University of Bergen
Department of Informatics

Line Harp: Importance-Driven Sonification for Dense Line Charts

Author: Egil Bru
Supervisors: Stefan Bruckner

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

May, 2023

Abstract

Accessibility in visualization is an important yet challenging topic. Sonification, in particular, is a valuable yet underutilized technique that can enhance accessibility for people with vision deficiencies. However, auditory information is presented sequentially, and for this reason, interactivity is key in making full use of its potential. In this thesis, we present a novel approach to visualization accessibility that utilizes a sonified interactive lens and selections in the context of dense line charts. This approach takes advantage of the inherent directions of lines for sonification and dynamically scales amplitude for improved density perception. Combined, this produces a visualization that not only uses sonification as an additional information channel but also as a complementary element. Finally, we discuss the potential of our contribution based on a set of case studies.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Stefan Bruckner, for his unwavering guidance and support throughout this thesis. I am also deeply thankful to Thomas Trautner for his assistance with the thesis and implementation, as well as for his creative ideas. I would also like to thank friends and family for their unwavering encouragement and support during this academic journey. Lastly, I want to give special thanks to my significant other for her unwavering support and helping me overcome the challenges faced during this thesis.

Egil Bru
Wednesday 31st May, 2023

Contents

Chapter 1

Introduction

Data visualization, the art of transforming complex data into visually intuitive representations, has emerged as a powerful tool for facilitating data exploration and comprehension. Visualizations primarily target the visual channel of the user, as it is the highest bandwidth information channel in the human perceptual system. However, despite visual displays becoming the standard in data analysis, there is a significant population that faces challenges in fully accessing and interpreting visual information.

Visual displays inadvertently exclude individuals with visual impairments or those who struggle with visual processing. By incorporating audio into data visualization, accessibility can be greatly improved, ensuring that everyone can access and comprehend data-driven insights. Audio cues, descriptions, and interactive auditory elements provide an alternative channel for perceiving and understanding data, offering a complementary approach. Humans are equipped with a powerful listening system which is often underutilized in data visualizations. The act of identifying sound sources, spoken words, and melodies, even under noisy conditions, is a supreme pattern recognition task often overlooked in data visualizations. The fact that it appears to work so effortlessly is perhaps the main reason that we are not aware of the incredible performance of auditory systems, and perhaps why it is often overlooked.

In general, extensive research has been conducted on accessibility within data visualizations [48, 50]. Several distinct approaches exist for different purposes, such as physicalization [45] and haptification [60]. However, one prominent yet underutilized tool for improving accessibility in data visualization is sonification [42]. Sonification offers the potential to enhance visualization accessibility by providing an additional channel to encode important information. Nevertheless, due to the limited degree of parallelism in auditory information, effectively conveying complex information in a responsive and efficient manner can be challenging.

1.1 Contribution

This thesis proposes a novel interactive approach for sonifying dense line charts. Our approach incorporates a frequency audio mapping that leverages the inherent directionality of lines to improve angle perception and enhance navigation. Additionally, we propose the use of dynamically scaled amplitude to improve the audible perception of density in clustered lines. By mimicking the responsiveness of real-life stringed instruments, our goal is to enhance the user experience for those with visual impairments. The main contributions of our work can be summarized as follows:

- We introduce a direction-based frequency mapping to improve angle perception and graph navigation.
- We present dynamically scaled amplitude, which adjusts the audio based on data density, improving the perception of density and clusters.
- We include various interactive tools to reduce overall complexity: a lens that elevates overlapping clusters and reduces the required sonification, and a selection tool for focusing on outliers or gaining more details.

Chapter 2

Related Work

In this chapters we outline related work conducted on accessible visualizations design. Our attention is directed towards sonification approaches that leverage auditory cues for the purpose of visual data analysis. Moreover, we will explore interactive lenses both in terms of visuals and audio.

2.1 Accessibility

Data visualization is the process of using visual elements such as charts, graphs, and maps to represent and communicate data. The goal of data visualization is to enable people to easily interpret and understand patterns, trends, and insights from data, which can help inform decision-making, support analysis, and improve communication. However, visualizations are visual in nature, which can create barriers for users with visual impairments or other disabilities. To ensure that all users can access and understand data visualizations, it is important to design visualizations with accessibility in mind [48]. While there are several guidelines and best practices for designing accessible visualizations [50], some users may still face barriers due to their individual needs and abilities. There exist many different limiting factors that can influence the accessibility of visualizations (e.g., motor, visual, cognitive impairment), however in this thesis we primarily focus on visual impairments. There is in fact a significant population of individuals who experience visual impairments, which encompass a wide range of conditions. According to the World Health Organization (WHO) [59], there are at least 1 billion people worldwide who suffer from visual disabilities, encompassing a spectrum from moderate to severe visual impairments or blindness.

Color blindness is one of the most common visual impairments, and as such it has received significant attention and development within the field of data visualization. Color blind friendly palettes have been a standard in many visualization programs [72] and libraries [53] for many years. In contrast, visual impairments such as reduced vision or even full blindness have been relatively under developed. The term visual impairment refers to reduced visual acuity of the visual field, ranging from blindness to low vision [50]. There are various vision symptoms, such as blurred vision, loss of central or peripheral vision, and extreme light sensitivity [59]. Figure 2.1 simulates some examples of seeing a data visualization with vision disabilities.

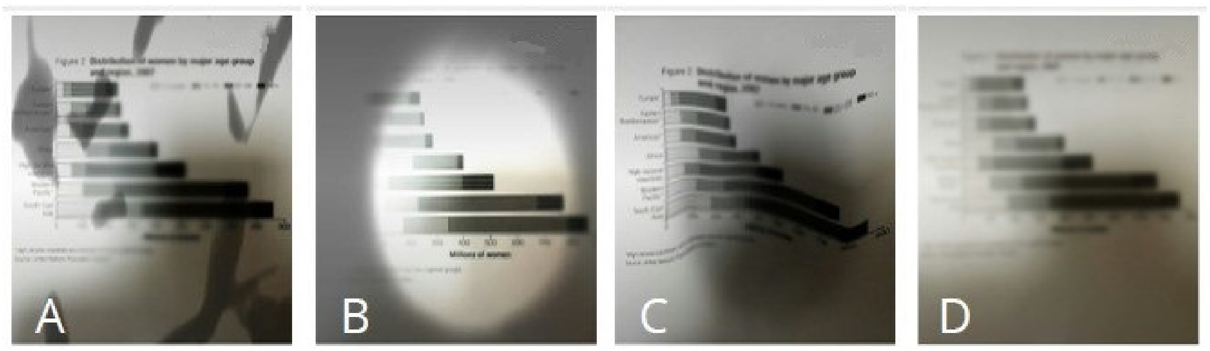


Figure 2.1: (A) Clouded and spotted vision as caused by Diabetic Retinopathy, (B) loss of peripheral vision as caused by Glaucoma, (C) loss of central vision as caused by Macular Degeneration and (D) blurry vision as caused by Cataracts. Figure from Kim et al. [50].

Physical representations, also known as *physicalizations*, offer tangible and spatial ways to explore, navigate, and interact with data. They have the potential to make data analysis more accessible for visually impaired individuals [45]. Traditional approaches have employed raised paper to convey charts and maps [28, 22], resulting in mostly static physicalizations. However, more dynamic physicalization methods have been explored, such as using 3D printers to create raised lines quickly [73], shape-changing interfaces [66], and robotics [51]. Nonetheless, these approaches often require specific devices and vary in their application across different scenarios.

In contrast, *haptification* focuses on incorporating haptic feedback and touch-based sensations into virtual models. Similarly, this technique can be employed to present information to blind or visually impaired users. Haptic data visualizations utilize abstract models that encode numerical values or abstract concepts rather than a physical environment [60]. Haptification has been explored in various applications, including line graphs [86], maps [47], and bar charts [85], with recent emphasis on its integration with virtual reality [68]. Optimal utilization of haptification often necessitates specific devices such as gloves or exoskeletons to produce force feedback. Conversely, sonification only requires an audio output device and can offer greater dynamism compared to haptification and physicalization.

2.2 Sonification

Sonification is a technique for representing data using sound, which can be particularly useful for users with visual impairments or for those who may benefit from a multimodal approach to data analysis. The Sonification Handbook [42] provides a comprehensive overview of the state of the art in sonification, including theory, practice, and applications contributed from several researchers in the field.

The interdisciplinary nature of sonification, as depicted in Figure 2.2, means that it is easy to become disoriented and overwhelmed when confronted with its many different facets, ranging from computer science to psychology, from sound design to data mining. For clarification sonification is a component of the *Auditory Display* domain, which encompasses all aspects of a human-machine interaction system, including the setup, speakers or headphones, modes of interaction with the display system, and any technical solution for the gathering, processing, and computing necessary to obtain sound in response to the data [41]. Specifically, sonification involves the technique of rendering sound in response to data and interactions.

Generally sonification has numerous real-life applications beyond that of data visualization like air traffic control [19] and medical monitoring equipment [69]. Furthermore, sonification has been extensively researched for real-time monitoring applications, including monitoring EEG signals [40] and the stock market [44]. However, these approaches are time-dependent and solely provide audio feedback for monitoring or altering purposes. These methods are often referred to as *audification*, which is defined as the direct translation of a data waveform into sound [41]. In contrast, sonifications often utilize abstract data-audio mappings.

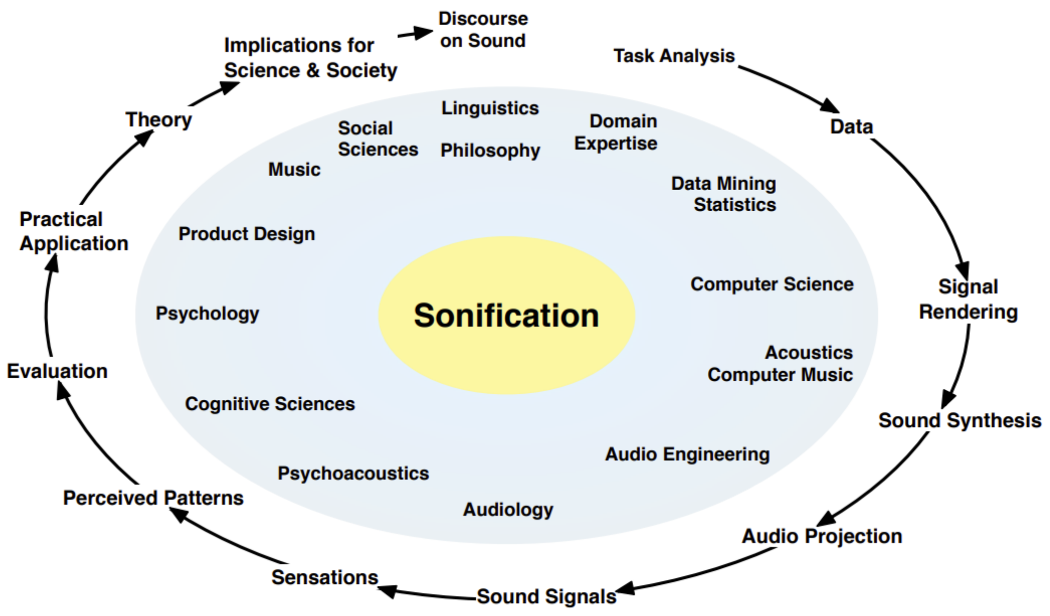


Figure 2.2: The interdisciplinary circle of sonification and auditory displays: the outer perimeter depicts the transformations of information during the use cycle, the inner circle lists associated scientific disciplines. Figure from Hermann et al. [41].

2.2.1 Auditory Icons

Auditory icons attempt to mimic everyday non-speech sounds that we might be familiar with from our everyday experience of the real world [56]. These are commonly associated with desktop interfaces, such as deleting a document being represented with the sound of crumbling paper. For instance *SonicFinder* [33] was incorporated with the Apple operating system's file management application using auditory icons. The strength of the *SonicFinder* was that it reinforced the desktop user interface metaphor, which enhanced the illusion that the components of the system were tangible objects that could be directly manipulated [41]. An evaluation of auditory icons was carried out to see whether they are an effective means of communicating information in sound and found that musical notes were more effective than simple tones [8]. In our approach, we build on this concept by representing our lines as musical strings, providing an abstract representation that further reinforces the audio-visual experience. However, it is important to note that auditory icons often serve specific purposes and may not always function effectively with complex data visualizations. Therefore, careful consideration is necessary when incorporating them into intricate visual representations.

2.2.2 Parameter Mapping Methods

Parameter mapping methods is widely used and is perhaps the most established technique for sonifying data. This technique involves mapping data to sound elements such as pitch, amplitude and timbre. The sound can either be used as an addition to the visualization or as a complement to it. In this section, we will review previous work and evaluations on parameter mapping methods for data exploration.

Substantial research have been conducted on evaluation of good or bad mappings [31]. Further research has been conducted to evaluate which types of data and polarities are more naturally mapped to particular sound attributes [79, 80]. The studies found that an increase in the sound dimension (rising frequency), should represent a natural increase in the data dimension, (rising temperature). Additionally, it was suggested that non-visually impaired participants preferred different polarities than visually impaired participants. Both studies were evaluated based on magnitude estimation tasks. However, parameter mapping has also been shown to be useful for interpreting line graphs.

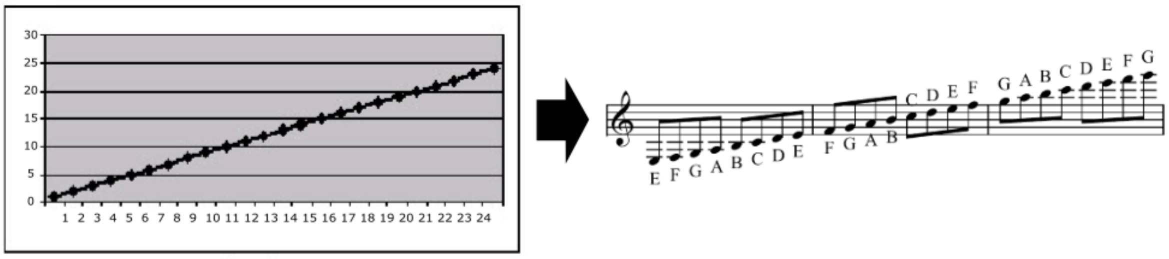


Figure 2.3: Depiction of a common approach to parameter mapping sonifications. As the y-value increases the pitch of the musical note gets higher. Figure derived from Brown et al. [12].

Research has previously demonstrated that participants can effectively interpret line graphs through only sonification, by mapping each data point to a musical note (frequency) and moving along the x-axis (time) [32] (see Figure 2.3). Participants were tasked with redrawing single line graphs based on a audio playback of the inherent data. Expanding on this one study found that participants were able to successfully redraw multiple lines and find intersections while listening to two separate lines in each ear [11]. Furthermore, even higher-dimensional audio environments have been proposed for visualizing even more lines [12]. However, the effectiveness of this approach is severely limited by the density of the data. As the number of lines being sonified increases, audio noise also increases, which can make it difficult for users to distinguish individual data points.

Moreover, parameter mapping lacks the interactive capabilities that are typically associated with modern visualization models. Common concepts such as pause, play, back, and forward may aid in the navigation of the parameter based graphs [10], but they do not fully address the limitations of the audio environment for complex visualizations.

2.2.3 Model-Based Approach

To address the aforementioned issues associated with parameter mapping approaches, researchers have proposed a model-based sonification approach [82]. This method differs from parameter mapping, as it involves transforming the data set into a dynamic model that can be interactively explored by the user, rather than directly sonifying the data. The concept of model-based sonification was first introduced by Hermann and Ritter [37]. There are various variations of this method with different interaction techniques, as its application can be highly specific to the task and problem at hand [41]. The underlying principle of model-based sonification stems from the observation that most of our daily interactions are accompanied by an acoustic response, suggesting that sonification can also benefit from this concept.

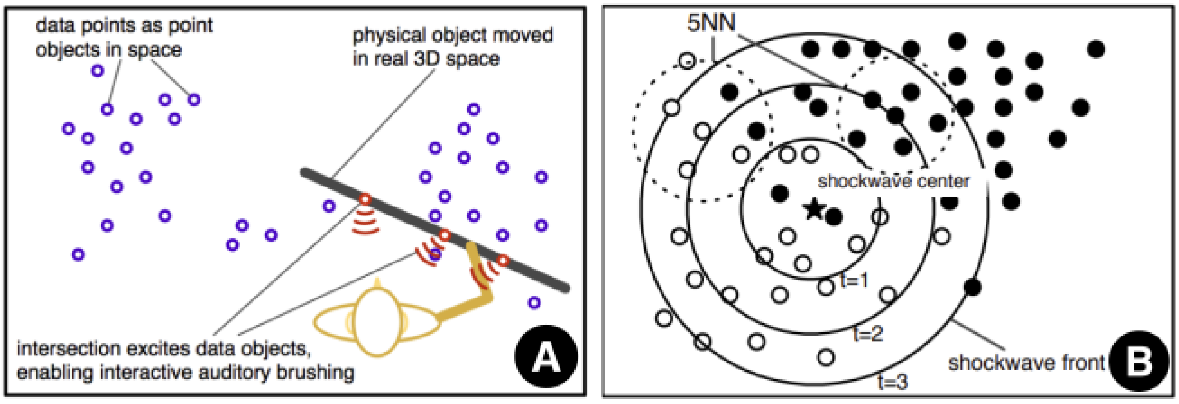

data points as point objects in space
physical object moved in real 3D space
intersection excites data objects, enabling interactive auditory brushing
A
5NN
shockwave center
t=1
t=2
t=3
shockwave front
B

Figure 2.4: Demonstration of two model-based approaches. From Bovermann et al. [7], (A) depicts a tangible line that can be oriented, enabling the intersection of the line with data points. From Herman and Ritter [37], (B) shows a shock wave gradually expanding to sonify data points intersecting with the wave. Figures from Hermann [38].

Building upon interactive concepts from parameter mappings, Bovermann et al. [7] propose an interactive tool that allows for altering the sequence in which sonified data is presented. Their approach involves using a tangible plane that can be oriented and positioned through touch or sliders, facilitating the intersection between the plane and data points. The intersection excites the data objects, enabling interactive auditory brushing. Figure 2.4 provides a visual depiction of this approach, showcasing the use of a line to intersect data points.

Another approach suggests utilizing the principle curve technique to compute a smooth path through the data set [39]. Users can then navigate this path sequentially, receiving audio feedback along the way. Both techniques heavily rely on the density and distribution of the data. Many previous model-based sonification approaches employ new and complex interaction methods, making them challenging to learn and use, often specific to particular domains. In our approach, we aim to address this by focusing on simple interactive methods that most analysts are familiar with. Additionally, we incorporate established interactive tools such as lens and selection to reduce the complexity of the user experience.

2.3 Interactive Lenses

In visualization, interactive lenses are an well-established class of visualization methods that facilitate multi-faceted data exploration [4]. With interactive lenses, the visualization can temporarily be altered to show more details or different arrangements [75]. In the data visualization field there exists many different types of lenses that achieve varying results depending methods. In this section we will mainly cover visual lenses that use distortion techniques.

Lenses can distort edges in graphs to reduce edge congestion. The approach by Wong et al. [84] uses a lens that bends edges based on a bubble or splines. The key attributes that their approach does is that it maintains the original layout and reveals hidden information underneath the graph structure. Likewise, a lens can distort 3-dimensional data, whereby the distortion clears a visual path to the focus. The approach by Carpendale et al. [21] works by distorting data points that are obscures in a 3-dimensional model or dataset and is further demonstrated in Figure 2.5. Their lens utilize the notion of a lens depth that can be toggled for specific data points in focus. Our approach aims to combine already established visualization techniques for lenses with the addition of sonification.

