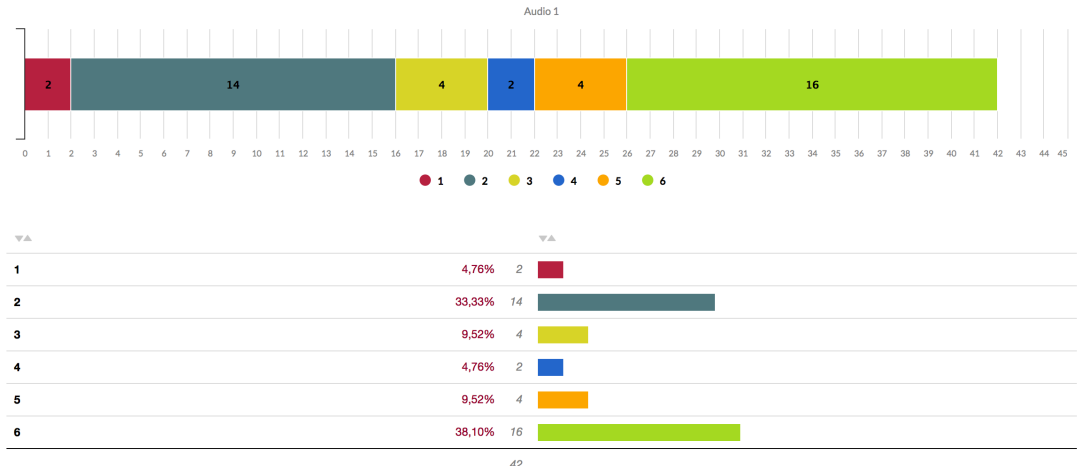


Figure 12: Audio 1 - Classic Music

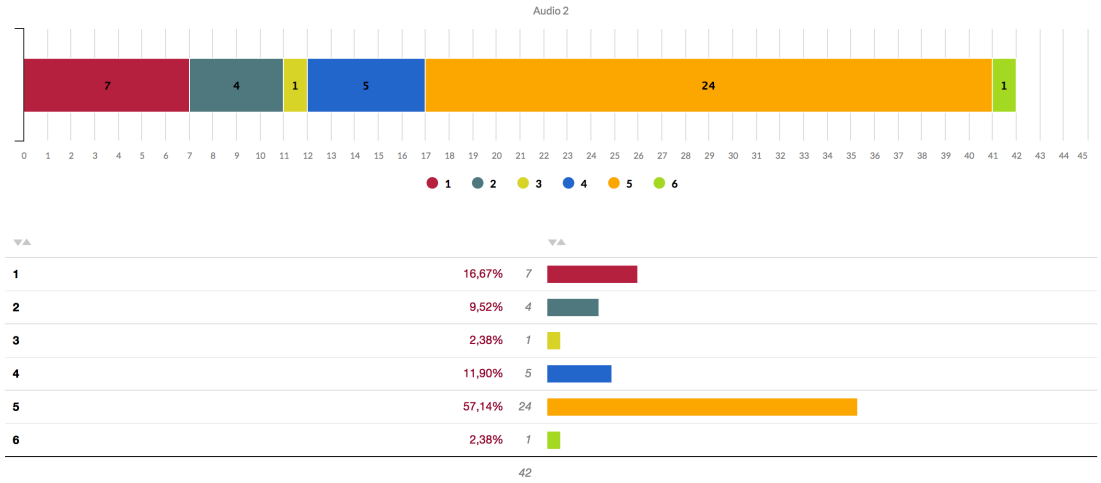


Figure 14: Audio - Construction Work Noise

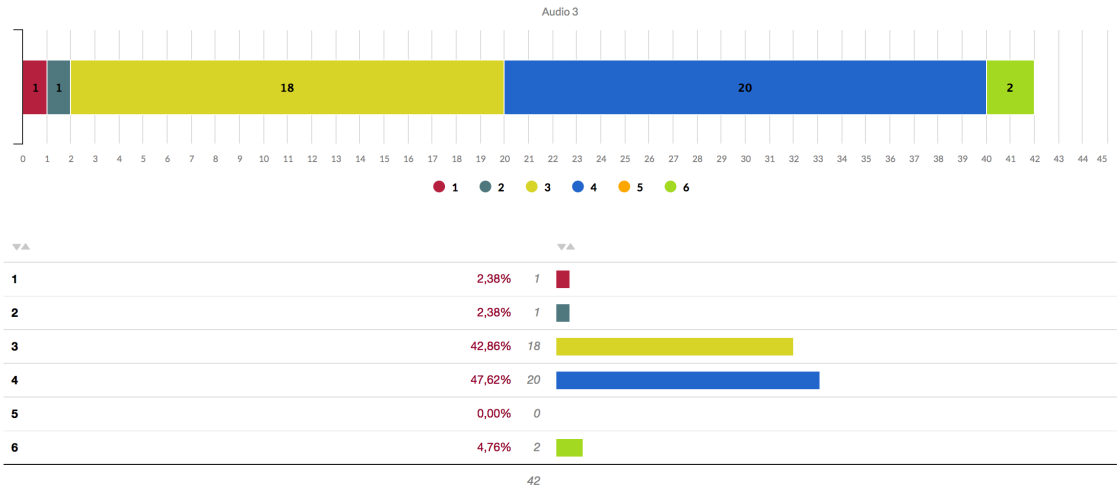
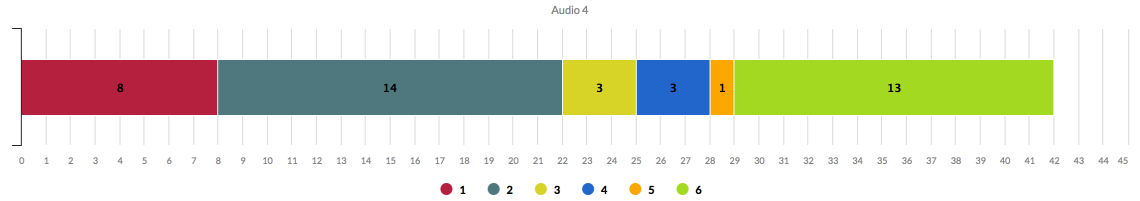
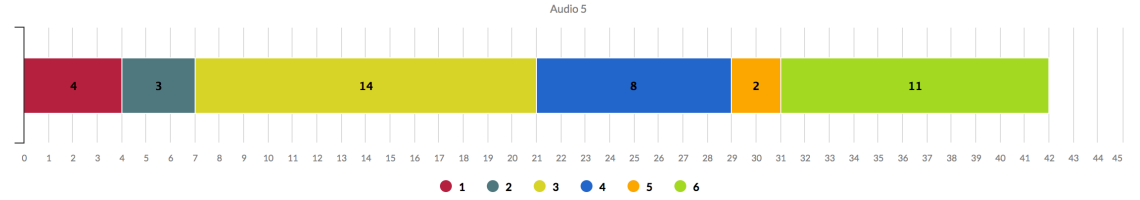


Figure 13: Audio 3 - TED Talk



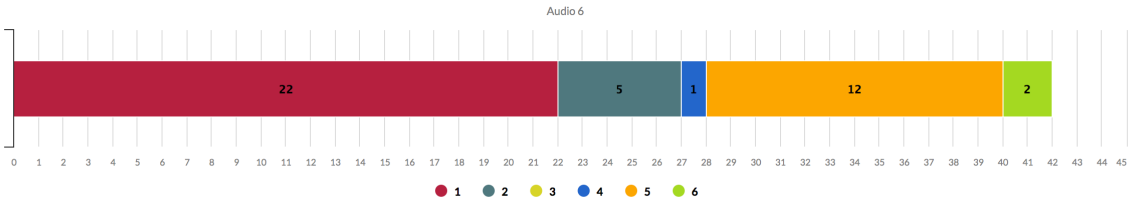
1	19,05%	8
2	33,33%	14
3	7,14%	3
4	7,14%	3
5	2,38%	1
6	30,95%	13
		42