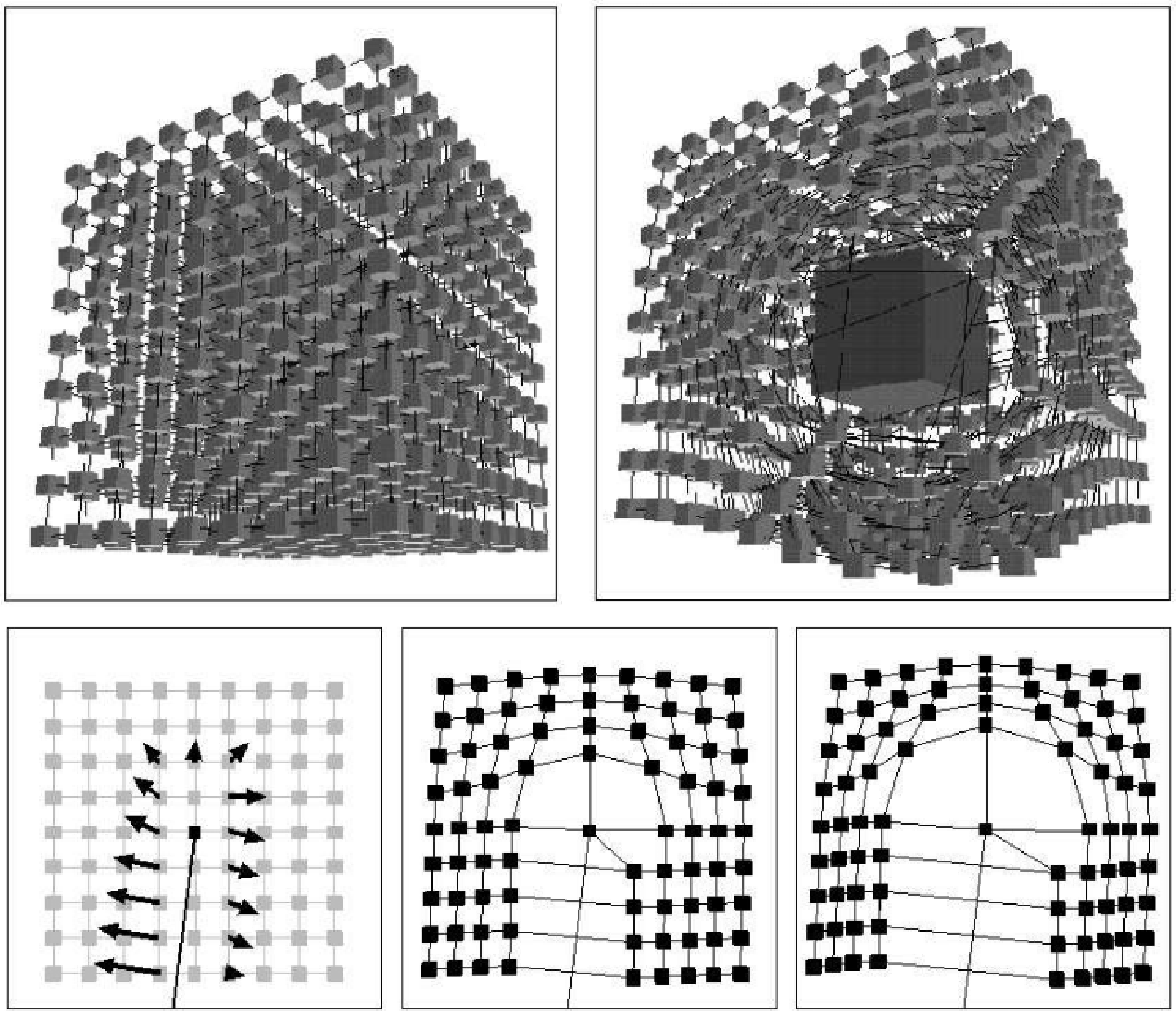


Figure 2.5: 3-dimensional distortion technique presented by Carpendale et al. [21]. Shows ordinal lattice with and without distortions and a 2D cross-section of a view aligned distortion.

2.4 Shock Wave Approaches

Radial sonification have already been attempted in the context of model-based sonification. Herman and Ritter [37] proposed using a shock wave, as depicted in Figure 2.4, that gradually enlarges to sonify data points that intersect with the wave. This technique was later expanded upon with multi-touch interactions [77]. While this approach was not originally intended for lenses, it can be effectively applied in this context. As shown by Enge et al. [29], where a shock wave was used within the lens to sonify intersecting data points in focus (see Figure 2.6). While the shock wave approach can be effective for certain types of visualizations, such as scatter plots, it is less suitable for line charts. This is because the shock wave approach does not effectively convey relationship information.

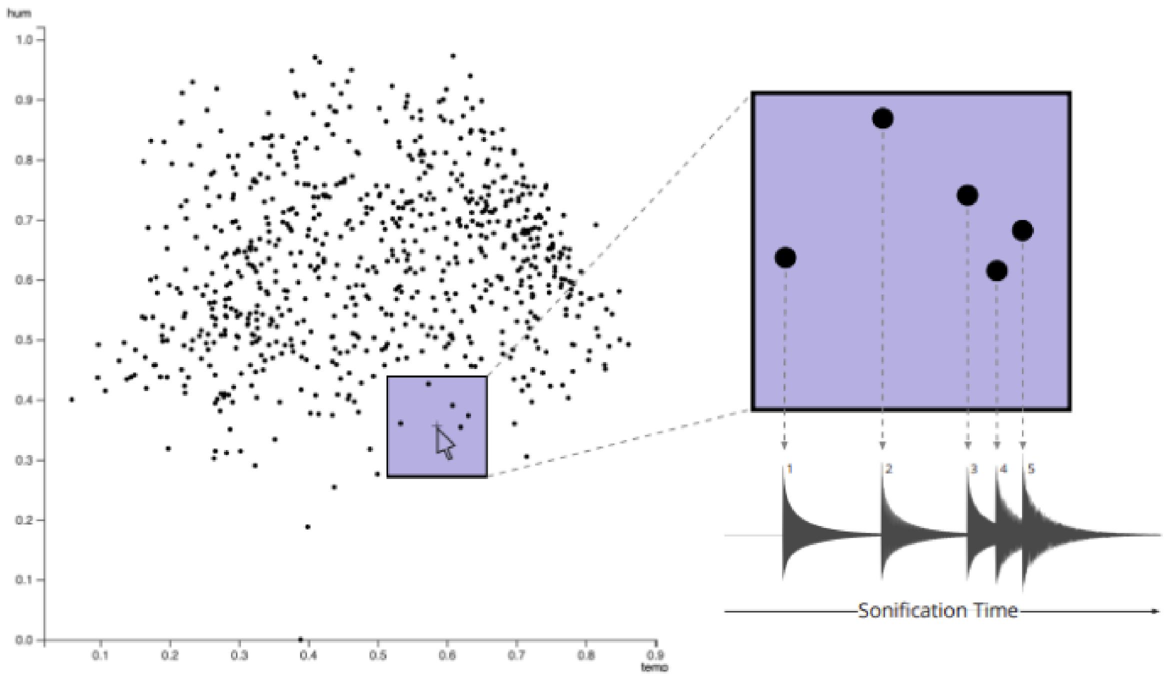


Figure 2.6: Demonstration of SoniScope from Enge at al. [29]. Users specify an area in the scatterplot that will be sonified and a mouse click triggers the sound generation.

Chapter 3

Methodology

Line Harp introduces a novel approach to enhance accessibility in the visualization of dense line charts by utilizing sonification. The objective of Line Harp is to improve the accessibility of dense line chart visualizations through the application of sonification. However, sonifying complex visualizations can be challenging due to the limited bandwidth of the auditory channel. To address this challenge, we also present additional interactive tools aimed at reducing complexity and improving the experience.

In this chapter, we first provide the background for our method, focusing on the visual approach, which is based on Line Weaver as presented by Trautner and Bruckner [76]. Subsequently, we introduce our sonification approach, highlighting its necessity, functionality, and how it works conceptually. Finally, we present our supplementary interactive tools, including a lens and selection mechanisms.

3.1 Background

We base our approach on the importance-driven visualization technique for dense line charts presented by Trautner and Bruckner [76]. This method uses the notion of an importance function associated with each line, which allows them to interweave individual lines such that the most important lines segments occlude those with lower importance.

Following Trautner and Bruckner [76], we regard line data as set $D = \{L_1, L_2, ..., L_n\}$ of N polylines with its members $L_i = (P_1, P_2, ..., P_m)$ represented as tuples of M ordered two-dimensional points $P_i = (x_i, y_i)$. The resulting parametric curve $l_i(u)$ of each member is a polyline generated by linear interpolation between its associated points. Furthermore, we use their notion of an importance function $\beta_i(u) \in [0,1]$ which associates a scalar importance value with every position along the curve. In practice, there are many different ways to define importance, such as using underlying data, results of a features detection algorithm, or fundamental properties of the lines. In their paper, for instance, Trautner and Bruckner [76] present an algorithm that generates importance values based on a heuristic optimization of screen space utilization for multiple sets of lines within the same chart.

Line Weaver addresses the problem that the rendering order can substantially affect the resulting visualization, when the number of lines in a chart increases. While normally transparent lines can be useful, the current approaches of ignoring or naively using the blending order can lead to misleading visuals. To overcome this issues they proposed a technique utilizing the aforementioned importance function that correctly considers the blending order without any prior sorting of the data. This is further demonstrated in Figure 3.1. The effectiveness of their approach was demonstrated through experiments on both synthetic and real-world data sets, where traditional or naive approaches would fail. Overall, their approach is an effective way to visualize dens line charts, and we therefore explore the use of interactive sonification to make it more accessible and potentially further improve the utility.

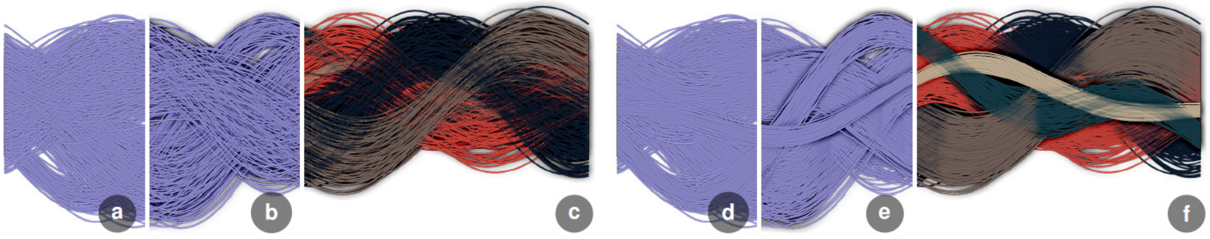


Figure 3.1: Line weaver was inspired by techniques from textile production where multiple threads are interwoven to form fabrics. If the blending order is (a) ignored or naively used, essential visual information is lost, even if (b) outlines and halos are added or (c) clusters are colored. If, however, the ordering of clusters is (d) optimized, both (e) outlines and halos as well as (f) colors can help perceiving clusters [76].

To enhance accessibility for Line Weaver and line charts in general, our investigation focuses on various techniques for incorporating sonification. These chosen techniques are guided by specific objectives identified as crucial for enhancing accessibility, both in a general sense for line charts and specifically for Line Weaver. Our objectives are also derived in the understanding that our approach should be applicable to both visual and auditory experiences. The following objectives summarize our aims:

Meaningful audio interactions: The Audible response produced by interaction should feels natural and important.

Audible density: The visual density should correspond to the audible density, while maintaining dynamism to capture information about outliers and individual lines.

Angle perception: The trends and characteristics of clusters and individual lines should be conveyed solely through audio frequency.

Audio navigation: The frequency mapping should improve graph navigation for users with visual impairments.

Reducing audio information with a lens: The lens should serve as a tool to minimize the amount of sonification, thereby reducing the required user bandwidth.

Overplotting reducing lens: The lens should be employed to address overplotting issues in dense line charts, both in terms of visual distortions and the audio channel.

Audio filtering with selection: Simple selection tools should be available for further filtering the sonification of single lines or clusters. These filtering capabilities should be applicable to both visual and audio representations.

3.2 Line Harp

While prior research has primarily explored static parameter mapping for sonification of line charts [11, 32], we argue that an interactive approach is better suited for the characteristics of dense line charts. As audio allows for limited parallelism, a static playback of dense data would either need to be excessively lengthy or too fast for proper perception, or require considerable abstraction. Considering the prevalence of acoustic responses in our daily interactions, it is reasonable to expect that data visualization can benefit from incorporating audio.

In real-world scenarios, audio feedback is influenced by various factors, such as material properties and interaction types. For example, hitting a table or throwing a rock produces distinct acoustic signatures that are distinguishable. Audio perception is typically an automatic and effortless process, with the transformation from audio to ”meaning” occurring within a fraction of a second [57]. Our interaction approach is based on line-mouse intersections, which is a natural extension of how humans interact with stringed instruments like the harp. However, since we are not constrained by real-world physics, we can generate audio based on any desired measure or magnitude.

In our approach, moving the mouse cursor ”plucks” the intersected lines, resembling the action of playing an instrument such as a harp. This reduces overall complexity, as most individuals have had some experience with stringed instruments. However, playing musical instruments can often be challenging, and similarly, sonification has been found to require more training than regular visualizations for complex data [71, 81]. To address this, we incorporate additional interactive tools based on established techniques, such as lenses and selections, to reduce complexity.

The frequency of sound generated by plucking a string in a real-world musical instrument is influenced by various factors, such as the material properties of the string and the tension it is under. For the purpose of sonification, we use frequency to encode the directionality of the corresponding line segment. In line charts, with their common left-to-right reading order, this is relevant in the identification of salient features such as trends (e.g., an upwards or downwards development in a time series). Amplitude is similarly an expressive audio channel and is naturally mapped to an increase in magnitude. Our amplitude is mapped to the notion of importance, which is further described in Section 3.1.

3.3 Sonification

In this section, we will introduce our sonification approach, encompassing a directional frequency mapping and dynamic amplitude based on importance. We will provide a comprehensive explanation of the purpose and necessity of our sonification approach, as well as outline how it works conceptually.

3.3.1 Frequency

In music, the frequency of a sound refers to the property that most determines its pitch [62] and is perhaps the auditory dimension most frequently used to represent data. The major advantage of using frequency is that it can easily be mapped to changes in data. However, previous work suggests that individual differences in musical ability impact the effectiveness of perceiving frequency changes [57]. We utilize the inherent directions of line segments for mapping frequencies. In contrast our frequency mapping can instead be used for identification of salient features such as trends. Additionally, it serves a tool for navigating the visualization especially for users with visual impairments.

In order to understand the concept of frequency, it is important to differentiate between the musical term "pitch" and the scientific term "frequency". While they both describe the same underlying phenomenon, they are not synonymous. Pitch is a relative measure commonly used in music and relies on consensus among musicians, whereas frequency is an objective, absolute measurement [54].

When a musical note is played, it produces a composite audio wave composed of multiple sinusoidal waves with different frequencies [54]. These individual sinusoidal waves are known as partials or overtones. Each partial corresponds to a specific frequency and contributes to the overall timbre or tone quality of the audio. Among these partials, the lowest frequency is called the fundamental frequency. It directly relates to the perceived pitch of the note. The concept of the fundamental frequency and partials/harmonics is illustrated in 3.2, where a composite audio wave is depicted as the combination of three sinusoidal waves with distinct frequencies. The combination of multiple frequencies is what gives a sound its characteristic quality[58].

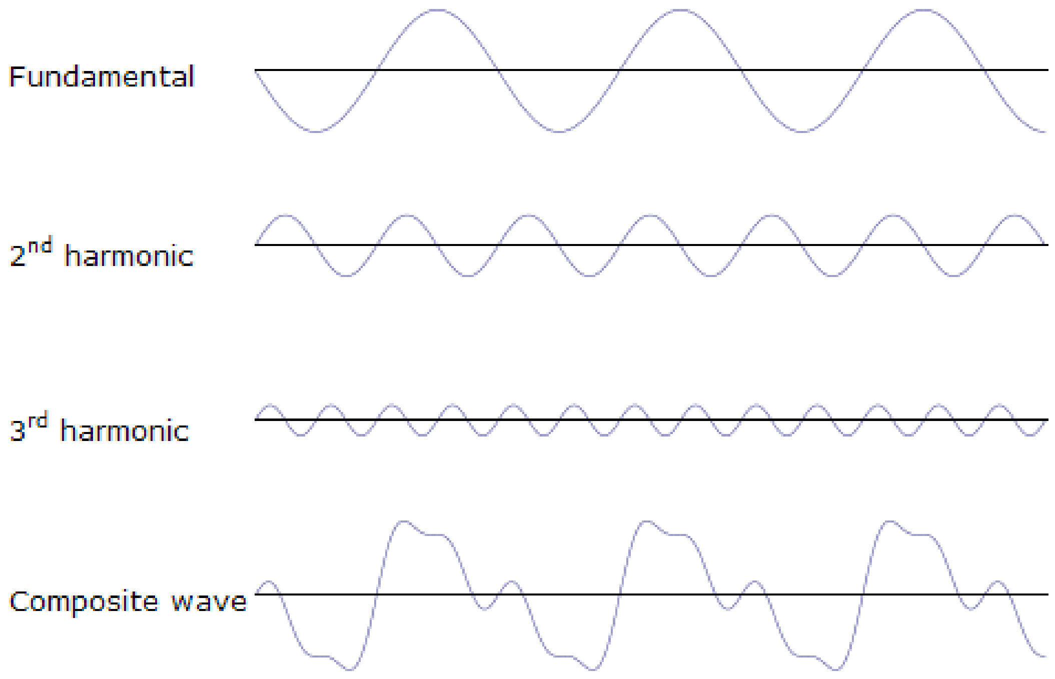


Figure 3.2: Demonstration of how a composite audio wave consists of partial tones (harmonics) and how the fundamental frequency (pitch) is the lowest frequency. Figure from University of North Carolina [58].

Due to the aforementioned perceptual characteristics of our ears, directly mapping numerical frequency values to angles does not yield a meaningful auditory experience. n contrast, mapping angles to discrete music notes more closely resembles how instruments produce sound and more importantly how humans perceive frequency. Therefore, we employ Scientific Pitch Notation (SPN) [43] for frequency mappings, which combines a musical note name and an octave number (e.g., B_4, $A_2^{\#}$, and A_4).

Each musical note corresponds to a specific frequency that is mathematically related to each other and is defined in relation to a central note, A_4, which is set at 440 Hz [43]. The SPN spans a range from 8.1 Hz to 31 KHz, but through experimentation, we determined that limiting the scale to a range of 110 Hz to 880 Hz (A_2 to A_5) produced the most pleasing auditory results. Furthermore, this range is commonly associated with stringed instruments and provides sufficient variation for our purposes. Figure 3.3 illustrates the musical pitch scale, where the highlighted red points indicates our frequency range.