Figure 17: Audio 4 - Pop Song



1	9,52%	4
2	7,14%	3
3	33,33%	14
4	19,05%	8
5	4,76%	2
6	26,19%	11
		42

Figure 16: Audio 5 - Birdsong



1	52,38%	22
2	11,90%	5
3	0,00%	0
4	2,38%	1
5	28,57%	12
6	4,76%	2
		42

Figure 15: Audio 6 - Techno

7.2. Code base

7.2.1. drawing.js

```
let audioStream;
let stream;
let analyzer;
// stereo-analyser-node.jslet audioArray;

let WIDTH = 1000;
let HEIGHT = 700;
const PAPER_OFFSET = 0;

// Compute movement required for new line
var X_MOVE = WIDTH * 0.001;
var Y_MOVE = HEIGHT * 0.001;

let currPitch = 0;

const currentText = '';
let textCoordinates;
let paper;
let area;

let mouseDown = false;
let start, coordys;

let topPos = 0;
let leftPos = 0;

var act_color = {
  h: 100,
  s: 100,
  l: 0
}

var prevColor = {
  h: 155,
  s: 100,
  l: 50
}

let colorSad = {
  hMin: 190,
  hMax: 280,
  sMin: 20,
  sMax: 45,
  l: 50
}

let colorNeutral = {
  hMin: 80,
  hMax: 190,
  sMin: 45,
  sMax: 70,
  l: 50,
}
```

```
let colorHappy = {
  hMin: 30,
  hMax: 60,
  sMin: 80,
  sMax: 100,
  l: 50
}

let colorExcited = {
  hMin: 0,
  hMax: 20,
  sMin: 80,
  sMax: 100,
  l: 50
}

// 1 sad - 2 neutral - 3 happy
let currMood = 0;
let prevMood = 0;
let moodChanged = 1;

let moodfactorOperation = true;
let moodfactor = 1;
let moodCount = 0;

var audioArray;

var emotionQ = [];
var energyQ = [];
var energyAvg = 10;
var energy = 1;

var harmony = false;
var slowDownFactor = 15;

var emotionChangeAvg = 10;
var energyChangeAvg = 10;

var curvefactor = {
  x: 10,
  y: 10
}

let withEmotions = 0;
const EMOTION_BUTTON_TXT_ADD = "add mood"
const EMOTION_BUTTON_TXT_STOP = "stop mood"
const EMOTION_BUTTON_TXT_LOADING = "mood ..."
const LINE_TXT = "do it in lines"
const CIRCLE_TXT = "do it in circles"
const AUTO_DRAW = "draw for me"
const AUTO_DRAW_STOP = "stop drawing"

var paperText;


var inCircles = false;
var darkMode = false;
```

```
// sound upload
let upload;
let sound;

// auto drawing
let auto = false;
let initAutoPos = {
  x: 0,
  y: 0
};

let request;

// for different particles
let vibrations = [];

// Filter for SVGs
let filter;



async function setup() {



  WIDTH = window.screen.width;
  HEIGHT = window.screen.height;

  paper = createPaper(WIDTH, HEIGHT);
  frameRate(200)
  textCoordinates = [WIDTH / 2, 30];

  newSheet();


  // Filter for lines
  filter = paper.createFilter();
  filter.chainEffect("feTurbulence", {
    type: "fractalNoise",
    baseFrequency: "0.1",
    numOctaves: "10"
  });
  filter.chainEffect("feDisplacementMap", {
    in: "SourceGraphic",
    scale: "10"
  });
  filter.merge(filter.getLastResult(), "SourceGraphic");


  var addEmotionButton = document.getElementById("addEmotion");
  var saveEmotionButton = document.getElementById("savePaper");

  audioContext = new AudioContext();
  stream = await navigator.mediaDevices.getUserMedia({
    audio: true,
    video: false
  });
```

```
// (START \ STOP) EMOTION
addEmotionButton.onclick = function(event) {
  if (!withEmotions) {
    addEmotionButton.textContent = EMOTION_BUTTON_TXT_LOADING
    withEmotions = 1
    startPitch(stream, audioContext);
    startEnergy(stream, audioContext);
  } else {
    addEmotionButton.textContent = EMOTION_BUTTON_TXT_ADD
    withEmotions = 0
  }
};

var clearButton = document.getElementById("clear");

// CLEAR PAPER
clearButton.onclick = function(event) {
  clearPaper();
};

// SAVING SVG
saveEmotionButton.onclick = function(event) {
  svg = paper.toSVG();

  a = document.createElement('a');
  a.download = 'mood.svg';
  a.type = 'image/svg';
  blob = new Blob([svg], {
    "type": "image/svg"
  });
  a.href = (window.URL || webkitURL).createObjectURL(blob);
  a.click();
};

var circleButton = document.getElementById("doItInCircles");
var darkModeButton = document.getElementById("darkMode");
circleButton.onclick = function(event) {
  if (!inCircles) {
    inCircles = true;
    darkModeButton.style.visibility = 'visible';