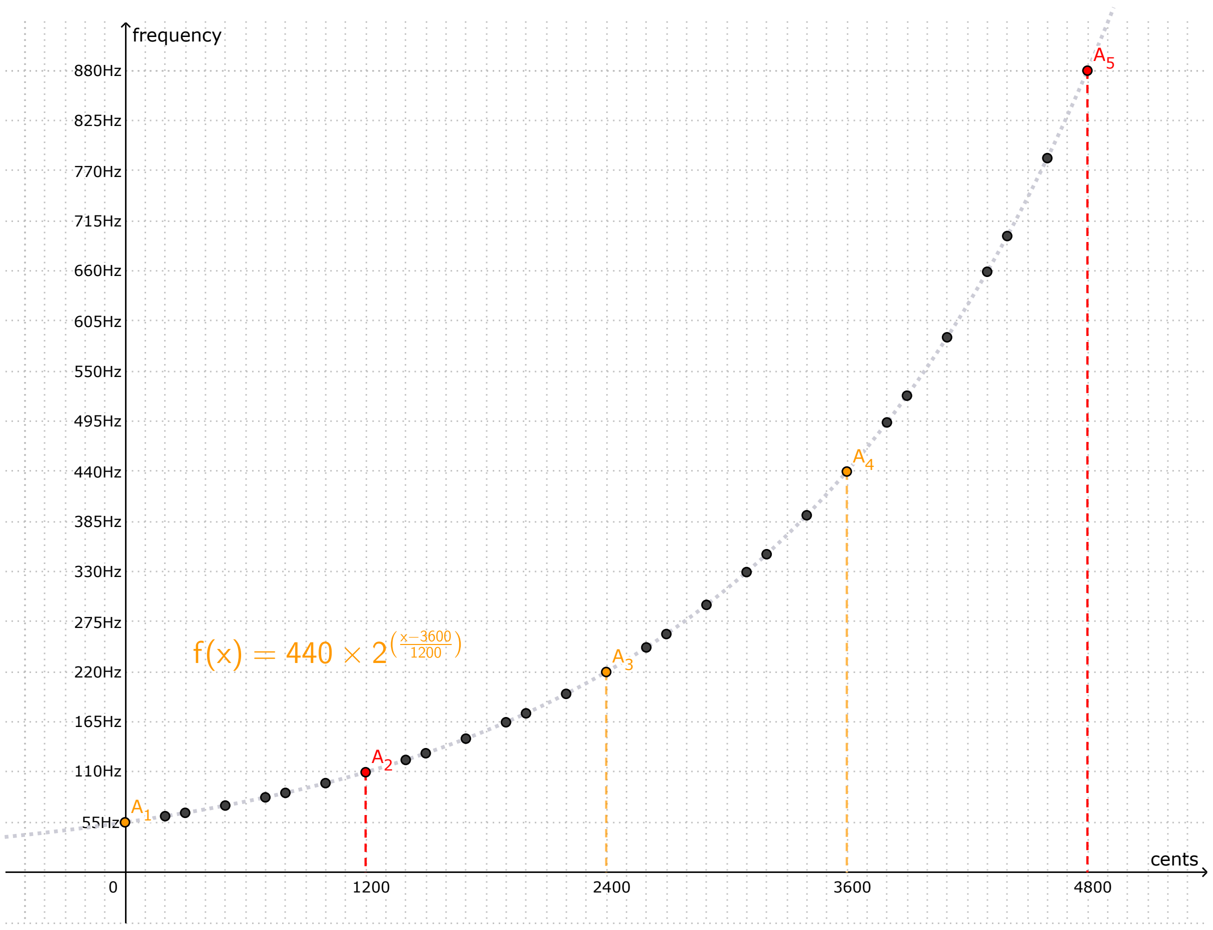


Figure 3.3: The frequencies and value in cents tuned to central pitch A_4. Twelve-tone equal temperament divides each octave into 12 semitones of 100 cents each, where the 6 distinct semitones are visualized as grey data point ($B_n \rightarrow G_n$). Figure derived from [25]
.

In the context of sonification, selecting the appropriate polarities for mapping audio parameters is essential. Polarities, in this context, refer to the directionality of a data attribute. For instance, ”positive polarity” maps lower values in a channel to lower data values and higher values in a channel to higher data values; ”negative polarity” maps the opposite [83]. For sonification establishing a logical relationship between changes in audio and corresponding changes in the data is crucial [57]. Previous studies have indicated that an increase in the sound dimension (e.g., rising frequency) should correspond to a natural increase in the data dimension (e.g., temperature) [79, 80]. Following this principle, our approach adopts a positive polarity mapping, where an increase in angle ideally corresponds to an increase in frequency. As a result, in our sonification , frequencies are positively correlated with angles.

For the directional mapping of frequencies, as mentioned earlier, we utilize the direction of a line segment to control the generated frequency. In practice, frequencies are mapped to degrees ranging from 0 to 180. Given the typical left-to-right reading order of line charts, straight upward lines are mapped to the highest frequencies, while straight downward lines are mapped to the lowest frequencies. Mapping angles exceeding 180 is redundant since the angle information is preserved, and the frequencies instead adopt a negative polarity mapping. Figure 3.4 depicts a polar chart, where frequency is mapped to angles from 0 to 180.

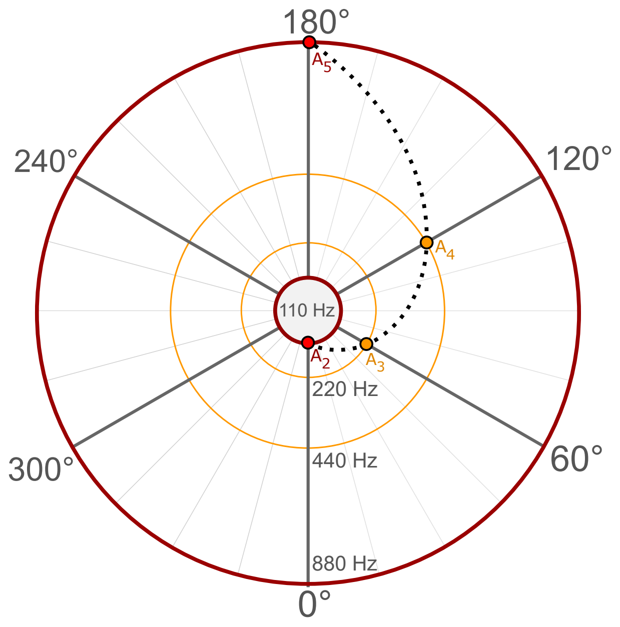


Figure 3.4: Polar chart demonstrating our directional frequency mapping. Each circle indicates a step in octave and is colored based on Figure 3.3.

3.3.2 Amplitude

Loudness is a perceptual dimension that correlates with the amplitude of an acoustic signal. The use of loudness change in sonification, although not as common as the use of frequency change, is nonetheless ubiquitous [57]. However, similar to frequency, loudness can pose challenges in discriminating between different intensities and lacks the resolution required to represent precise magnitudes. Additionally, the perception of loudness is influenced by other audio dimensions, such as frequency and timbre, leading to inconsistencies. To address these issues, we propose a mapping of loudness to a measure that does not necessitate exact estimations, such as the importance of individual lines.

Another crucial consideration in visually dense line charts, particularly parallel coordinates [36], is overplotting. Overplotting refer to the situation, where multiple data points or lines overlap or cluster closely together, making it difficult to distinguish individual elements and extract meaningful information from the chart. This issue is also applicable to our sonification approach and the sonification of dense line charts in general. From an audio perspective, overplotting occurs when too many lines are sonified, leading to a noisy and loud output. To address this, we propose utilizing the importance value, as outlined in section 3.1, to regulate the loudness of the note. In other words, lines that are visually occluded due to their lower importance will also be deprioritized in the sonification process, mitigating the issue of overplotting in the auditory representation.

Loudness or amplitude is typically quantified using decibels (dB), a logarithmic unit that represents the relative intensity or power of a sound wave. The relationship between decibel and amplitude is logarithmic rather than linear. This is because the human perception of sound intensity follows a logarithmic response [20]. Recognizing this characteristic, we have determined that an exponential mapping of importance to amplitude is more appropriate than a linear mapping. By utilizing an exponential map to scale the amplitudes, the non-linear nature of perceived loudness can be accurately captured, aligning with human perception.

In most cases, auditory feedback is presented sequentially, which essentially requires the user to wait for the next note. However, this approach is not ideal as it contradicts the swift response time of visual interactions, which typically require a response within 50-100ms [74], equivalent to playing notes at a tempo of 1200-600 beats per minute (bpm).

A study on auditory inspection time, where participants where tasked with discerning the difference between a high tone and a low tone, found that 95% of participants responded correct when the frequency changed every 100-200 ms[46]. However, since the study solely focused on the ability of participants to distinguish between two easily discernible frequencies, its findings may have limited applicability to our specific context. Nevertheless, the results suggest that slower changes in frequencies may not always be optimal for perception.

As mentioned in Section 3.3.1, a composite audio wave is composed of multiple sinusoidal waves with distinct frequencies. This principle also applies to the accumulation of different sounding notes [49]. An analogy can be drawn to a guitar, where playing musical chords involves plucking multiple notes simultaneously. For example, when the notes C, E, and G are played together, a C major chord is produced. In practice, this entails adding the sinusoidal components of all currently active notes, with ”active notes” referring to objects that are currently vibrating and producing sound. Objects capable of producing sound, such as guitar strings, exhibit a more complex spectrum compared to pure sine waves. This complexity contributes to a more pleasing and perceptually distinguishable sound [65]. In our approach, we utilize synthesized instrument to generate sound, incorporating the previously mentioned principles.

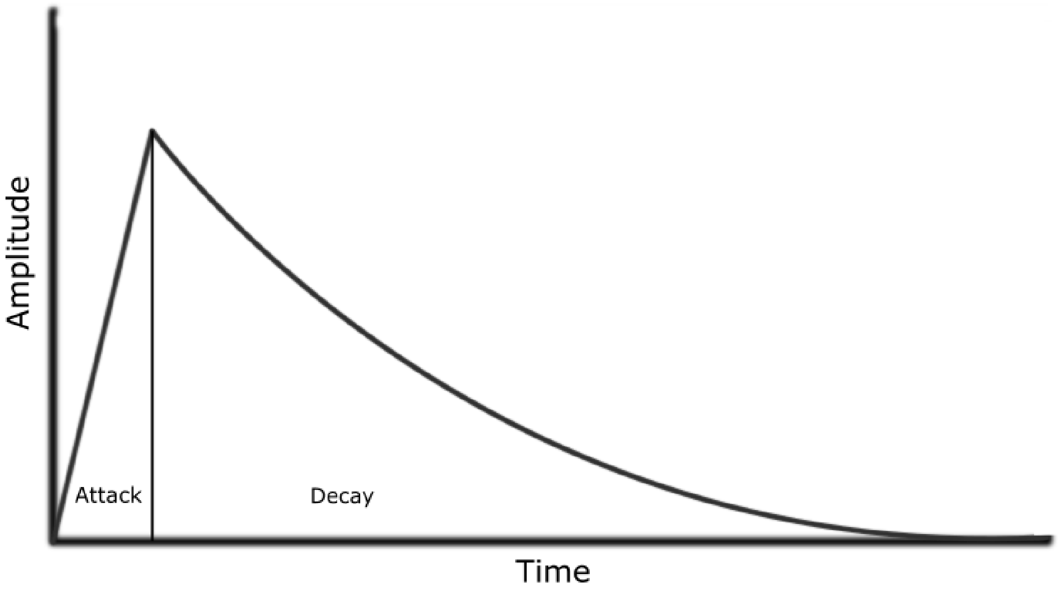


Figure 3.5: The Attack and Decay (AD) envelope commonly used in producing plucked string sound. Attack represents the initial build-up of sound, while decay characterizes the gradual decrease in amplitude over time.

In the realm of music and sound, the envelope refers to the dynamic changes a sound undergoes over time, and different instruments may exhibit distinct envelopes (see Figure

3.5). For instance, when a piano key is struck and held, it produces an immediate initial sound that gradually diminishes in volume until it reaches silence.

To convey the acoustic impression of our lines, we have opted to utilize the sound of a plucked string instrument. Previous studies have employed various strategies, such as using different instruments to represent different lines [11] or assigning different instruments to separate clusters [67]. In contrast, our approach involves employing a single instrument that remains consistent throughout, with only the frequency and amplitude varying. This choice is based on the fact that not all instruments are capable of producing all frequencies, and even if they are, the resulting sound may differ depending on the frequency [9]. Moreover, the plucked string instrument possesses a distinct pitch and a decay that aligns well with our objectives.

In the context of a plucked string instrument, the decay refers to the duration it takes for the amplitude of a sound to decrease from its peak to zero. For these types of instruments, an AD (Attack, Decay) envelope is commonly used, with the attack representing the time it takes for the sound to reach its peak loudness after the key is pressed. In our case, we aim to achieve a near-immediate initial sound (Attack) followed by a relatively quick initial decay that gradually fades away to silence. This envelope is visually represented in Figure 3.5.

In order to achieve rapid response times, our approach does not involve waiting for the next note, but instead focuses on accumulating the amplitudes of all notes simultaneously. However, it is crucial to consider that when accumulating the amplitudes of numerous notes, there is a potential risk of audio clipping or excessive loudness, as demonstrated in the lower audio visualization in Figure 3.6.

To address this issue, we incorporate a dynamically scaled amplitude and decay. Specifically, the amplitude is adjusted based on the total cumulative loudness of all active notes, ensuring that it remains within an acceptable range. Additionally, the decay is proportionally reduced based on the number of active notes being played. By dynamically scaling the amplitude and decay, we maintain a balanced and pleasant auditory experience while mitigating the risk of audio clipping and maintaining clarity in the sound output. The result of this approach is depicted in the upper audio visualization in Figure 3.6.

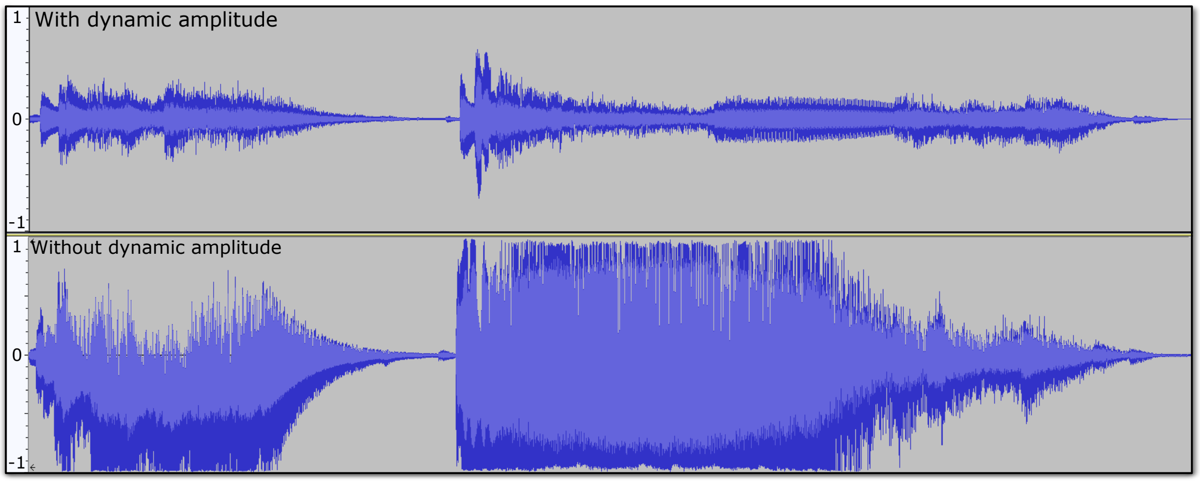


Figure 3.6: Audio visualization of the sound generated by a predefined mouse paths, with both visualizations indicating the amplitude of the audio. The top row represents the audio with dynamic amplitude scaling applied, while the bottom row shows the audio without any scaling.

Considering that each note has a unique amplitude, which is exponentially mapped to its importance, the sound produced by a note n is defined as:

$$\text{note}_n(t) = a \cdot \text{pluck}(t) \tag{3.1}$$

Here, $\text{pluck}(t) \in [-1.0, 1.0]$ represents the unique wave equation (e.g., $\sin(t)$), a is the amplitude for a specific $\text{note}_n(t)$, and t represents time. The amplitude a is defined within the range [0.0, 1.0] and is derived from the importance of the line.

The total cumulative amplitude can be described as the sum of the absolute values of all active notes:

$$c = \sum_n |\text{note}_n(t)| \tag{3.2}$$

The final audio output produced by a set of active notes $N = \{\text{note}_1(t), \text{note}_2(t), ..., \text{note}_n(t)\}$ is defined as:

$$\text{out}(t) = \sum_n \text{note}_n(t) \cdot \frac{m}{c} \tag{3.3}$$

where c represents the cumulative amplitude defined in Equation 3.2, and $m \in [0.0, 1.0]$ refers to the maximum volume or loudness. In implementation it is recommended for m to be less than 1.0 to avoid clipping. In practice, this approach ensures that when playing loud (important) lines rapidly, they do not become excessively loud, while playing less important lines still results in a relatively loud sound. This maintains a balanced audio output that accurately reflects the relative importance and density of the data.

On the other hand, the dynamic decay ensures that transitions between different loudness and frequency levels occur more quickly when multiple notes are played, making the output more responsive to interactive changes. Each amplitude a is reduced proportionally to the number of active notes in N. In practice, this means that as more notes are played, the decay happens more rapidly. Consequently, the audio feedback generated by the system emphasizes the first note slightly louder and extends the duration of the last note slightly longer [9]. This concept is further illustrated in Figure 3.7.

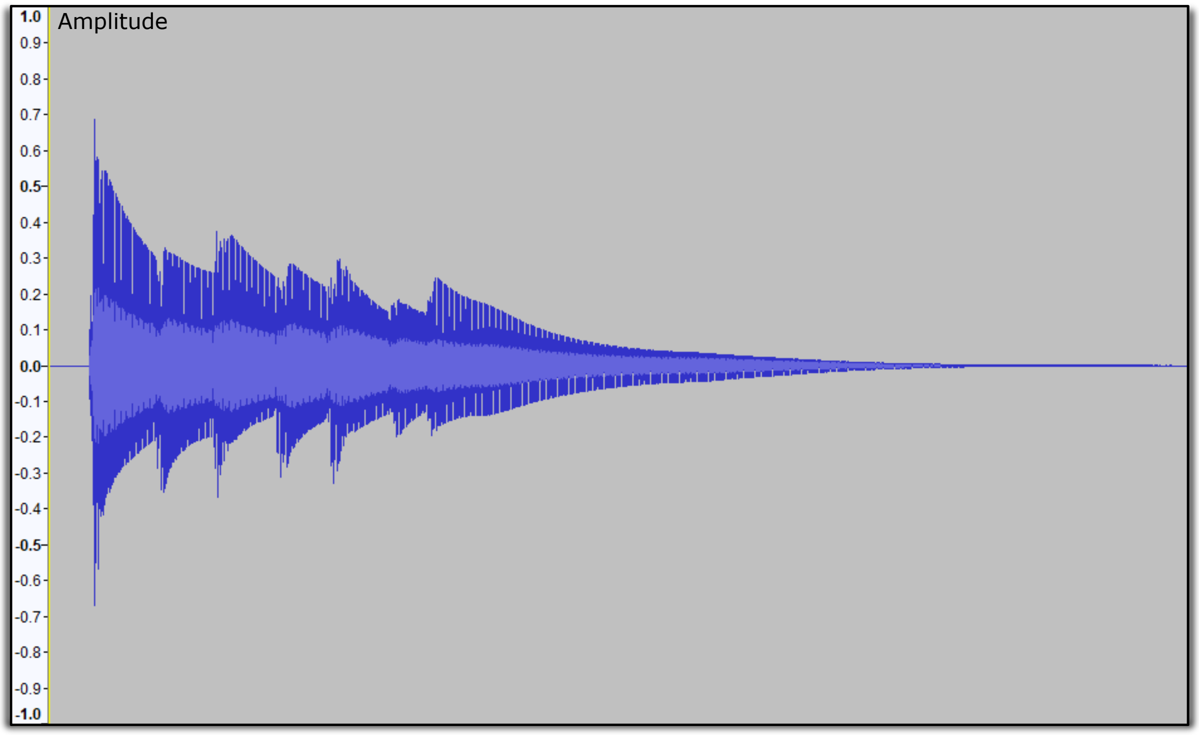


Figure 3.7: Audio feedback produced from evenly spaced lines played in succession, illustrating the accentuation of the initial note and the slight extension of the last note.

3.4 Interactive Tools

By default, our system "plucks" all line segments intersecting the current location whenever the mouse cursor moves, generating notes according to the importance values at the point of intersection. However, we also offer additional interactive tools that allow for more user control and interactivity. In this section, we will explore these tools and discuss how our auditory and visual components interact with each other. We will also examine the conceptual foundations of each tool and its benefits.

To begin, we must consider the need for additional interactive tools. Interaction has been viewed as an essential aspect of data visualization, both for managing increasingly large data sets [27] and for enhancing human-driven exploration [26]. These ideas also extend to sonified data visualization, particularly with model-based sonification approaches, as detailed in Section 2.2.2. However, sonified data visualization can be complex and often requires training for optimal results. Our interactive tools provide additional means of interacting with the data visualization, beyond mouse-line interactions, that reduce the overall complexity.

Our approach includes two distinct interactive tools that are controlled by different key bindings: an interactive lens and line selection capabilities. Although other techniques may be suitable for our data types, we found that they were most effective for our objectives. The interactive lens enables users to zoom in on specific regions of interest, while the line selection feature allows users to select individual line segments for further analysis. These tools are useful for our sonified data visualization system as they provide an additional layer of control and interactivity for users.

3.4.1 Interactive Sonic Lens

In many cases sonification can be overwhelming for various reasons, such as lack of training or prior music knowledge. This is especially the case for dense line charts where a lot of different sounds are produced. We therefore propose a lens that can potentially reduce this complexity by limiting the number of sonified lines. For dense line charts and specifically parallel coordinates, overplotting is also a potential issue [36]. We address this issue with the use of spatial distortion based on the importance of lines. It also is useful for investigating regions where multiple clusters intersect. We therefore offer the capability of allowing users to reveal lines with lower importance.

The lens can be toggled on/off and is placed at the mouse position. Additionally, the user can adjust the radius and modify the importance threshold. Visually, we employ a variation of the distortion lens approach by Carpendale et al. [21] to displace lines (see Figure 3.8) with importance values above a certain threshold, while sonically, this amounts to only playing notes for lines below the threshold. Furthermore, we also provide a lens playback feature, which quickly sonifies every line in the lens in importance order. This provides a quick and easy way to gain an overview of all lines within the lens radius. Both the lens and the lens playback is further demonstrate in Figure 5.5.

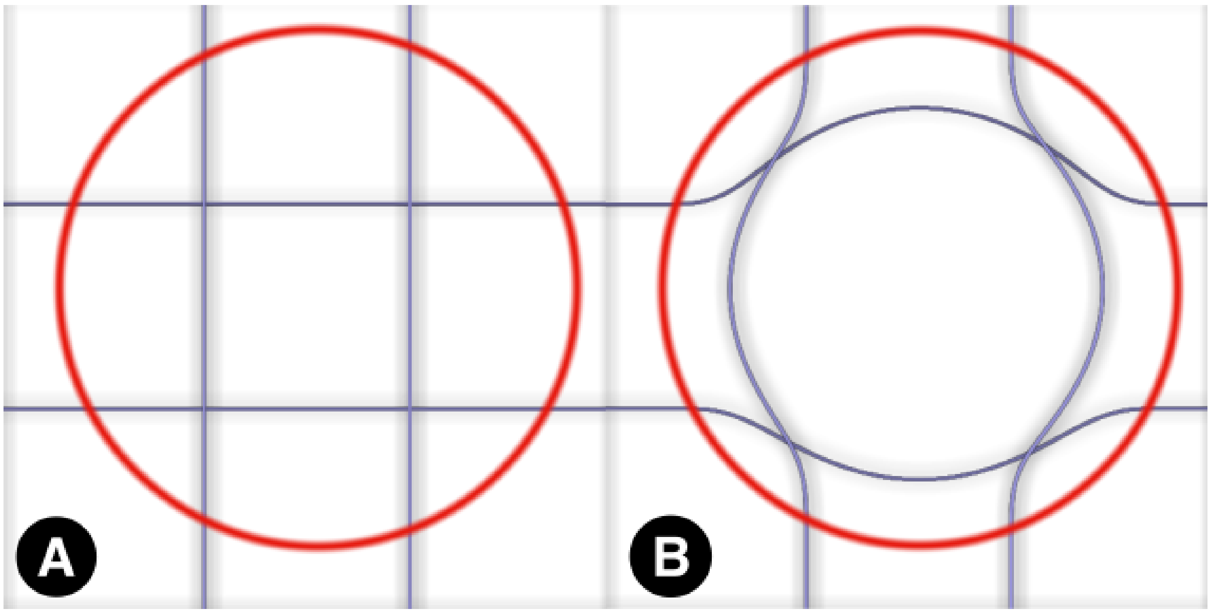


Figure 3.8: Showcase of visual displacement, where (A) is without displacement, and B) is with full displacement.

3.4.2 Auditory Selections

When it comes to Shneiderman's information-seeking mantra [70], the lens provides the zoom and filter, while the selection offers the details-on-demand part. In dense datasets, visualizing every detail can be impractical. Therefore, our auditory selection method offers a way to sonify data based on the users requests. While selecting individual lines may be appropriate for providing details on specific data points, a similarity based selection approach is better suited for dense line charts, particularly parallel coordinates [36].

Our selection approach is based on similarity measures, more specifically, line similarity, which is a real-valued function that measures the similarity between two lines. There exist several line similarity measures with varying purposes and performance [61], and in our approach, we include several types outlined by Writz and Paulus [61]. We also incorporate a single line selection approach, which is useful for providing details for individual lines. We enable line selection through mouse click, and selected lines are highlighted in a different color as seen by the red lines in Figure 3.9. For all types of similarity measures, a range can be selected that determines how similar a line must be to the original to be considered selected and highlighted. The intensity of redness indicate their similarity.

The sonification can be toggled to a "similar mode," which utilizes similarity as the amplitude, instead of the original importance. This means that the actual selection can be further analyzed without other non-similar lines interfering. This same concept

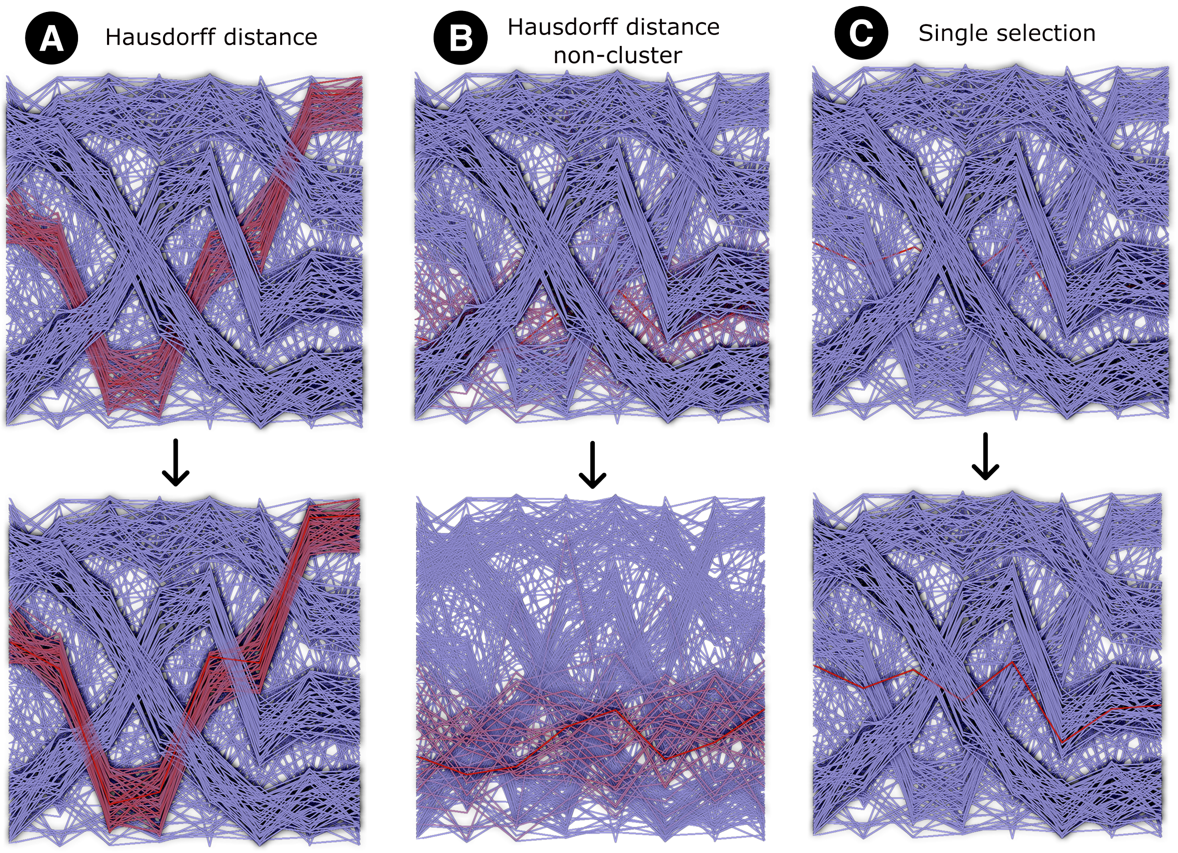


Figure 3.9: Various selection methods with different selected lines. We see in the bottom row how the visualization changes when the similarly decided which lines are rendered on top of each other. In A) and C) the non-selected lines remain constant, however in B) the non-selected lines are deprioritized.

also applies to all the lens functions which depend on importance. The visuals depend on importance to determine which lines are rendered on top of each other. With the selections this can be changed to instead depend on the line similarly measure, which means that the most similar lines are rendered on top of the other lines. This concept is further demonstrated in the bottom row of Figure 3.9, where the selected lines, highlighted in red, are rendered on top of the other lines. Ultimately, this results in a dynamic selection method that supports cluster selection and provides details-on-demand.

3.5 Auditory-Visual Interactions

While human perception of audio is quite robust, we often associate audio with some form of visual stimulus. Auditory-visual interactions refer to how audio and visuals interact and how we perceive them together. Therefore, it is crucial for sonification to match or provide some visual indication of the sound produced. The "McGurk Effect" is a phenomenon that occurs when visual and auditory speech tokens are mismatched but presented simultaneously [55]. This effect can "trick" the brain into hearing a different sound depending on the visuals presented (see [3] for a video demonstration). As such, it has been found to be advantageous for sonification to be accompanied by appropriate visuals [57]. To tackle this issue, we visually indicate the lines played using a vibration effect to reinforce the musical instrument analogy. The played lines can optionally be highlighted using color.

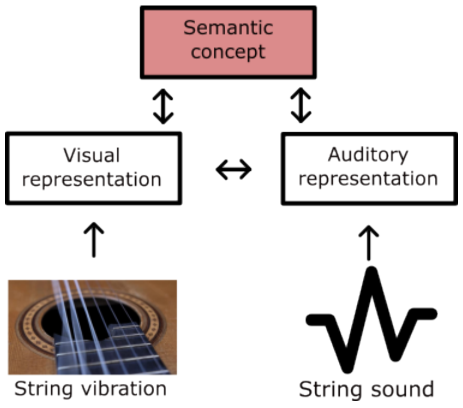


Figure 3.10: Possible processing route whereby activation from auditory (e.g., string sound) or visual (e.g., visual vibrations) stimuli could strengthening the salience of the visual object.

In fact, most of our sonifications are accompanied by some visual animations to varying degrees. The primary animation being the vibration effect when plucking strings, which is essentially a direct mapping of the sound to the line "vibration." This ensures that the user is aware of which lines they have actually played and which lines are currently contributing to the soundscape. Furthermore, in cases where the vibration effect might be obscured, the user has the option to highlight the lines that produce sound.

Chapter 4

Implementation

We developed our Line Harp approach using OpenGL [35], while the sonification was implemented with Gamma [64]. In this chapter, we will elaborate on the implementation process and demonstrate how we utilized the OpenGL and Gamma Application Programming Interfaces (API). Firstly, we will provide an overview of the project, building upon the source code from Line Weaver [76]. Secondly, we will delve into the additional visual implementations that were incorporated. Finally, we will elaborate on the integration of sonification with the visuals.

We built our application on top of the existing source code from Line Weaver [76]. This allowed us to focus on the implementation of the audio without needing to recreate the entire visualization from scratch. However, we also needed to make additional visual changes to support our sonification objectives. While we will briefly introduce the original Line Weaver source code in the following sections, our primary focus will be on the new audio/visual implementations.

Line Weaver [76] source code is available at:
`https://github.com/TTrautner/LineWeaver`

Our source code is available at:
`https://github.com/Egglis/LineHarp`

4.1 Chosen Technologies

Our application was implemented using C++ and OpenGL [35]. The Graphical User Interface (GUI) was implemented using ImGui [23]. The sonification was implemented with the use of the Gamma library, which is a generic synthesis library for C++ [64]. We chose this library as it provides some flexibility for audio while also being a high-level library. Gamma is oriented towards real-time sound and graphics applications, which is important for producing responsive interactive audio. The Gamma library also provides a set of tools and functions for working with digital audio signals, including filters, oscillators, envelopes, and more. However, we decided to utilize the already provided `Pluck` class, which provides a synthesised plucked string sound based on frequency. This meant that we did not have to implmented our own synthesised string. However, the `Pluck` class was encapsulated within a `Note` class, which provided us with control over the key parameters of amplitude and frequency.

4.2 Visuals

In this section, we will delve into the details of our visual implementations. To enable line displacements, we integrated a tessellation stage into the rendering pipeline (see Figure 4.1). This addition facilitates the necessary deformations and enhances the visual quality of the lines. Next, we will discuss the visual implementation of our interactive tools. Furthermore, we will explore the implementation of auditory-visual interactions, as conceptually outlined in Section 3.5.

4.2.1 Overview

In our implementation loading as well as pre-processing the data are performed on the Central processing unit (CPU) while the rendering itself runs in parallel on the Graphics Processing Unit (GPU). Selection of detests to be visualized as well as additional user-dependent parameters, such as line color or width, can be adjusted with the GUI.

Line rasterization: The input of our rendering pipeline is three buffers containing the x and y coordinates, and corresponding importance. Ordering of lines within the buffer is unnecessary as they are sorted in the blending phase [76]. In the fragment shader, we utilized a Signed Distance Function (SDF) in order to determine the fragments covered by the line segment and evaluate their colors. The result, together with its importance value, is then added to a per-pixel linked list. We used an image object to store the index of the last list entry for each pixel, as well as a Shader Storage Buffer Object (SSBO) to store the fragment data. The fragment data includes line directions and identification in order to facilitate our directional frequency mapping and enable line mouse interactions.

Fragment blending: We used a screen-filling quad to traverse the linked list for each pixel, performing blending based on their importance-driven blending [76]. We also implemented halos, if enabled, based on the method of Luft et al [52].

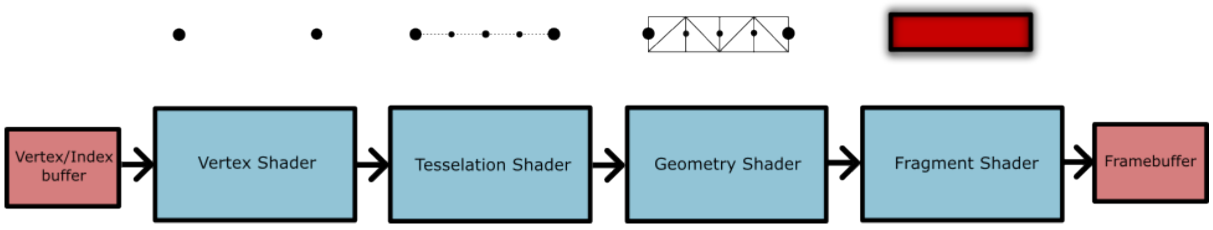


Figure 4.1: Simplification of the rendering pipeline.

4.2.2 Tessellation Stage

To achieve the desired bending or distortion effect for line displacements, we introduced an additional tessellation stage in the shading pipeline. This implementation effectively supports the required line deformations. Traditionally, a line spanning two data points is represented by two triangles forming a triangle strip. However, in our approach, we needed the lines to be bendable as well. To accomplish this, we employed linear interpolation to interpolate additional vertices, allowing for bendable lines.

While this interpolation could be performed as a pre-processing step on the CPU, OpenGL offers a tessellation shader that can handle this task efficiently. The tessellation shader also has the advantages of requiring fewer executions of the vertex shader and generally less CPU draw calls. Additionally, performing the computation for per-vertex lens displacements within the shader pipeline offered the flexibility to dynamically adjust the number of vertices representing a single line segment. This dynamic adjustment enabled us to finely control the level of detail and smoothness of the line deformations.

The tessellation shader is an optional stage in the OpenGL rendering pipeline where patches of vertex data are subdivided into smaller primitives [34]. This stage consists of two components: the Tessellation Control Shader (TCS), which determines the number of subdivisions, and the Tessellation Evaluation Shader (TES), which computes new vertex values and performs vertex displacement.

For our implementation, we utilized the OpenGL primitive type `isolines` to generate Nsubdivisions for each line segment. The TES provides a interpolation value $t = [0, 1]$, which allowed us to use the GLSL function `mix(a, b, t)` for linear interpolation between vertex values `a` and `b`. This interpolation was essential for achieving the bending effect of the lines. The level of detail (LOD) or the number of subdivisions N was computed based on whether any vertex contributing to a line fell within the lens circle. This calculation determined the appropriate LOD for each line segment, ensuring that the line deformations were accurately represented.

To render thick continuous lines, we utilize triangle strips that maintain connectivity even for bent lines and acute angles. The generation of triangle strips are preformed in the TES stage since the necessary neighborhood information was only available at this stage of the rendering pipeline. Within each iteration of the TES, we perform interpolation for both the previous and next vertices, which is essential for constructing a miter joint. A miter joint is formed by cutting each of the two parts to be joined across the main surface, and it can comprise any angle greater than 0 degrees. This technique enables us to create smooth and visually pleasing transitions at the corners of the lines. Figure 4.2 provides a visual demonstration of how the miter joints are constructed.

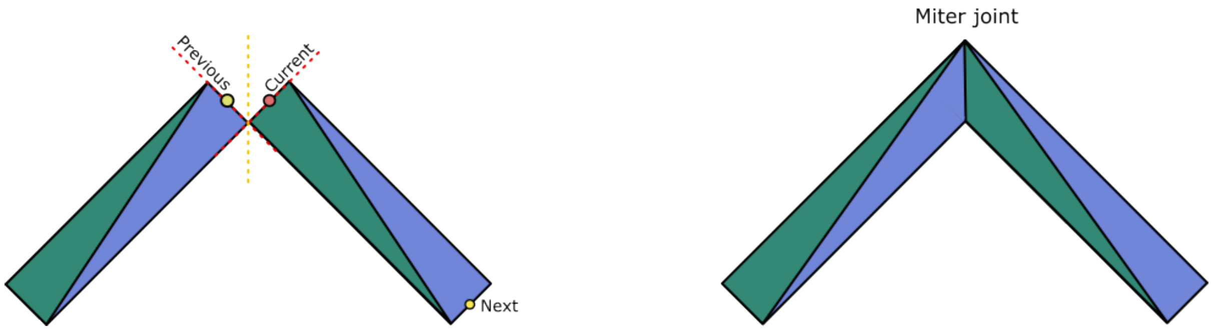


Figure 4.2: Demonstration of how previous and next vertex information is used to construct a miter joint.