    circleButton.textContent = LINE_TXT;
  } else {
    inCircles = false;
    darkModeButton.style.visibility = 'hidden';
    circleButton.textContent = CIRCLE_TXT;
  }
}

darkModeButton.onclick = function(event) {
  darkMode = !darkMode;
}

var draw4me = document.getElementById("draw4me");

draw4me.onclick = function(event) {
  auto = !auto;

  if (auto) {
```

```
      for (let i = 0; i < 5; i++) {
        vibrations.push(new Line({
          x: random(window.innerWidth),
          y: random(window.innerHeight)
        }, p5.Vector.random2D()));
      }

      draw4me.textContent = AUTO_DRAW_STOP;
      paperText.remove()
      initAutoPos.x = random(topPos, WIDTH);
      initAutoPos.y = random(leftPos, HEIGHT);
    } else {
      draw4me.textContent = AUTO_DRAW;
      setPaperText();
    }

  };

}

function setPaperText() {
  paperText = paper.text(WIDTH / 2, HEIGHT / 2, "").attr({
    fill: '#000000',
    "text-align": "center",
    "font-family": "\"Lucida Console\", \"Courier New\", monospace",
  });
}

function newSheet() {
  mouseDown = false;
  area = paper.rect(PAPER_OFFSET, PAPER_OFFSET, WIDTH, HEIGHT);

  area.attr({
    fill: "white"
  });

  setPaperText();

  // EVENTS
  area.click(function(event) {
    if (!mouseDown) {
      paperText.remove();
      var paperElem = document.getElementById("paper");
      topPos = paperElem.offsetTop;
      leftPos = paperElem.offsetLeft;

      mouseDown = true;
      start = {
        x: event.clientX - leftPos,
        y: event.clientY - topPos
      };
    } else {
      start = 0;
      coordys = 0;
      mouseDown = false;
    }
  });;
```

```
  area.mousemove(function(event) {
    if (mouseDown) {
      coordys = {
        x: event.clientX - leftPos,
        y: event.clientY - topPos
      };
    }

  });
}

// Pitch is calculated via ML5.js
function startPitch(stream, audioContext) {
  pitch = ml5.pitchDetection('./model/', audioContext, stream,
modelLoaded);
}

// Energy is calculated via Meyda.js
function startEnergy(stream, audioContext) {
  // Meyda.js
  if (typeof Meyda === "undefined") {
    console.log("Meyda could not be found! Have you included it?");
  } else {

    var source = audioContext.createMediaStreamSource(stream);

    const analyzer_energy = Meyda.createMeydaAnalyzer({
      "audioContext": audioContext,
      "source": source,
      "bufferSize": 2048,
      "featureExtractors": ["energy"],
      "callback": features => {
        energy = features.energy;
      }
    });
    analyzer_energy.start();
  }
}

function modelLoaded() {
  document.getElementById("addEmotion").textContent =
EMOTION_BUTTON_TXT_STOP;
  getPitch();
}

// Get pitch and convert it to Midi values - so it is easier to work
with
function getPitch() {
  pitch.getPitch(function(err, frequency) {
    if (frequency) {
      currPitch = freqToMidi(frequency);
      getEmotion();
    }
    if (withEmotions) {
      getPitch();
    }
  })
```

```
}

function draw() {

  // Automatic drawing
  if (auto) {
    if (withEmotions) {
      getEmotion();
    }

    for (let i = 0; i < vibrations.length; i++) {
      // add new path to the Line object
      vibrations[i].update();
      // draw the latest added path
      vibrations[i].show();
    }

  } else {

    // Manual drawing
    if (mouseDown && start && coordys) {
      var xMovement = Math.abs(start.x - coordys.x);
      var yMovement = Math.abs(start.y - coordys.y);

      if (xMovement > X_MOVE || yMovement > Y_MOVE) {
        if (withEmotions) {
          getEmotion();
        }

        var color_rgb = getColorStr()
        var path;
        if (inCircles) {
          path = paper.circle(start.x, start.y, energyAvg * 0.5 < 1 ? 1
: energyAvg * 0.5).attr({
            fill: darkMode ? "rgb(0,0,0)" : color_rgb,
            "stroke-linejoin": "round",
            "stroke-linecap": "round",