The final stage of the tessellation process involves implementing lens distortion as a per-vertex displacement operation. This displacement is calculated by considering the relative positions of each vertex and the lens. By implementing this in the tessellation stage, we can accurately adjust the LOD of the lines based on the magnitude of the displacement. This ensures that the visual representation of the lines is appropriately adjusted to reflect the lens distortion effect.

The displacement vector is computed as a product of a direction vector $\vec{d}$ and a weight scalar w. The direction vector is determined by the relative positions of the vertex and the lens. The weight scalar w is defined as:

$$w = 1 - \text{smoothstep}(0, r, d) \tag{4.1}$$

Where r is the radius of the lens and d is the distance between the lens and vertex position and is used to determine the magnitude of the displacement vector. The smoothstep function, which is a Hermite interpolation function available in OpenGL Shading Language (GLSL) [34], provides a smooth transition between 0 and 1 based on the relative distance between the lens and the vertex positions. This approach is effective in avoiding visual artifacts that can arise from sudden changes in displacement magnitude, while simultaneously adhering to the physical constraints of the lens geometry. Furthermore, the use of the smoothstep function in the weight formula ensures that the displacement operation is computationally efficient and suitable for real-time graphics applications.

4.2.3 Selections

The selection of lines is achieved by utilizing a framebuffer-sized texture image that contains line identification. However, a detailed explanation of this process is provided in Section 4.3.1. Therefore, the focus of this subsection is on the implementation how our similarity functions.

We incorporate several types of similarity measures as outlined by Writz and Paulus [61]. With the exception of importance and single selection, we calculate similarity matrices that contains the similarity between all combinations of line pairs. This is demonstrated in Figure 4.3, where a corresponding similarity matrix is visualized. The computation of the similarity matrix is performed as a pre-processing step during dataset loading due to its computational complexity. To optimize loading time, we have implemented caching of similarity matrices to improve efficiency.

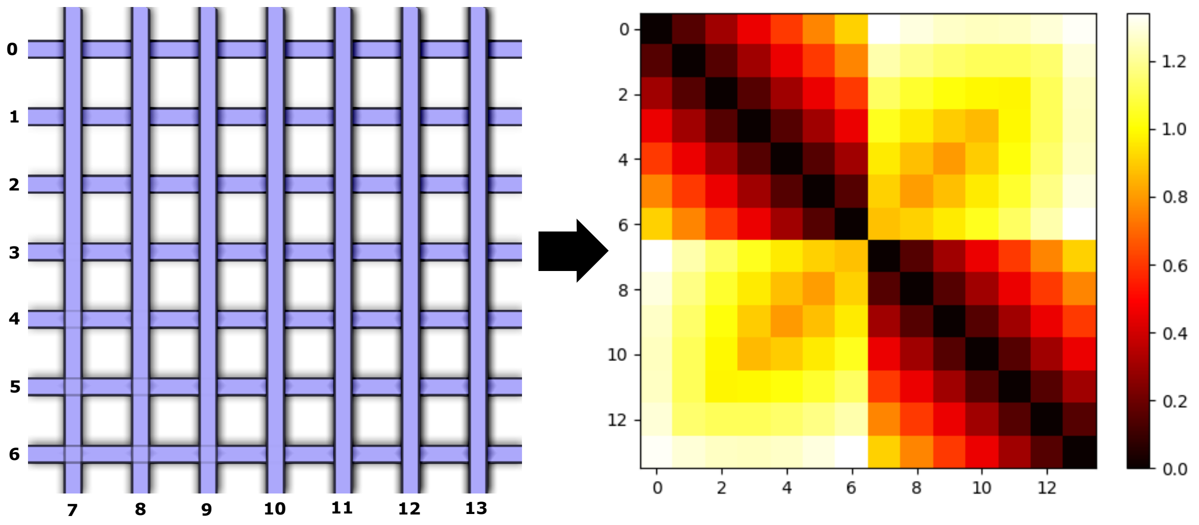


Figure 4.3: A grid dataset and a visualization of its similarity matrix. The matrix visualization is of the *Hausdorff* [5] similarity measure.

4.2.4 Auditory-Visual Interactions

In our approach, we have incorporated auditory-visual interactions to emulate the physical vibrations of stringed musical instruments. Further details on this can be found in Section 3.5. Simulating the physics and movement of a plucked string is a computationally demanding task. To address this, we implemented optimization techniques that aim to mimic the real-world auditory-visual interactions.

For animations or movements with a fixed start and end, we employed an elastic easing function, which provides a smooth and natural transition between the two states. This ensures that the visual representation of the string's movement closely aligns with the expected auditory response. Additionally, for string vibrations resulting from plucking with the mouse, we applied a ”vibration” effect that displaces each line segment in the direction of its normal. This effect creates a visually dynamic representation of the string's oscillation, as depicted in Figure 4.4. These optimization techniques enable us to approximate the complex auditory-visual interactions, without the computational impact.

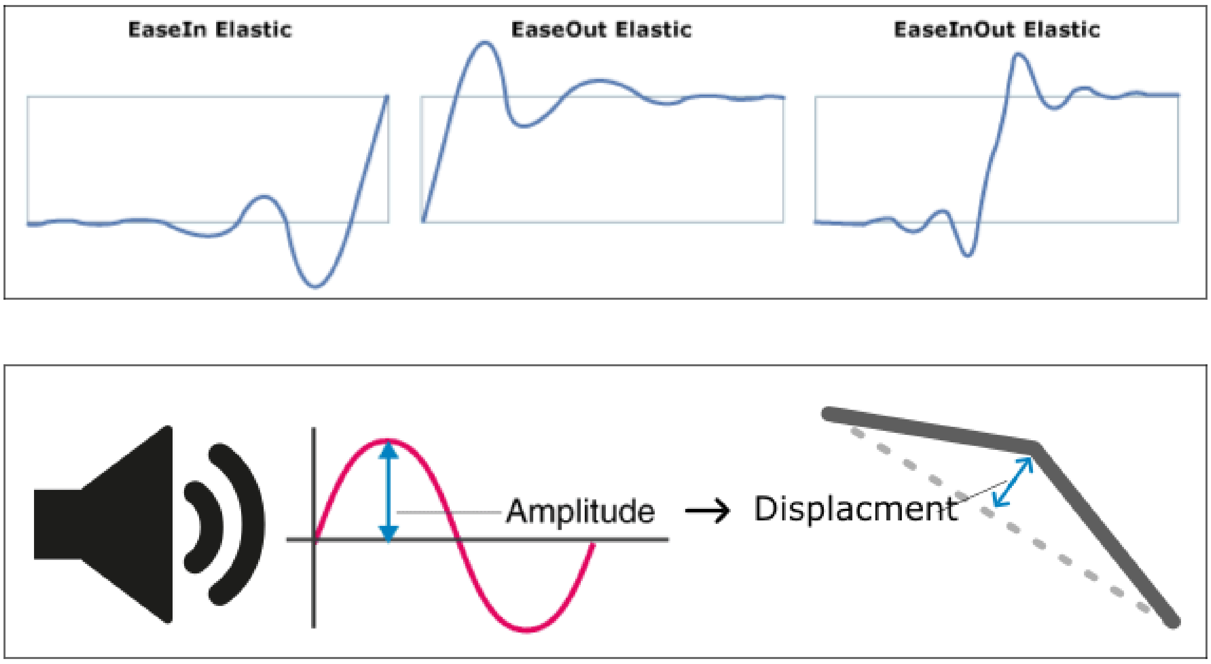


Figure 4.4: Easing functions used for animations and how we implemented the direct mapping of amplitude to "vibration".

4.3 Sonification

As previously mentioned, the audio was implemented with Gamma and requires a separate thread from the visual per their specification [64]. Since our audio is dependent on visual interaction it meant that the main visual thread needed to communicate with the audio thread. This presents the possibility of major synchronization issues and possible corrupted audio. To prevent synchronization issues and enable dynamic amplitude control, we employed a buffer with a producer-consumer semantics (see Figure 4.5). The buffer ensures that data is available when required, allowing the visual and auditory threads to operate independently without interference or data loss. It also simplifies the accumulation of all active notes, which is crucial for our dynamic amplitude. The general functions of the buffer can be summarized as follows:

1. A interaction that requires a audible feedback adds a note to the buffer.
2. The audio thread reads the buffer and accumulates the final audio output.
3. The audio thread removes notes from the buffer that are no longer active.

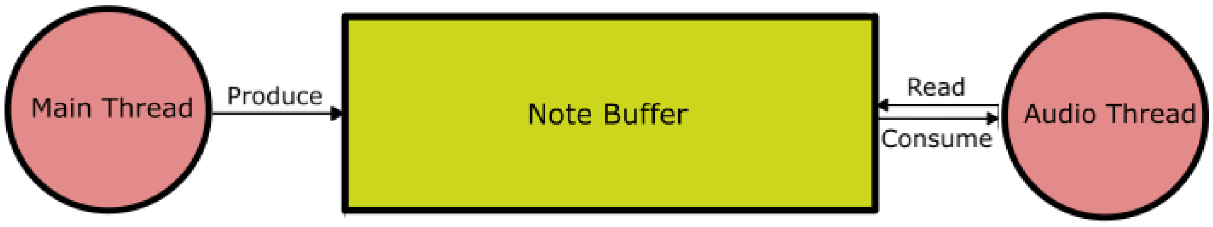


Figure 4.5: Note buffer operations. The main visual thread produces notes through user-generated interactions, while the audio thread both reads the buffer and consumes the notes.

4.3.1 Producing Notes

In the blending phase, outlined in section 4.2.1, we assign a unique identifier to each line. This information is written to a framebuffer-sized texture image that additionally contains x and y directions for the particular line segment. We furthermore utilize a mechanism to read from that texture at a particular pixel, that in OpenGL is performed using `glReadPixel`. With this implementation we can perform mouse lines intersection by reading the pixel at the mouse position and adding the corresponding note to the *note buffer*. `glReadPixel` requires the CPU to wait until the GPU has finished all rending before it can receive any results [34]. However, since we only need to read a single pixel we reduce this performance impact.

Another way in which our sonification system operates is through the use of a note queue system, which differs from the single-note approach outlined earlier. This approach is necessary for the sonification of the lens, as it requires information about each line within the lens to be processed simultaneously. To accomplish this, we implemented an additional SSBO that contains information about each line in the lens. This "lens buffer" is then accessed from the CPU via the OpenGL function `glGetBufferSubData` [34]. Each note in the buffer is pushed into a first-in-first-out (FIFO) queue, which is popped at intervals of n seconds. and added to the *note buffer*. The delay n can be adjusted in the GUI, but by default, it is set to 0.025 seconds. To facilitate synchronization between visuals and audio, the adding of new notes is handled by the main thread, rather than the audio thread. While reading an SSBO can be expensive in terms of performance, its use is limited to key presses, and therefore has negligible impact on the overall performance of the program. Both approaches for handling the audio interactions is demonstrated in Figure 4.6.

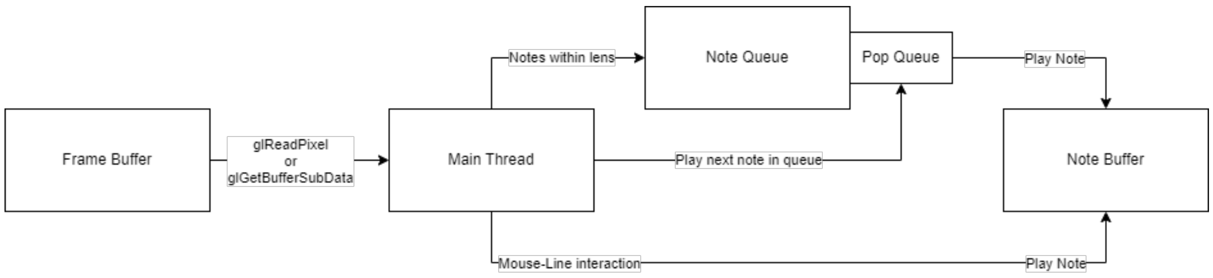


Figure 4.6: Overview of the audio pipeline

4.3.2 Reading Notes

Commonly audio programming involves buffering, which is a separate concept from our *note buffer*, and acts as a temporary storage area for incoming or outgoing audio samples. This buffering is entirely handled by the Gamma library [64], however the specification for the buffer is adjustable. In audio programming, buffers are commonly utilized to process audio in fixed-size chunks or frames, where each frame contains a predetermined number of audio samples. The crucial factors to consider in this context are the buffer size and sample rate.

The buffer size refers to the quantity of audio samples contained within a single buffer, determining the volume of audio data processed or played back at any given time. A larger buffer size enables the processing or playback of a greater number of audio samples in each iteration, reducing the risk of audio glitches or dropouts. However, it is important to note that larger buffer sizes also introduce increased latency between audio input/output and processing. In our implementation, we have opted for a buffer size of 128 samples and a sample rate of 44.1 kHz, a widely adopted standard for audio CDs. This choice ensures a relatively low audio latency without excessively burdening the computational load on the system.

During each iteration of the buffer, the audio thread reads from the note buffer and accumulates the final sound wave. The resulting sound wave is then scaled according to our dynamic amplitude, as defined by Equation 3.3. The final audio wave is represented a floating-point value ranging from -1.0 to 1.0, which is subsequently processed by the Gamma library to generate waveform data that is ultimately passed to the sound hardware.

4.3.3 Pluck Sound Synthesis

The synthesis of the sound wave is performed by the `Pluck` class from the Gamma library, while the management of loudness is handled separately in our custom `Note` class. The composite sound wave comprises the combined contributions of both the `Pluck` class and our `Note` class. By integrating various elements such as noise generation, decay envelope, filtering, and comb filtering, the `Pluck` class offers a convenient and efficient solution for synthesizing plucked string sounds [64]. Our plucked sound synthesis algorithm can be described as follows:

1. Initially, a **noise generator** represented by `noise()` produces a random white noise signal that emulates the plucking action. This noise signal serves as the initial excitation for the plucked string sound.
2. The noise signal is then scaled based on the **decay envelope** , represented by `decay()`, which is a function that returns a floating-point value representing the amplitude at the current time. The expression `noise() * decay()` combines the generated noise signal with the amplitude envelope defined by the decay function. This multiplication scales the noise signal according to the envelope, effectively applying the desired shape to the noise signal. The shape of the decay envelope is further illustrated in Figure 3.5.
3. The resulting signal obtained by multiplying the noise and decay functions is passed through a **filter**. In our case, a Biquad low-pass filter is employed [63], which attenuates higher frequencies while allowing lower frequencies to pass through. The output of the filtering operation, `filter(noise() * decay())`, represents the filtered signal that has been shaped by the low-pass filter.
4. The filtered signal is then fed into a **comb filter**. The comb filter simulates the resonant properties of a vibrating string, emphasizing or dampening certain frequencies to generate the characteristic resonances and harmonics associated with plucked string instruments. The comb filter frequency and decay properties are set during its initialization [63].

The final plucked sound synthesis algorithm can thus be expressed as:

```
comb(filter(noise() * decay()))
```

This formulation represents the unique wave function for a note, referred to as *pluck*() in Equation 3.1. The sound synthesis algorithm is conceptually depicted in Figure 4.7, where a initial noise signal is decayed and filtered to produce a plucked string sound with a frequency of 440 Hz.

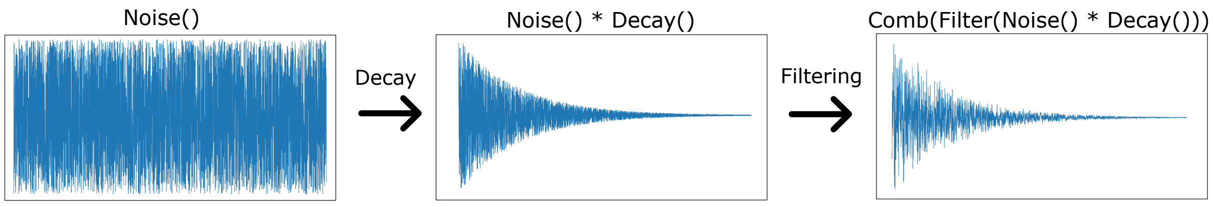


Figure 4.7: Demonstration of how the plucked sound synthesis algorithm transforms a noise signal into a music note with a frequency of 440 Hz

4.3.4 Consuming Notes

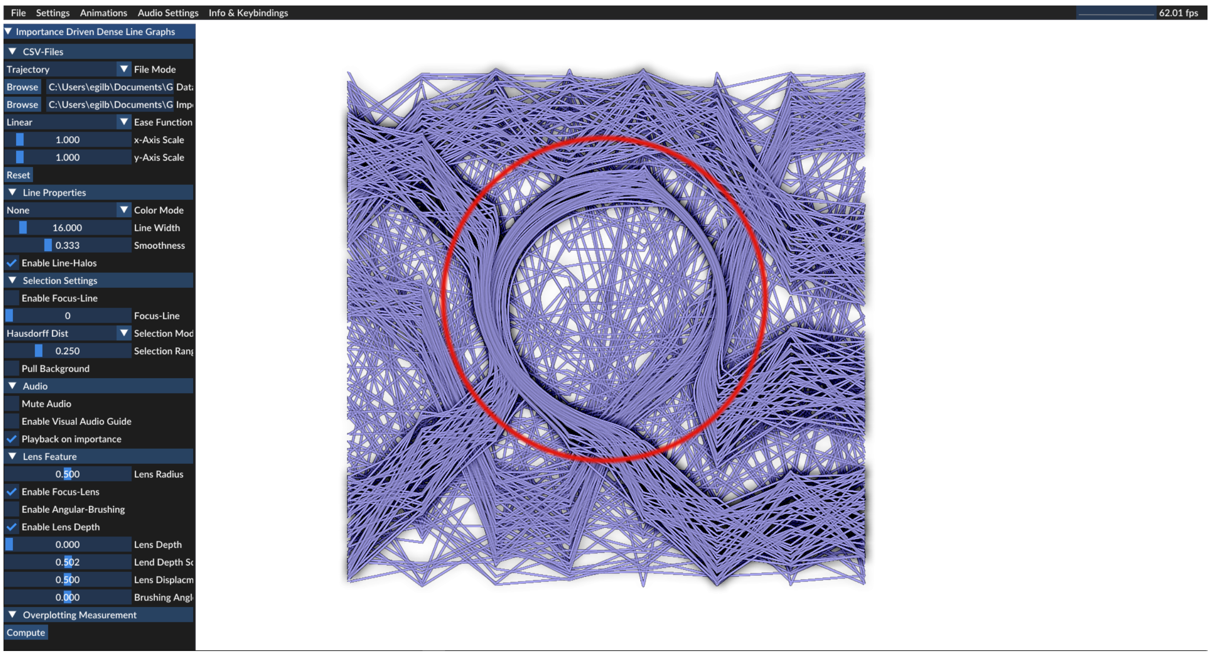
In the case of real-world instruments that vibrate, they eventually lose all energy and cease producing sound. This is simulated with the decay envelope, as illustrated in Figure 4.7, where the amplitude is gradually reduced. However, according to the specifications of the `Decay` in Gamma [63], the decay does not actually reach zero but is considered "complete" when the amplitude falls below 0.001. The decay of the `Pluck` class is static, meaning that once it is initialized, it maintains a consistent reduction of amplitude over time.

In our implementation, we have introduced an additional element called dynamic decay, which adjusts the decay based on the current soundscape. In practice, the amplitude of a note is reduced at each time step, depending on the magnitude of the notes present in the *note buffer*. We implemented this as an extra reduction in amplitude for each iteration of the audio thread. Consequently, when the *note buffer* is read, any notes with an amplitude less than 0 are identified for removal. These marked notes are then safely removed from the *note buffer* after it has been read.

Chapter 5

Results

In this chapter we will present our results. First we will present the performance of our application both visually and audibly. Finally, we will demonstrate the strengths and versatility of Line Harp based on usage examples.



```
Trajectory                  File Mode
C:\Users\egilb\Documents\G  Data
C:\Users\egilb\Documents\G  Imp
Linear                      Ease Function
1.000                       x-Axis Scale
1.000                       y-Axis Scale
None                        Color Mode
16.000                      Line Width
0.333                       Smoothness
✓ Enable Line-Halos
  Enable Focus-Line
0                           Focus-Line
Hausdorff Dist              Selection Mod
0.250                       Selection Rang
  Pull Background
  Mute Audio
  Enable Visual Audio Guide
✓ Playback on importance
0.500                       Lens Radius
✓ Enable Focus-Lens
  Enable Angular-Brushing
✓ Enable Lens Depth
0.000                       Lens Depth
0.502                       Lend Depth Sc
0.500                       Lens Displacm
0.000                       Brushing Angl
```

Figure 5.1: Screenshot of the GUI, settings correspond to those used when measuring FPS in Table 5.1.

5.1 Performance

Considering that our application is dependent user interactions, it is important the the application runs with interactive frame rates which we consider to be at least 20 Frames per Second (FPS). The visualization can depict different amounts of lines and vertices. The FPS is also heavily dependent on the type of interactions and we therefore measure both a static FPS and a interactive FPS. The interactive FPS is measured based on a pre determined movement, while the static FPS is measured without any user inputs. All interactive FPS measures are performed on equal settings, since displacement magnitude, line width and lens radius impact the overall performance. The tests was run on computer connected to a screen with resolution 1920x1080. The computer was running Windows 10 as operating system, with an Intel(R) Core(TM) i5-12600K CPU and a NVIDIA GeForce GTX 1070 graphics card. The results for this setup were as follows:

Dataset - Figure	**Lines**	**Static FPS**	**Interactive FPS**
`4C.6-150N` - 5.4	350	$107.5_{106.2}^{108.9}$	$96.4_{90.0}^{100.8}$
`1C.4-out` - 5.6	49	$411.2_{385.2}^{432.0}$	$394.8_{375.3}^{415.1}$
`3C.6-out` - 5.7	144	$144.7_{140.6}^{147.1}$	$132.6_{126.2}^{139.7}$
`AndrewsPlot` - 3.1	1000	$19.4_{16.3}^{22.2}$	$13.3_{10.3}^{17.4}$
`IrisData` - 5.8	150	$180.9_{179.3}^{183.8}$	$172.4_{160.5}^{179.9}$
`PlainWeave` - 4.3	14	$665.4_{610.4}^{772.6}$	$640.5_{574.0}^{731.2}$

Table 5.1: Summary table of performance for varying datasets, including figure numbers, the number of lines in the dataset, and the average FPS (avg_{min}^{max}) for static and interactive measurements.

The performance measurements presented in Table 5.1 indicate that, in most scenarios, the implementation achieves interactive frame rates. However, when the number of lines becomes excessive, the performance starts to impact the user experience. For larger datasets, simple adjustments can be made to achieve better frame rates. For example by adjusting the line width, which is set to a default value of 16.0, interactive frame rates can be achieved even for larger datasets. However, it should be noted that this comes at the cost of reduced readability of individual lines.

5.2 Audio Performance

Another important aspect for our implementation is the audio performance. There are numerous different aspects that can impact the audio performance, for example audio quality, latency, dynamic-range and noise. Audio performance can also be affected by several factors that do not depend on our implementation, such as the acoustic environment of the user, the quality of audio equipment, and the auditory perception capabilities of users. Therefore, we will only focus on audio performance measures that falls within the scope of our implementation.

Audio latency commonly refers to the delay or lag between the generation of an audio signal and its playback. In our case, we define audio latency as the measured time between an interaction and the corresponding audio output. This approach allows us to specifically isolate and assess the latency within our implementation, thereby bypassing the latency introduced by the input device and the audio device. Lower latency is highly desirable, particularly in real-time applications, as it ensures that audio events are perceived without noticeable delays. Additionally, minimizing audio latency is crucial for maintaining synchronization between our visual elements and their accompanying audio feedback.

In our application, we measure audio latency across the entire interaction-audio pipeline, which considers the combined processing time of both the interaction pipeline, specifically the `glReadPixel` operation, and the audio processing. Most of our auditory actions are controlled by the visual main thread, making them dependent on the frame rate (FPS). However, audio processing, which involves note accumulation, dynamic amplitude scaling, and the removal of finished notes, relies entirely on the performance of the audio thread. Utilizing our hardware, we observed an average audio processing speed of under 2 ms across all densities and playback speeds. The audio processing time is solely determined by the number of notes in the *note buffer* and, due to our dynamic decay, it consistently remains relatively short. These results indicate that, in the majority of cases, audio processing time will have a negligible impact on the overall audio latency.

On the other hand, we measured the total audio latency using our hardware and found an average latency of 9.1 ms, with a maximum of 19 ms for mouse line interactions. This result highlights that the interaction pipeline is the primary contributor to audio latency. Nonetheless, the audio latency for mouse line interactions still remains relatively low. As detailed in section 4.3, we also employed a alternative implementation for providing audio feedback for the lens. This audio latency was measured from key press to audio output, yielding an average latency of 10.0 ms, with a maximum of 18 ms using our hardware. These findings suggest that the audio interactions in our application meet the timing requirements for real-time visual interactions of 50-100 ms, as established by Tominski et al. [74]. Overall, our measurements indicate that the audio latency in our implementation falls within acceptable limits, providing users with a responsive and immersive audio-visual experience.

5.3 Audio Visualizations

To visualize the audio generated by our method, we utilized the recording software Audacity [1]. Our audio visualizations encompass both amplitude (waveform) and frequency visualizations (spectrogram). The amplitude is plotted along the y-axis, while the x-axis represents time. The amplitude is represented by a numerical value ranging from -1.0 to 1.0 and centered on zero. The dark blue waveform displays the tallest peak in the area, meanwhile the light blue part displays the average RMS (Root Mean Square) value of the same group (see Figure 5.2). Although, Audacity support decibel (db) visualizations the numerical approach more accurately reflects our representation of amplitude in our implementation [2].

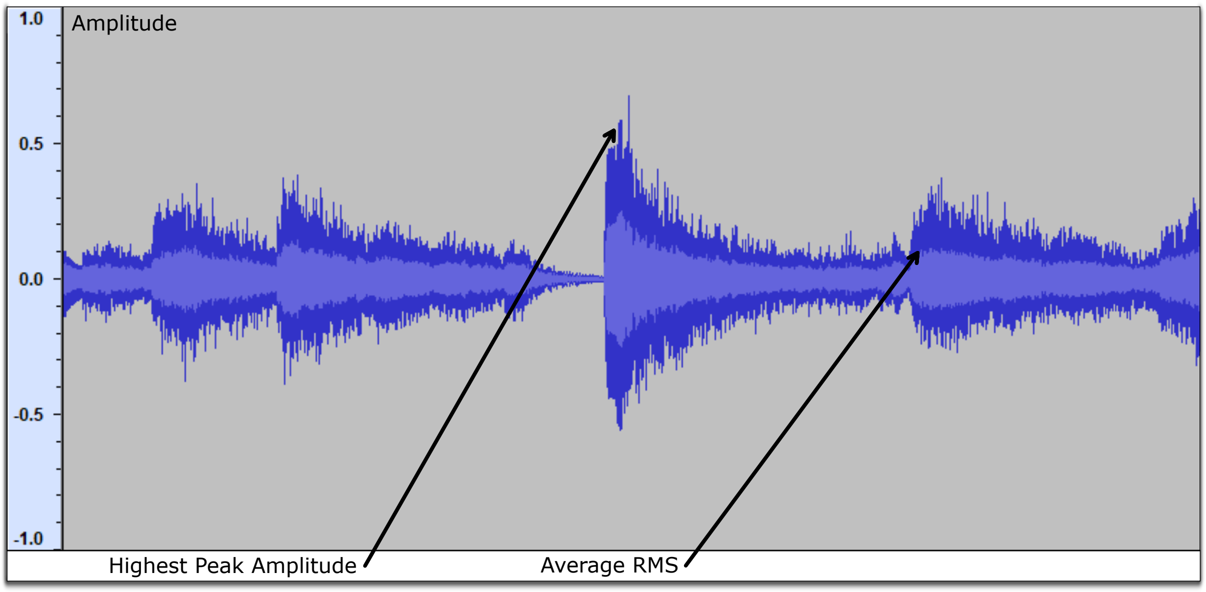


Figure 5.2: Demonstration of what the two shades of blue represents. The dark blue is the peak amplitude, meanwhile the light blue is the average RMS.

The frequency visualization is a spectrogram that highlights the contour of the fundamental frequency (musical pitch) of the audio, utilizing the Enhanced Autocorrelation (EAC) algorithm [1]. This algorithm provides a mathematical representation of pitch changes in an audio piece and is the recommended method for visualizing note pitch. The frequency axis in the spectrogram is displayed on a logarithmic scale, as recommended by Audacity [1], aligning with the logarithmic nature of the SPN pitch scale.

The fundamental frequency, or pitch, characterizes the perceived frequency of the sound. While the audio visualizations include frequencies above and below the pitch, it is the pitch that defines the perceptual tone of the sound. For instance, in the audio visualization depicted in Figure 5.3, the most intense regions indicate the pitch, while the less intense areas represent partial tones that are whole multiples of the pitch, called harmonics. This concept of fundamental frequency (pitch) and harmonics is further illustrated in Figure 3.2.

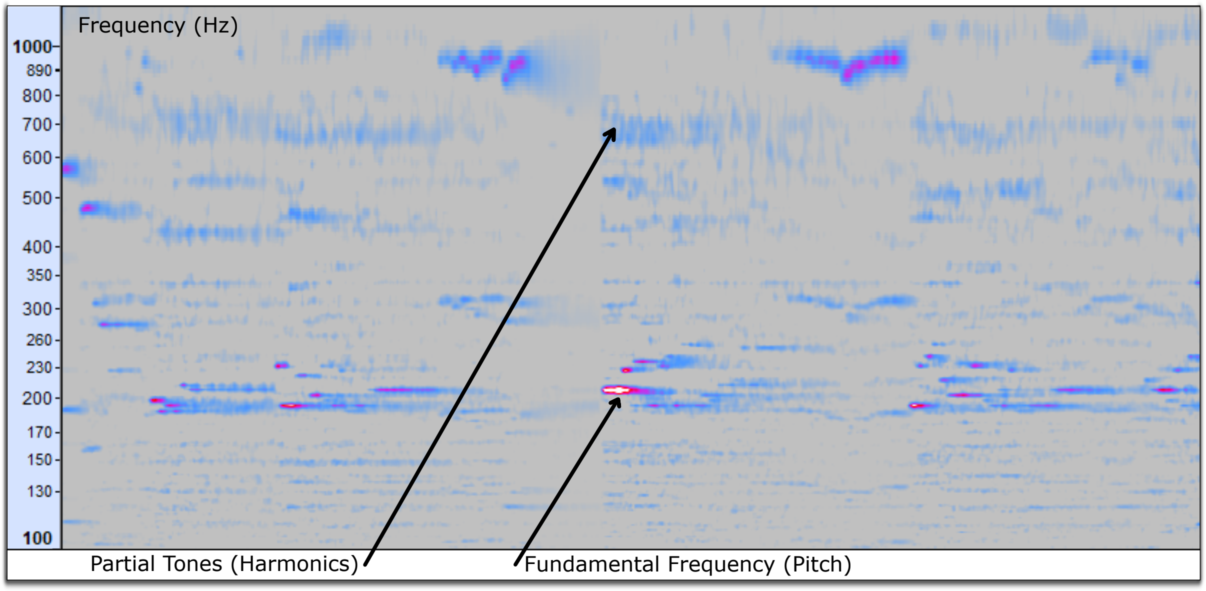


Figure 5.3: Demonstration of the frequency spectogram. The most intense sections indicate the pitch, meanwhile the less intense areas represents its partial tones.

Since our audio feedback is interaction-dependent, we utilized a consistent method of recording the audio feedback. This means a fixed programmed mouse paths that also has a fixed duration. The path is visually represented with a red line as seen on the top in Figure 5.4. The audio produced by the mouse following this path is then visualized in the corresponding audio visualizations. Most of the following usage example follows this mouse-path audio visualizations format, however we also recommend watching the supplementary videos showcasing both the audio and the visuals for the examples. The videos can be found on the figures and as references in the bibliography.

5.4 Usage Examples

In this section we present several usage examples created with our implementation of Line Harp. Each example is accompanied by audio visualizations as outlined in Section 5.3 and video demonstrations [13, 14, 17, 18, 15, 16]. The usage examples demonstrates common data analysis tasks, lens functionality and analysis of a real-world dataset. All of our synthetic datasets are derived from Blumenschein et al. [6], which provided a synthetic benchmark dataset for clustering analysis. The real-world dataset is based on the Iris dataset [30].

5.4.1 Clusters

Identifying clusters in dense line charts and parallel coordinates is a common task in data analysis, and understanding the trends and features within these clusters is equally vital in many cases. In this context, we argue that our sonification approach can enhance the perception of clusters. To demonstrate this, we utilized an example dataset that comprises four distinct clusters, each exhibiting varying features, as depicted in Figure 5.4. This synthetic dataset [6] also includes 150 random lines (depicted in dark green), which are of lower importance compared to the clustered lines.

With our sonification approach, the speed of mouse movement plays a crucial role in determining the level of detail conveyed through audio feedback. Faster mouse movements result in a more generalized overview of the clusters, while slower movements provide more detailed and nuanced audio representations. In Figure 5.4, we employed a fixed mouse movement speed for a duration of 5 seconds, represented by the red line. This is a relatively slow mouse movement and provides adequate details. This video [13] demonstrates pseudo random mouse movements and how they impact the audio feedback.

In Figure 5.4 we have visualized both the amplitude and frequency of the audio feedback produced by the highlighted red line. Each cluster is marked in the graph and the audio visualization, additionally the duration of each cluster is further depicted in the timeline. A video demonstration of this sonification can be seen in [14]. Observing the audio visualization in Figure 5.4, we can discern that whenever the mouse transitions from one cluster to another, there is a noticeable spike in amplitude, accompanied by a corresponding change in frequency.

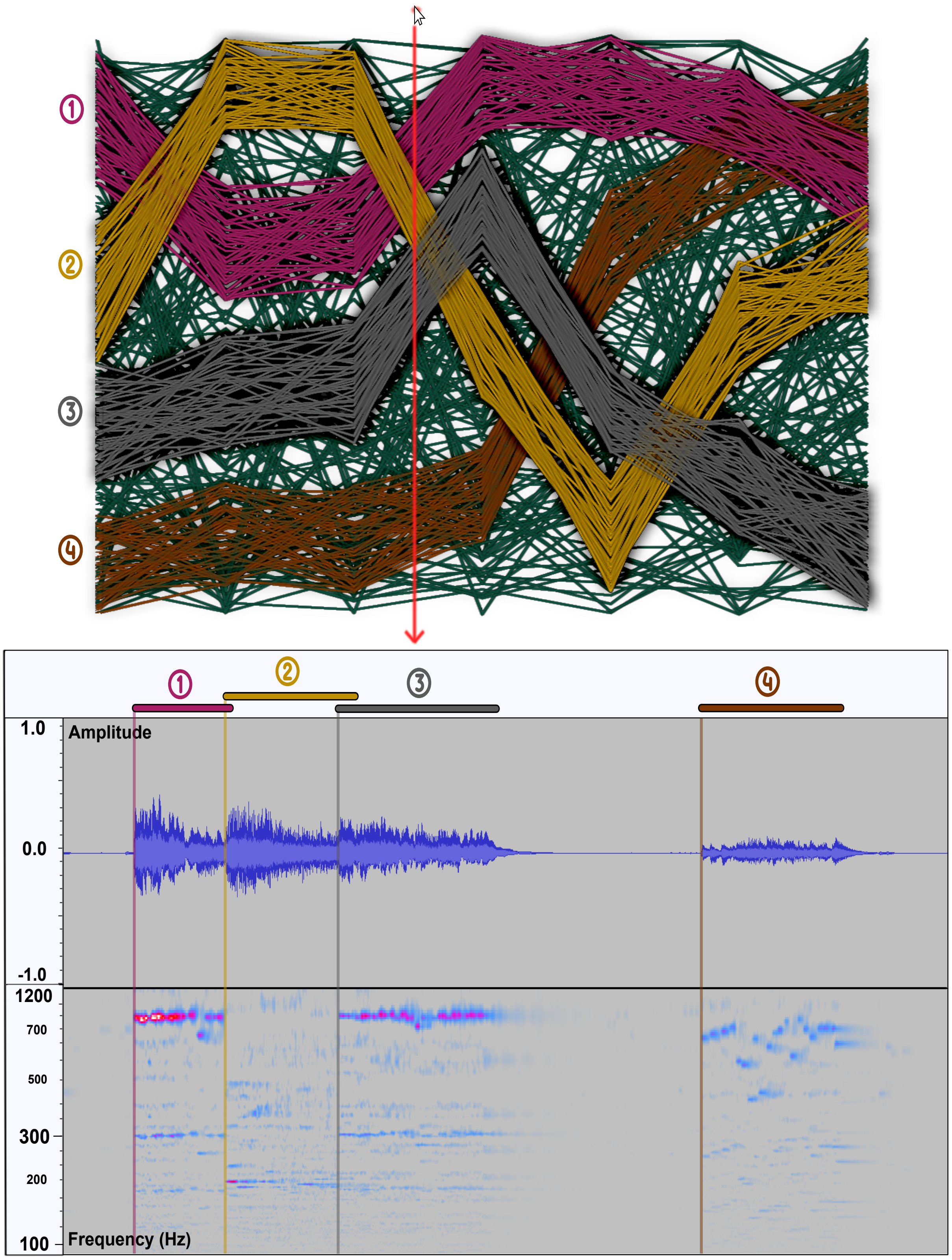


Figure 5.4: The audio visualization represents the sound produced by the mouse path highlighted in red. Clusters within the dataset are numbered, colored, and marked accordingly in the audio visualization. The dataset consists of 150 randomly generated lines, with four distinct clusters, each containing 50 lines. See Video [14].

5.4.2 Lens Functionality

When clusters overlap, the audio feedback generated by our sonification approach prioritizes the most important lines similar to the visuals. However, this is not ideal when we want to analyze obscured clusters, as shown in Figure 5.4, where the gray cluster is rendered on top of the yellow cluster. For the visual channel this can be solved with the use of lenses that displace overlapping clusters [4, 21]. For our application such an approach for the visual channel was incorporated, but the audio channel also requires a function to filter and focus the audio feedback. Figure 5.5 shows how our lens feature can be utilized for filtering and focusing, based on a section of a dataset that has two overlapping clusters.