            stroke: color_rgb
          });

          // path.filter(filter);
        } else {
          path = paper.path("M{0} {1}Q{0} {1} {2} {3}", start.x,
start.y, coordys.x, coordys.y).attr({
            "stroke-width": energyAvg * 0.5 < 1 ? 1 : energyAvg * 0.5,
            "stroke-linejoin": "round",
            "stroke-linecap": "round",
            stroke: getColorStr()
          });

          // path.filter(filter);
        }

        path.mousemove(function(event) {
          if (mouseDown) {
            coordys = {
              x: event.clientX - leftPos,
              y: event.clientY - topPos
```

```
            };
            //drawLine();
          }
        });
        path.click(function(event) {
          if (!mouseDown) {
            mouseDown = true;
          } else {
            mouseDown = false;
          }
        });
        start = coordys;
      }
    }
  }
}

function getEmotion() {

  // PITCH
  emotionQ.push(currPitch);
  var avg = averageQ(emotionQ);
  const allTheSame = (currentValue) => currentValue == currPitch;

  harmony = emotionQ.every(allTheSame);

  // ENERGY by Meyda
  energyQ.push(energy);
  energyAvg = averageQ(energyQ);

  emotionChangeAvg = calculateAverageChange(emotionQ);
  energyChangeAvg = calculateAverageChange(energyQ);

  if (harmony) {
    moodChanged = 0;
  } else {

    if ((avg < 60 && energy <= 30)) {
      if (currMood != 1) {
        prevMood = currMood;
        currMood = 1;
        moodChanged = 1;
        moodCount = 1;
      } else {
        moodChanged = 0;
      }
    } else if ((avg >= 60 && avg < 74 && energyChangeAvg < 1.0) || (avg
< 65 && energy > 20)) {
      if (currMood != 2) {
        prevMood = currMood;
        currMood = 2,
          moodChanged = 1;
        moodCount = 1;
      } else {
        moodChanged = 0;
      }
    } else if (avg > 74 && energyChangeAvg < 2.5) {
      if (currMood != 3) {
```

```
      prevMood = currMood;
      currMood = 3;
      moodChanged = 1;
      moodCount = 1;
    } else {
      moodChanged = 0;
    }
  } else { // if (avg > 74 && emotionChangeAvg >= 0.5){
    if (currMood != 4) {
      prevMood = currMood;
      currMood = 4;
      moodChanged = 1;
      moodCount = 1;
    } else {
      moodChanged = 0;
    }
  }
}

if (moodChanged) {
  moodCount = 0;
}

moodfactor = moodCount / 10;

if (currPitch > 0) {
  // sad
  if (currMood == 1) {

    curvefactor = {
      x: 0,
      y: 0
    }

    calculateEmotionColor(colorSad, 10, 20);
  }
  // neutral
  else if (currMood == 2) {

    curvefactor = {
      x: random(-10, 10),
      y: random(-10, 10)
    }

    calculateEmotionColor(colorNeutral, 10, 10);
  }
  // happy
  else if (currMood == 3) {

    curvefactor = {
      x: random(-70, 70),
      y: random(-70, 70)
    }

    calculateEmotionColor(colorHappy, 1, 10);
  }

  // Excited
```

```
    else if (currMood == 4) {

      curvefactor = {
        x: random(-100, 100),
        y: random(-100, 100)
      }

      calculateEmotionColor(colorExcited, 1, 10);

    }

    prevColor = act_color;
    moodCount++;
  }

}


function calculateEmotionColor(emotionColor, moodMultiplyer,
pitchReducer) {
  act_color.h = Math.round(prevColor.h + interpolateColor(act_color.h,
emotionColor.hMax) + (moodfactorOperation ? (moodfactor *
moodMultiplyer) : moodfactor * (-1 * moodMultiplyer)));
  if (act_color.h < emotionColor.hMin) {
    moodfactorOperation = true;
    moodCount = 1;
  } else if (act_color.h > emotionColor.hMax) {
    moodfactorOperation = false;
    moodCount = 1;
  }

  // + greyer
  act_color.s = Math.round(prevColor.s + interpolateColor(act_color.s,
emotionColor.sMax));

  // + brighter
  act_color.l = Math.round(currPitch - pitchReducer)
}

function interpolateColor(color1, color2) {
  return Math.round(0.7 * (color2 - color1));
};

function averageQ(q) {
  if (q.length > 20) {
    q.shift();
  }