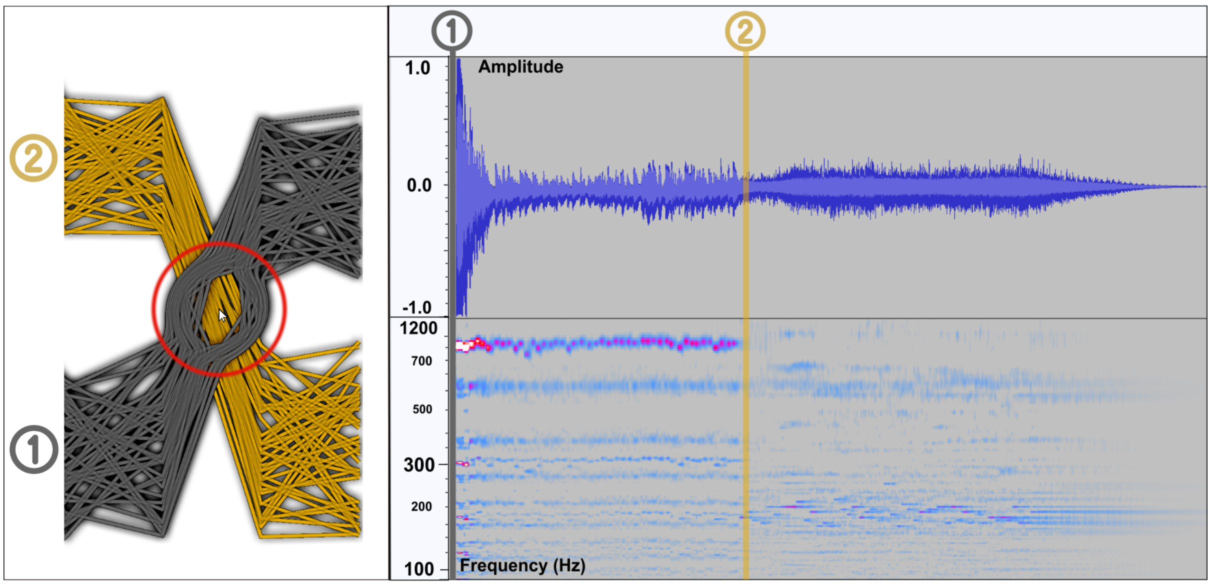


Figure 5.5: On the left two overlapping clusters are visualized, where the grey cluster is visually displaced by the lens. Right audio visualization demonstrates our playback feature, where all intersecting lines within the lens is sonified in order of importance. Each line is "plucked" with a 0.05-second delay (1200 bpm). See Video [17].

The operations for the lens are all mapped to various key bindings, while the most frequent settings are mapped to the mouse, thus ensuring that they are easily available at all times. This can be specifically useful for users with visual impairments as they may have difficulties navigating the GUI. Settings that are mapped to the mouse include the lens radius and the importance threshold. When the threshold is set to visually reveal the yellow cluster, as shown on the left-hand side of Figure 5.5, the corresponding sonification ignores the displaced lines.

Importantly when the importance threshold is manipulated a corresponding line with equal importance and within the lens is sonified. This ensures that one can tune the importance threshold based on audio as well. This functionality can further be utilized for inspecting the lens region in more details, by gradually altering the threshold. Furthermore, using our lens playback feature, all line segments contained within the lens radius are played back in importance order. This use useful for gaining a quick overview without having to manually change the importance threshold. The resulting audio output, visualized on the right-hand side of Figure 5.5, includes both amplitude and frequency. We see that the audio feedback changes frequency when the iteration reaches the lower second cluster (yellow), allowing users to distinguish between overlapping clusters and providing an indication for their overall directionality as well as their homogeneity.

5.4.3 Outliers and Selections

Another important task for dense line graphs, and especially parallel coordinates, is outlier detection [36]. Identifying outliers is important in data analysis because they can have a substantial impact on statistical measures and data interpretations. With or sonification approach detecting outliers can be troublesome, and we therefore also demonstrate how our selection tools can be used to enhance this task. To illustrate this, we will examine two synthetic dataset [6] containing a single artificially introduced outliers with appropriate importance.

In Figure 5.6 the outlier is easily detectable both visually and audibly and is demonstrated in video [18]. However, when dealing with more complex datasets, we recommend utilizing the selection tools. To demonstrate this, we present an example involving a more intricate dataset containing three distinct clusters that produce similar sounds when sonified. This dataset is depicted in Figure 5.7, where the outlier is annotated. In this Figure 5.7 the outlier is selected and the intensity of redness indicate the similarity between the selected line and the remaining lines. For the audio visualizations we include two amplitude visualizations, the first is in regular "importance mode", meanwhile the second is "similarity mode". In similarity mode, the selected line becomes the loudest, while other similar lines are quieter based on their level of similarity. This feature is further demonstrated in video [15]. Both modes can easily be swapped with key presses.

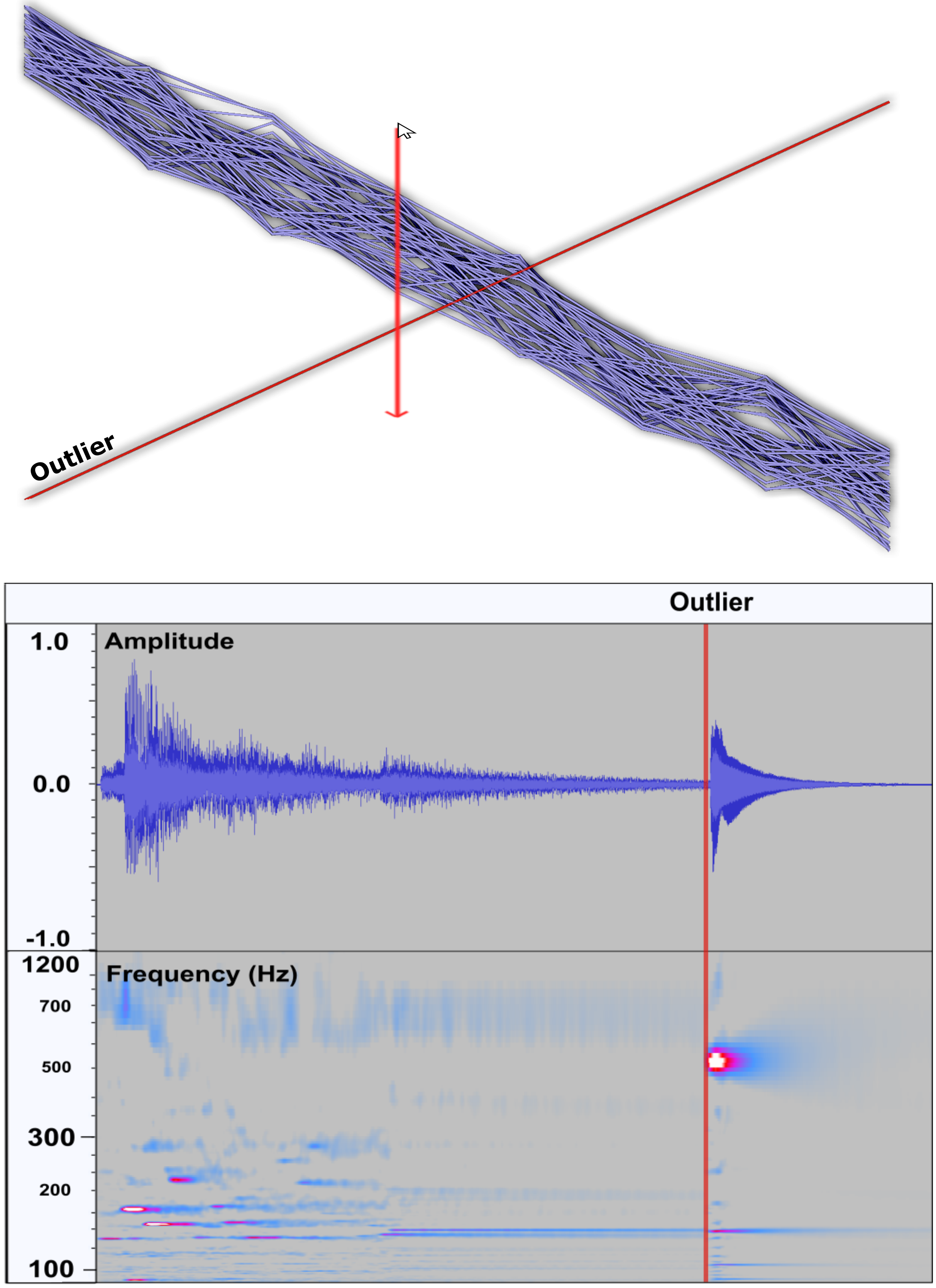


Figure 5.6: Simple scenario where the outlier is easily detected both visually and audibly. The audio is produced based on the red mouse path and has a duration of 5 seconds. See Video [18].

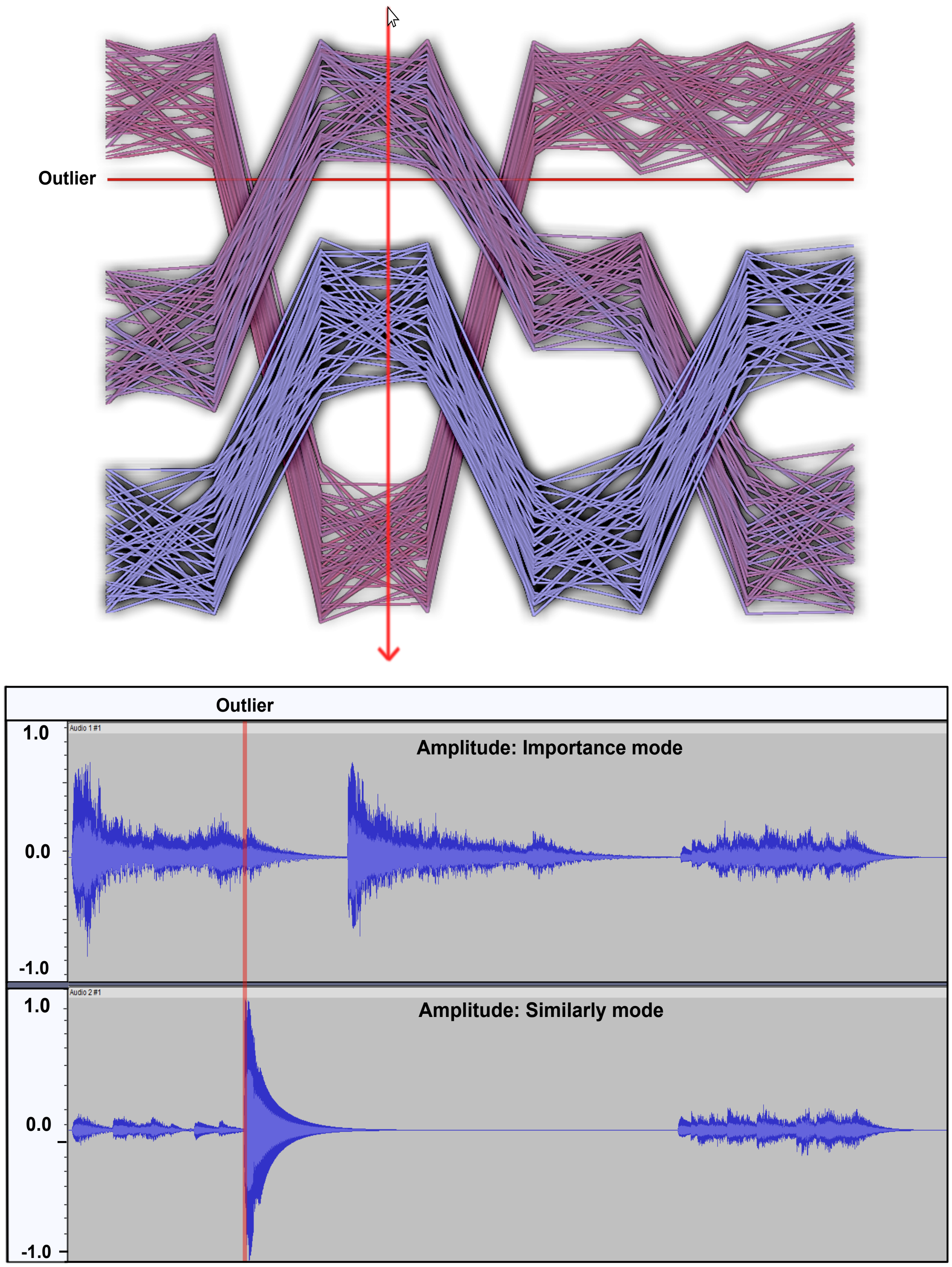
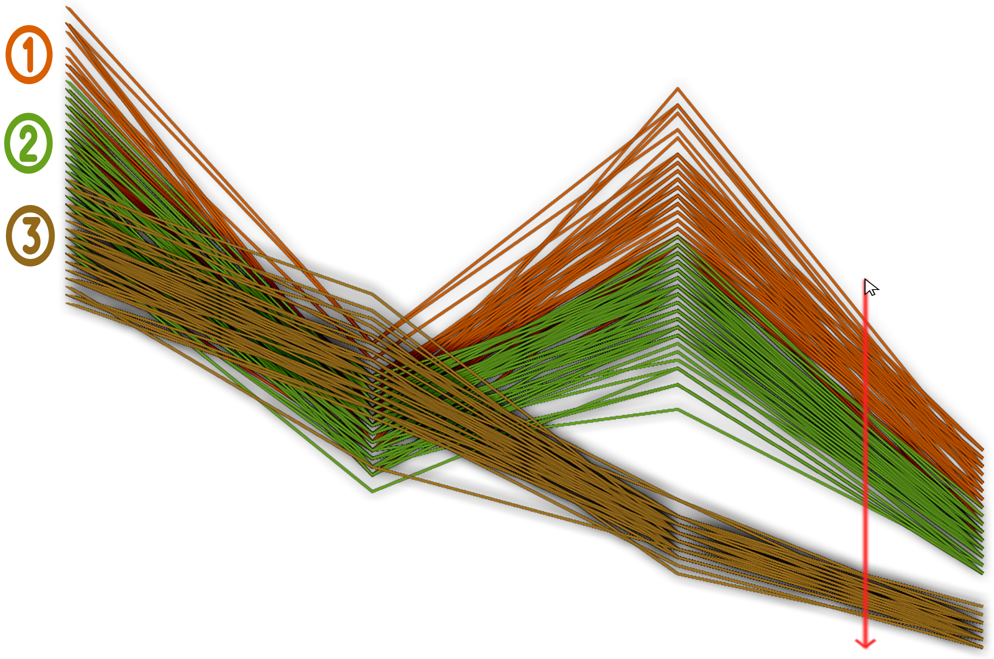


Figure 5.7: Comparison of amplitude output from ”importance mode” and ”similarity mode”. The intensity of redness of lines indicate how similarly lines are. The dataset consists of 3 clusters and 1 outlier. In the audio visualization the outlier is highlighted by a red line. See Video [15].

5.4.4 Real-World Datasets

In previous sections, we demonstrated our sonification approach using synthetic datasets. However, in this section, we present our results using a real-world dataset for a more practical analysis. The dataset we used is derived from the Iris dataset [30], which involves the classification of iris flowers. The goal is to analyze the various characteristics of the dataset with the aid of our sonification technique. The Iris dataset consists of 150 lines, with each cluster or flower species containing 50 lines. Each cluster is assigned a distinct importance value based on its clustering.

The audio visualization in Figure 5.8 represents the amplitude and frequency of a defined mouse path, highlighted in red. By analyzing the audio visualizations, we can identify distinct clusters and approximate their trends in specific areas of the dataset.



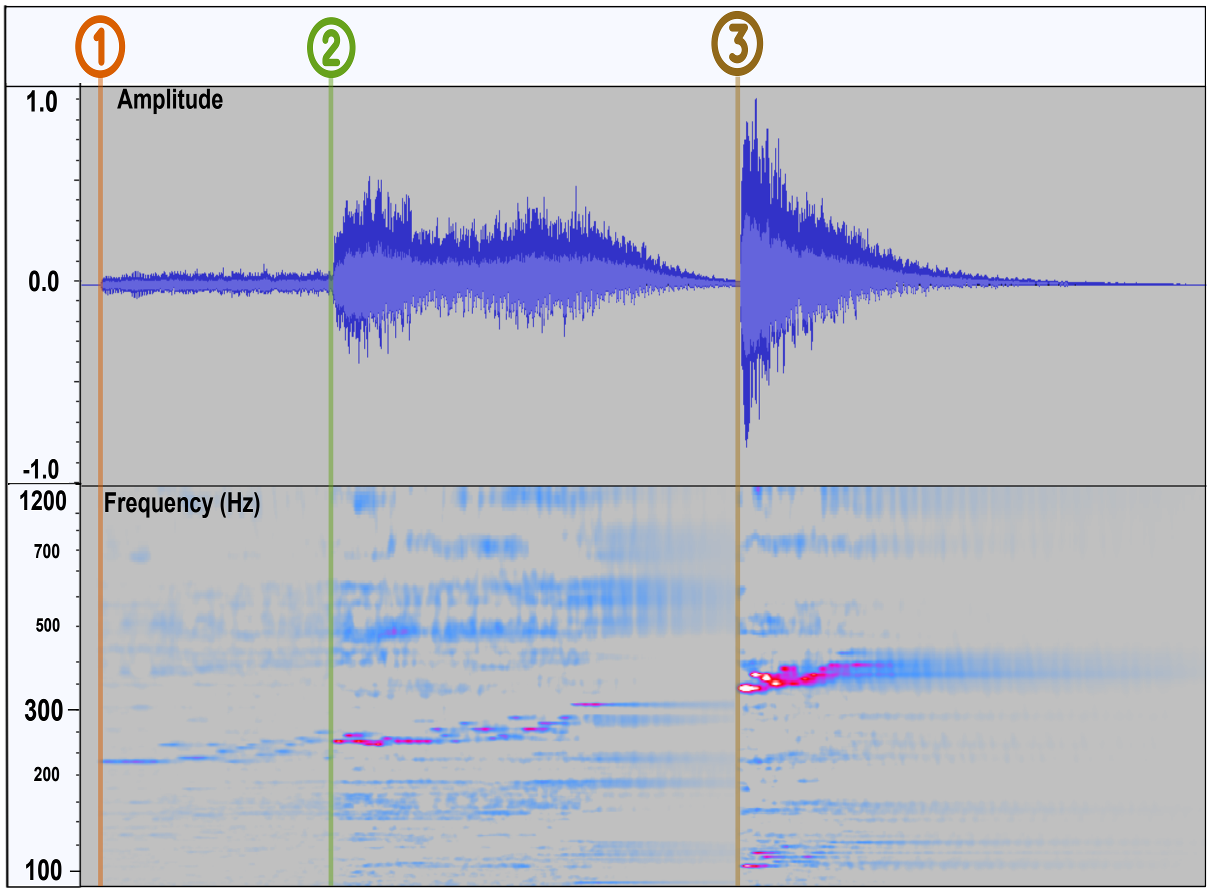


Figure 5.8: The audio visualization represents the sound produced by the mouse path highlighted in red. Clusters within the dataset are numbered, colored, and marked accordingly in the audio visualization. This dataset is derived from the Iris dataset [30]. See Video [16].

Chapter 6

Discussion and Limitations

Based on our experiments, we believe that our approach can support the perception of density variations, clusters and outliers in line charts. Furthermore, we believe that the addition of a sonified lens can further improve the sonification by filtering the audio feedback. However, to substantiate our claims, empirical evaluations and comparisons are required. Previous research already suggests that an interactive approach to sonification of line charts is preferable for complex data visualization over direct mappings [42]. For instance, a study that aimed to identify high density areas in complex line charts found that sonification can improve accuracy [67]. Their approach to audio interactions is comparable to ours, since they used similar mouse line interactions. The study suggests that while sonification can be effective in identifying the most densely populated areas, it may not be as useful for other tasks.

While the main focus of our sonification approach is to enhance data visualizations, it also can benefit the entertainment value of a data visualization. While audio is often overlooked as a channel for displaying data, it has the capacity to transform dull and uninspiring visualizations into captivating auditory experiences. Given that data analysts can spend countless hours poring over dry and unengaging visualizations, incorporating sonification techniques can help to add a new level of interest and engagement to the task at hand. As such, exploring the use of sonification as a means of improving the entertainment value of data visualization is a worthwhile pursuit even if it may not always be viewed as strictly scientific or objective.

Like many sonification models based on complex data visualizations, training is often required for optimal performance. Unlike visual charts, which most humans are taught how to read from a very young age through formal education systems, auditory charts require longer training for optimal usage. While our approach aims to provide an intuitive sonification model, it still suffers from the fact that the audio feedback can be difficult to interpret at times. However, studies have shown that training can improve accuracy in point estimation tasks and may even lead to better performance than a standard visual graph with sufficient training [71, 81]. Take, for instance, an inexperienced individual who listens to the sound of a car engine and senses that there is an issue without being able to pinpoint it. In contrast, a skilled car mechanic can extract highly specific details about the exact source of the problem from that very same sound clue. How training and experience might impact the accuracy of our model requires further evaluation, but generally accuracy should improve over time.

In the context of point estimation-tasks, our sonification approach is limited by lack of auditory context provided, where context refers to the purposeful addition of non-signal information to a display [82]. In visual displays context refers to axes or tick marks which provide some way to estimate the value at any point. For auditory graphs it is suggested to inform users when they reach graph boundaries and use stereo panning to indication location [12]. However, we argue that when using a touch screen, assuming the graph uses the entire screen space, a user will be able detect boundaries and location due to human proprioception. Proprioception is the human ability to sense self-movement, force, and body position [78]. This has also been found to be just as effective for people with reduced eye sight [24].

Chapter 7

Conclusion and Future Work

In this work, we introduced Line Harp, a new sonification approach for dense line charts that combines interactive audio feedback with visual representations of data. We presented an importance-driven sonification method that combines a frequency-based encoding of line direction with interactive lenses and provides a way to focus the audio output. Our directional frequency mapping supports line chart angle perception while also serving as an aid for chart navigation. Furthermore, we proposed a technique that dynamically scales amplitudes to emphasize clustered lines and reduce the overall influence of less important lines to improve density perception. Overall, Line Harp is a promising tool that could potentially enhance the accessibility of visualizations for individuals with visual impairments, or provide a more immersive experience for data analysts.

In the future, there are several directions that can be explored to further expand the scope and applicability of Line Harp. Firstly, it would be valuable to investigate the sonification of other interactive tools beyond dense line charts. For instance, applying a similar approach to sonification for network data [84]- While Line Harp has combined sonification with lenses and selections, the visualization field offers numerous other exciting interactive tools that could also benefit from audio integration.

By exploring the use of sonification in conjunction with different visualization techniques, such as parallel coordinates and other interactive tools, we can enhance the understanding and analysis of complex datasets. This opens up new possibilities for audio-based data exploration and provides a multi-modal experience for users.

To further validate and justify the effectiveness of Line Harp, future empirical evaluations should ideally involve both normal users and visually impaired participants [71].

This approach would provide valuable insights into the overall usability and accessibility of our sonification approach. By comparing the experiences and feedback of different user groups, we can gain a deeper understanding of the potential benefits and limitations of LineHarp for individuals with visual impairments, as well as its potential utility for data analysts in general.

Bibliography

[1] Audacity. Spectrogram view. `https://manual.audacityteam.org/man/spectrogram_view.html`, 2000. Accessed: 2023-March.

[2] Audacity. Audacity waveform. `https://manual.audacityteam.org/man/audacity_waveform.html`, 2000. Accessed: 2023-March.

[3] BBC. Try this bizarre audio illusion! - bbc. `https://www.youtube.com/watch?v=G-lN8vWm3m0`, 2011. Accessed: 2023-May.

[4] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of the ACM SIGGRAPH*, page 73–80, 1993. doi: 10.1145/166117.166126.

[5] T. Birsan and D. Tiba. One hundred years since the introduction of the set distance by dimitrie pompeiu. In F. Ceragioli, A. Dontchev, H. Futura, K. Marti, and L. Pandolfi, editors, *System Modeling and Optimization*, pages 35–39, 2006.

[6] Michael Blumenschein, Xuan Zhang, David Pomerenke, Daniel A. Keim, and Johannes Fuchs. Evaluating reordering strategies for cluster identification in parallel coordinates. *Computer Graphics Forum*, 39(3):537–549, 2020. doi: https://doi.org/10.1111/cgf.14000.

[7] Till Bovermann, Thomas Hermann, and Helge J. Ritter. Tangible data scanning sonification model. In *Proceedings of ICAD*, pages 77–82, 2006.

[8] Stephen A. Brewster, Peter C. Wright, and Alistair D. N. Edwards. An evaluation of earcons for use in auditory human-computer interfaces. In *Proceedings of INTERACT and ACM CHI*, page 222–227, 1993. doi: 10.1145/169059.169179.

[9] Stephen A Brewster, Peter C Wright, and Alistair DN Edwards. Experimentally derived guidelines for the creation of earcons. In *Proceedings of HCI*, pages 155–159, 1995.

[10] Lorna Brown, Stephen Brewster, Rameshsharma Ramloll, Wai Yu, and Beate Riedel. Browsing modes for exploring sonified line graphs. *Proceedings of BCS-HCI*, 2:6–9, 2002.

[11] Lorna M Brown and Stephen A Brewster. Drawing by ear: Interpreting sonified line graphs. In *Proceedings of ICAD*, pages 152–156, 2003.

[12] Lorna M Brown, Stephen A Brewster, SA Ramloll, R Burton, and Beate Riedel. Design guidelines for audio presentation of graphs and tables. In *Proceedings of ICAD*, pages 284–287, 2003.

[13] Egil Bru. Line harp - mouse sonification. `https://youtu.be/155dVn3P8XU`, 2023. Accessed: 2023-May.

[14] Egil Bru. Line harp - clustering. `https://youtu.be/er8QORJrm3g`, 2023. Accessed: 2023-May.

[15] Egil Bru. Line harp - outlier with selection tools. `https://youtu.be/s97j8yurh8Y`, 2023. Accessed: 2023-May.

[16] Egil Bru. Line harp - iris dataset. `https://youtu.be/AIaN44Ou-aU`, 2023. Accessed: 2023-May.

[17] Egil Bru. Line harp - lens playback. `https://youtu.be/6_x4cghfRcQ`, 2023. Accessed: 2023-May.

[18] Egil Bru. Line harp - simple outlier. `https://youtu.be/vp2yT9_rocs`, 2023. Accessed: 2023-May.

[19] Densil Cabrera, Sam Ferguson, and G. W. Laing. Considerations arising from the development of auditory alerts for air traffic control consoles. In *Proceedings of ICAD*, pages 242–245, 2005.

[20] Simon Carlile. Psychoacoustics. In *The Sonification Handbook*, chapter 3, pages 41–61. Logos Publishing House, 2011.

[21] M Sheelagh T Carpendale, D J Cowperthwaite, and F D Fracchia. Distortion viewing techniques for 3-dimensional data. In *Proceedings of IEEE InfoVis*, pages 46–53, 1996. doi: 10.1109/INFVIS.1996.559215.

[22] Ben P Challis and Alistair DN Edwards. Design principles for tactile interaction. In *Proceedings of Haptic HCI*, pages 17–24, 2001. doi: https://doi.org/10.1007/3-540-44589-7_2.

[23] Omar Cornut. Imgui. `https://github.com/ocornut/imgui`, 2023. Accessed: 2020-October.

[24] Hassan Daneshmandi, Ali Norasteh, and Hamed Zarei. Balance in the blind: A systematic review. *Physical Treatments: Specific Physical Therapy Journal*, 11:1–12, 2021. doi: 10.32598/ptj.11.1.430.2.

[25] Datumizer. File:music frequency diatonic scale.svg. `https://commons.wikimedia.org/wiki/File:Music_frequency_diatonic_scale.svg`, 2008. Wikipedia, Accessed: 2023-March.

[26] Evanthia Dimara and Charles Perin. What is interaction for data visualization? *IEEE Transactions on Visualization and Computer Graphics*, 26(1):119–129, 2019. doi: 10.1109/TVCG.2019.2934283.

[27] Alan Dix and Geoffrey Ellis. Starting simple - adding value to static visualisation through simple interaction. In *Proceedings of AVI*, pages 124–134, 1998. doi: 10.1145/948496.948514.

[28] Polly Edman. *Tactile graphics*. American Foundation for the Blind, 1992.

[29] Kajetan Enge, Alexander Rind, Michael Iber, Robert Höldrich, and Wolfgang Aigner. Towards multimodal exploratory data analysis: Soniscope as a prototypical implementation. In *Proceedings of EuroVis*, pages 67–71, 2022. doi: 10.2312/evs.20221095.

[30] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. doi: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[31] John H. Flowers. Thirteen years of reflection on auditory graphing: Promises, pitfalls, and potential new directions. In *Proceedings of ICAD*, pages 406–409, 2005.

[32] John H. Flowers and Terry A. Hauer. Musical versus visual graphs: Cross-modal equivalence in perception of time series data. *Human Factors: The Journal of Human Factors and Ergonomics Society*, 37(3):553–569, 1995. doi: 10.1518/001872095779049264.

[33] William Gaver. The sonicfinder: An interface that uses auditory icons. *Human-Computer Interaction*, 4:67–94, 1989. doi: 10.1207/s15327051hci0401_3.

[34] The Khronos Group. Opengl 4.5 reference pages. `https://registry.khronos.org/OpenGL-Refpages/gl4/`, 2023. Accessed: 2023-April.

[35] The Khronos Group. Opengl. `https://www.opengl.org`, 2023. Accessed: 2023-April.

[36] Julian Heinrich and Daniel Weiskopf. State of the art of parallel coordinates. In *Proceedings of Eurographics*, pages 95–116, 2013. doi: 10.2312/conf/EG2013/stars/095-116.

[37] T. Hermann and H. Ritter. Listen to your data: Model-based sonification for data analysis. *Advances in Intelligent Computing and Multimedia Systems*, 8:189–194, 1999.

[38] Thomas Hermann. Model-based sonification. In *The Sonification Handbook*, chapter 16, pages 399–427. Logos Publishing House, 2011.

[39] Thomas Hermann, Peter Meinicke, and Helge J. Ritter. Principal curve sonification. In *Proceedings of ICAD*, pages 81–86, 2000.

[40] Thomas Hermann, Gerold Baier, Ulrich Stephani, and Helge Ritter. Vocal sonification of pathologic eeg features. In *Proceedings of ICAD*, pages 158–163, 2006.

[41] Thomas Hermann, Andy Hunt, and John G. Neuhoff. Introduction. In *The Sonification Handbook*, chapter 1, pages 1–6. Logos Publishing House, 2011.

[42] Thomas Hermann, Andy Hunt, and John G Neuhoff. *The sonification handbook*, volume 1. Logos Verlag Berlin, 2011.

[43] ISO. Acoustics — standard tuning frequency (standard musical pitch). Technical Report ISO 16:1975, International Organization for Standardization, 1975.

[44] Petr Janata and Edward Childs. Marketbuzz: Sonification of real-time financial data. In *Proceedings of ICAD*, 2004.

[45] Yvonne Jansen, Pierre Dragicevic, Petra Isenberg, Jason Alexander, Abhijit Karnik, Johan Kildal, Sriram Subramanian, and Kasper Hornbæk. Opportunities and challenges for data physicalization. In *Proceedings of ACM CHI*, pages 3227–3236, 2015.

[46] Arthur R. Jensen. Reaction time as a function of experimental conditions. In *Clocking the Mind*, pages 43–54. Elsevier Science Ltd, 2006. doi: https://doi.org/10.1016/B978-008044939-5/50004-5.

[47] Wooseob Jeong and Myke Gluck. Multimodal geographic information systems: Adding haptic and auditory display. *Journal of the American Society for Information Science and Technology*, 54(3):229–242, 2003. doi: https://doi.org/10.1002/asi.10202.

[48] Shakila Cherise S Joyner, Amalia Riegelhuth, Kathleen Garrity, Yea-Seul Kim, and Nam Wook Kim. Visualization accessibility in the wild: Challenges faced by visualization designers. In *Proceedings of ACM CHI*, pages 1–19, 2022. doi: 10.1145/3491102.3517630.

[49] Timothy Justus and Jamshed Bharucha. Music perception and cognition. In *Stevens' Handbook of Experimental Psychology*, pages 453–492. John Wiley & Sons Inc, 2002. doi: 10.1002/0471214426.pas0111.

[50] N. W. Kim, S. C. Joyner, A. Riegelhuth, and Y. Kim. Accessible visualization: Design space, opportunities, and challenges. *Computer Graphics Forum*, 40(3):173–188, 2021. doi: https://doi.org/10.1111/cgf.14298.

[51] Mathieu Le Goc, Lawrence H. Kim, Ali Parsaei, Jean-Daniel Fekete, Pierre Dragicevic, and Sean Follmer. Zooids: Building blocks for swarm user interfaces. In *Proceedings UIST*, page 97–109, 2016. doi: 10.1145/2984511.2984547.

[52] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph.*, 25(3):1206–1213, 2006. doi: 10.1145/1141911.1142016.

[53] matplotlib. Choosing colormaps in matplotlib. `https://matplotlib.org/stable/tutorials/colors/colormaps.html#overview`, 2002. Accessed: 2023-May.

[54] Electronic Musician. Understanding the difference between pitch and frequency. `https://www.musicradar.com/how-to/understanding-the-difference-between-pitch-and-frequency`, 2020. Accessed: 2023-May.

[55] Audrey Nath and Michael Beauchamp. A neural basis for interindividual differences in the mcgurk effect, a multisensory speech illusion. *NeuroImage*, 59:781–7, 2011. doi: 10.1016/j.neuroimage.2011.07.024.

[56] Michael Nees and Bruce Walker. *Auditory Interfaces and Sonification*, pages 507–521. CRC Press, 2009. doi: 10.1201/9781420064995-c32.

[57] John G. Neuhoff. Perception, cognition and action in auditory display. In *The Sonification Handbook*, chapter 4, pages 63–85. Logos Publishing House, 2011.

[58] University of North Carolina. Frequency analysis of sound waves. `https://www.webassign.net/sample/unc/lab_8/manual.html`, 2011. Accessed: 2023-May.

[59] World Health Organization. Blindness and vision impairment. `https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment`, 2022. Accessed: 2023-May.

[60] Sabrina Paneels and Jonathan C. Roberts. Review of designs for haptic data visualization. *IEEE Transactions on Haptics*, 3(2):119–137, 2010. doi: 10.1109/TOH.2009.44.

[61] Dietrich Paulus and Stefan Wirtz. Evaluation of established line segment distance functions. *Pattern Recognition and Image Analysis*, 26:354–359, 2015. doi: https://doi.org/10.1134/S1054661816020267.

[62] M. Pilhofer and H. Day. *Music Theory For Dummies.* Wiley, 2007. ISBN 9780470167946.

[63] Lance Putnam. Gamma: C++ generic synthesis library tutorial. `https://w2.mat.ucsb.edu/gamma/dl/gammaTutorial.0.9.5.pdf`, 2012. Accessed: 2023-May.

[64] Lance Putnam. Gamma. `https://w2.mat.ucsb.edu/gamma/`, 2023. Accessed: 2023-January.

[65] Rameshsharma Ramloll, Stephen Anthony Brewster, Wai Yu, and Beate Riedel. Using non-speech sounds to improve access to 2d tabular numerical information for visually impaired users. In *Proceedings of BCS HCI/IHM*, pages 515–529, 2001. doi: https://doi.org/10.1007/978-1-4471-0353-0_32.

[66] Majken K. Rasmussen, Esben W. Pedersen, Marianne G. Petersen, and Kasper Hornbæk. Shape-changing interfaces: A review of the design space and open research questions. In *Proceedings of the SIGCHI*, page 735–744, 2012. doi: 10.1145/2207676.2207781.

[67] Niklas Rönnberg and Jimmy Johansson Westberg. Interactive sonification for visual dense data displays. In *Proceedings of ISon CITEC*, pages 63–67, 2016.

[68] K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering: introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004. doi: 10.1109/MCG.2004.1274058.

[69] Penelope Sanderson, S Eunice, L Philippe, and W Alexandra. Auditory alarms, medical standards, and urgency. In *Proceedings of ICAD*, 2006.

[70] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of IEEE Symposium on Visual Languages*, pages 336–343, 1996. doi: 10.1109/VL.1996.545307.

[71] Daniel R. Smith and Bruce N. Walker. Effects of auditory context cues and training on performance of a point estimation sonification task. *Applied Cognitive Psychology*, 19(8):1065–1087, 2005. doi: https://doi.org/10.1002/acp.1146.

[72] Tableau software. 5 tips on designing colorblind-friendly visualizations. `https://www.tableau.com/blog/examining-data-viz-rules-dont-use-red-green-together`, 2016. Accessed: 2023-April.

[73] Saiganesh Swaminathan, Thijs Roumen, Robert Kovacs, David Stangl, Stefanie Mueller, and Patrick Baudisch. Linespace: A sensemaking platform for the blind. In *Proceedings of CHI*, page 2175–2185, 2016. doi: 10.1145/2858036.2858245.

[74] Christian Tominski and Heidrun Schumann. *Interactive Visual Data Analysis*. AK Peters Visualization Series. CRC Press, 2020. doi: 10.1201/9781315152707.

[75] Christian Tominski, S. Gladisch, Ulrike Kister, Raimund Dachselt, and H. Schumann. Interactive lenses for visualization: An extended survey. *Computer Graphics Forum*, 36(6):173–200, 2017. doi: 10.1111/cgf.12871.

[76] T. Trautner and S. Bruckner. Line weaver: Importance-driven order enhanced rendering of dense line charts. *Computer Graphics Forum*, 40(3):399–410, 2021. doi: https://doi.org/10.1111/cgf.14316.

[77] René Tünnermann, Kolbe, Till Lukas Bovermann, and Thomas Hermann. Surface interactions for interactive sonification. In *Proceedings of ICAD*, page 166–183, 2009. doi: https://doi.org/10.1007/978-3-642-12439-6_9.

[78] John C. Tuthill and Eiman Azim. Proprioception. *Current Biology*, 28(5):194–203, 2018. doi: https://doi.org/10.1016/j.cub.2018.01.064.

[79] B Walker and David Lane. Psychophysical scaling of sonification mappings: A comparison of visually impaired and sighted listeners. In *Proceedings of ICAD*, pages 90–94, 2001.

[80] Bruce Walker. Magnitude estimation of conceptual data dimensions for use in sonification. *Journal of experimental psychology: Applied*, 8(1):211–21, 2003. doi: 10.1037/1076-898X.8.4.211.

[81] Bruce N Walker and Michael A Nees. Brief training for performance of a point estimation sonification task. In *Proceedings of ICAD*, 2005.

[82] Bruce N Walker and Michael A Nees. Theory of sonification. *The sonification handbook*, 1:9–39, 2011.

[83] R. Wang, C. Jung, and Y. Kim. Seeing through sounds: Mapping auditory dimensions to data and charts for people with visual impairments. *Computer Graphics Forum*, 41(3):71–83, 2022. doi: https://doi.org/10.1111/cgf.14523.

[84] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: an interactive method for managing edge congestion in graphs. In *Proceedings of IEEE InfoVis*, pages 51–58, 2003. doi: 10.1109/INFVIS.2003.1249008.

[85] Wai Yu and S. Brewster. Comparing two haptic interfaces for multimodal graph rendering. In *Proceedings of HAPTICS*, pages 3–9, 2002. doi: 10.1109/HAPTIC.2002.998934.

[86] Wai Yu, Ramesh Ramloll, and Stephen Brewster. Haptic graphs for blind computer users. In *Haptic Human-Computer Interaction*, pages 41–51, 2001. doi: https://doi.org/10.1007/3-540-44589-7_5.