  var sum = q.reduce((a, b) => a + b, 0);
  return avg = Math.round(sum / q.length) || 0;
};

function calculateAverageChange(q) {
  var changeSum = 0;
  var i;

  // CHANGE AVERAGE
  for (i = 0; i < q.length - 1; i++) {
```

```
    changeSum += Math.abs(q[i] - q[i + 1]);
  }

  return changeSum / q.length;
};


function getColorStr() {

  // Convert HSL to RGB so that the SVG can be saved properly
  // Actually SVG work with HSL as well but saving does not work with
HSL in this project
  var h = act_color.h;
  var s = act_color.s;
  var l = act_color.l;

  s /= 100;
  l /= 100;

  let c = (1 - Math.abs(2 * l - 1)) * s,
    x = c * (1 - Math.abs((h / 60) % 2 - 1)),
    m = l - c / 2,
    r = 0,
    g = 0,
    b = 0;

  if (0 <= h && h < 60) {
    r = c;
    g = x;
    b = 0;
  } else if (60 <= h && h < 120) {
    r = x;
    g = c;
    b = 0;
  } else if (120 <= h && h < 180) {
    r = 0;
    g = c;
    b = x;
  } else if (180 <= h && h < 240) {
    r = 0;
    g = x;
    b = c;
  } else if (240 <= h && h < 300) {
    r = x;
    g = 0;
    b = c;
  } else if (300 <= h && h < 360) {
    r = c;
    g = 0;
    b = x;
  }
  r = Math.round((r + m) * 255);
  g = Math.round((g + m) * 255);
  b = Math.round((b + m) * 255);


  return "rgb(" + r + "," + g + "," + b + ")";
}
```

```
// Clear the paper
function clearPaper() {
  paper.clear();
  newSheet();
}

function createPaper() {

  // Create drawing Area
  // const paperElement1 = document.createElement("paper");
  var paper = Raphael("paper", WIDTH, HEIGHT);

  return paper;
}

class Line {

  constructor(pos, direction) {
    this.startPos = pos;
    this.endPos = pos;
    this.history = [];
    this.direction = direction;
  }

  update() {
    this.startPos = this.endPos;

    if (!harmony) {
      this.direction = p5.Vector.random2D();
      slowDownFactor = 15;
    } else {
      slowDownFactor = slowDownFactor / 1.5;
    }

    var longness = (energyChangeAvg + emotionChangeAvg) *
slowDownFactor;
    var newVec = this.direction.mult(longness);

    this.endPos = {
      x: this.startPos.x + newVec.x,
      y: this.startPos.y + newVec.y
    };

    var angle = 10;
    var savetyCounter = 0;

    // If the line is outside the window try to bring it back in the
screen by rotating
    while (((this.endPos.x > (window.innerWidth) || this.endPos.x < -
(window.innerWidth)) || (this.endPos.y > window.innerHeight ||
this.endPos.y < -(window.innerHeight))) && savetyCounter < 36) {

      this.endPos = {
        x: this.startPos.x + newVec.rotate(angle).x,
        y: this.startPos.y + newVec.rotate(angle).y
      };
```

```
      angle += 10;
      // Savety count
      savetyCounter++;
    }

    var lineInfo = {
      start: this.startPos,
      end: this.endPos,
      c: getColorStr(),
      e: (energy * 0.7 < 1 ? 1 : energy * 0.7),
      curvefactor: curvefactor
    }

    this.history.push(lineInfo);
  }

  // draw recently added path
  show() {

    var hist = this.history[this.history.length - 1];
    var color_rgb = hist.c;
    var e = hist.e;
    var start = hist.start;
    var end = hist.end;
    var cf = hist.curvefactor;

    if (this.history.length > 2) {

      var path = paper.path("M {0} {1} Q {2} {3} {4} {5}", end.x,
end.y, (end.x + cf.x), (end.y + cf.y), start.x, start.y).attr({
        "stroke-width": e,
        "stroke-linejoin": "round",
        "stroke-linecap": "round",
        stroke: color_rgb
      });

      // Add listeners to SVG, so manually drawing is also possible.
      path.mousemove(function(event) {
        if (mouseDown) {
          coordys = {
            x: event.clientX - leftPos,
            y: event.clientY - topPos
          };
          //drawLine();
        }
      });
      path.click(function(event) {
        if (!mouseDown) {
          mouseDown = true;
        } else {
          mouseDown = false;
        }
      });
    }
  }
}
```

7.2.2. index.html

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Ambient Sound Visualization</title>
    <script
src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
<script
src="https://unpkg.com/meyda/dist/web/meyda.min.js"></script>
    <script src="libs/raphael.min.js"
type="text/javascript"></script>
    <script src="libs/raphael.export.js"
type="text/javascript"></script>
    <script src="libs/p5.js" type="text/javascript"></script>
    <script src="libs/p5.sound.js" type="text/javascript"></script>
    <script src="libs/stereo-analyser-node.js"
type="text/javascript"></script>
    <script src="libs/fRaphael.js" type="text/javascript"></script>
    <link href="style.css" rel="stylesheet" type="text/css">
  </head>

  <body>
  <div id="paper", class="paper">
      <h1>Ambient Sound Visualization</h1>
      <button id="addEmotion" class="myButtons">add mood</button>
      <button id="clear" class="myButtons">clear</button>
      <button id="savePaper" class="myButtons" >save mood</button>
      <button id="doItInCircles" class="myButtons">do it in
circles</button>
      <button id="darkMode" class="darkButton">dark mode</button>
      <button id="draw4me" class="draw4me">draw for me</button>
    </div>

    <script src="drawing.js"></script>
  </body>
</html>
```

7.2.3. style.css

```
.myButtons{
   font-size: 16px;
   font-family: "Lucida Console", "Courier New", monospace;
   text-align: center;
   height: 40px;
   width: 180px;
   border: none;
   background: transparent;
   top: -1050px;
   position: relative;}

.darkButton{
  background-color: midnightblue;
  font-size: 16px;
  font-family: "Lucida Console", "Courier New", monospace;
  text-align: center;
  height: 40px;
  width: 160px;
  color: white;
  visibility: hidden;
  top: -1050px;
  position: relative;
}

.draw4me{
   background-color: transparent;
   width: 100px;
   font-size: 16px;
   font-family: "Lucida Console", "Courier New", monospace;
   text-align: center;
   height: 40px;
   width: 180px;
   -- border-radius: 50%;
   border: black
   color: black;
   top: -1050px;
   position: relative;
}

.paper{
   background-color: white;
   border: none;
   position: relative;
}




h1{
  font-size: 40px;
  color: black;
  stroke: white;
```

```
  font-family: "Lucida Console", "Courier New",
monospace;
  font-weight: bold;
  text-transform: uppercase;
  background: transparent;
  top: -1050px;
  left: 50px;
  position: relative;
}
```