Digital Media Technology Lab
Birmingham City University

Automating the Production of the Balance Mix in Music Production

Nicholas Jillings

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy, May 2022

Abstract

Historically, the junior engineer is an individual who would assist the sound engineer to produce a mix by performing a number of mixing and pre-processing tasks ahead of the main session. With improvements in technology, these tasks can be done more efficiently, so many aspects of this role are now assigned to the lead engineer. Similarly, these technological advances mean amateur producers now have access to similar mixing tools at home, without the need for any studio time or record label investments. As the junior engineer's role is now embedded into the process it creates a steeper learning curve for these amateur engineers, and adding time onto the mixing process.

In order to build tools to help users overcome the hurdles associated with this increased workload, we first aim to quantify the role of a modern studio engineer. To do this, a production environment was built to collect session data, allowing subjects to construct a balance mix, which is the starting point of the mixing life-cycle. This balance-mix is generally designed to ensure that all the recordings in a mix are audible, as well as to build routing structures and apply pre-processing. Improvements in web technologies allow for this data-collection system to run in a browser, making remote data acquisition feasible in a short space of time. The data collected in this study was then used to develop a set of assistive tools, designed to be non-intrusive and to provide guidance, allowing the engineer to understand the process.

From the data, grouping of the audio tracks proved to be one of the most important, yet overlooked tasks in the production life-cycle. This step is often misunderstood by novice engineers, and can enhance the quality of the final product. The first assistive tool we present in this thesis takes multi-track audio sessions and uses semantic information to group and label them. The system can work with any collection of audio tracks, and can be embedded into a poroduction environment.

It was also apparent from the data that the minimisation of masking is a primary task of the mixing stage. We therefore present a tool which can automatically balance a mix by minimising the masking between separate audio tracks. Using evolutionary computing as a solver, the mix space can be searched effectively without the requirement for complex models to be trained on production data.

The evaluation of these systems show they are capable of producing a session structure similar to that of a real engineer. This provides a balance mix which is routed and pre-processed, before creative mixing can take place. This provides an engineer with several steps completed for them, similar to the work of a junior engineer.

Acknowledgements

I would like to thank my supervisory team Ryan Stables, Cham Athwal and Joshua Reiss for all their support, guidance and patience throughout my PhD life. The time spent doing this work was made all the more enjoyable, and I cannot give enough thanks for this. The lab has changed so much during my years there, but whenever I needed to look up I could always be counted on to see a friendly face. To name but a few in Sean 'Grand Orca' Enderby, Matthew 'With two T's' Cheshire, Samuel 'Grip' Smith, Spyros 'The Cypriot' Stasis, Dr. Ian 'You submitted it yet' Williams, Carl 'Empire State' Southall, Jason 'Mod-Man' Hockman, Alan 'Tricky' Dołhasz, Muadh 'Nose' Al-Kalbani. Thank you all so much for the amazing time spent solving all the worlds problems over a pint or three.

Thank you to my family for giving me the support to do this, whenever I was questioning if I should keep going you all provided me with the right words of encouragement!

And now I have transformed, into my lifelong dream of being a Simpsons character (pls don't sue me Matt Groening).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In popular music, releasing a piece of music commercially requires multiple stages, which are completed by a team of people. Song-writers will help compose a song with an artist, with guidance from a producer to ensure the record label's intentions are accurately represented. The artist will then record the performance of the song during a dedicated set of sessions with a recording engineer in a studio. Then a mix engineer, alongside a team of producers and assistants, will turn these recorded tracks into a production mix for distribution.

Arguably, the most complex task is the production phase, where artists and engineers interact with each other to create the end product. The engineer has to take the recordings by the artist and produce a mix which conveys their intention in the release. Years of expertise and training is required for producers to become experts in their field.

One early phase of the mixing life cycle is the balance mix. This stage is used to set up the mixing structures, by ensuring that all aspects of the mix are not just heard but fit to a style the artist wants. However there is conflicting agreement on what tasks should be taken to complete a balance mix, and no information on how engineers even achieve this preliminary goal. With this in mind, this thesis will show how even the balance mix requires hundreds of user interactions, and how certain decisions can be detrimental to the mix quality from this early goal. With this new data set it is possible to extract novel processes to help the engineer achieve a higher quality mixing outcome, both in time saving and quality output.

1.2 Objectives and Research Questions

The objective of this thesis is to understand the processes of the balance mix from the engineers perspective and build a set of assistive tools to help the engineer produce a balance mix. This is supported by several research questions: To achieve this aim, we attempt to answer the following research questions:

1. Can high quality data be gathered outside of the studio environment?
2. What mixing decisions have the highest impact on the mix quality?

3. What grouping structures are made at the balance mix phase?
4. What is the goal of the balance mix and what features are most important to the engineer?
5. Can the track grouping and naming be automated?
6. Can the balance mix be automated to produce a mix which is suitable for a mix engineer to use?

Each of these questions are answered in the following chapters of the thesis.

1.3 Thesis Structure and Publications

Each of the following chapters aims to answer a different research question, along with the introduction of novel tools and techniques throughout. The following list gives an overview of each chapter with their outputs,

- **Chapter 2 Literature Review** This chapter is focused on the review of existing materials surrounding mixing production, roles of the studio and the production life cycle of a song. The chapter provides the context of music production from early historical productions through to modern automated digital systems to show the evolution of the role of the engineer in the production life cycle.
- **Chapter 3 Data collection** This chapter presents an overview of applicable research methodologies for analysing mixing practices. Traditional data collection methodologies would not provide a suitable level of detail or would place significant burdens on the research time. This then shows the development of the web powered Digital Audio Workstation for data collection, showing the frameworks used and how the data is collected over time. The system provides a familiar environment for the engineer, which is used to efficiently gather mixing information directly from engineers.
  - Web Audio Evaluation Tool: A framework for subjective assessment of audio (Jillings et al., 2016c)
  - Js-xtract: A real-time audio feature extraction library for the web (Jillings et al., 2016b)
  - JSAP: A Plugin Standard for the Web Audio API with Intelligent Functionality (Jillings et al., 2016d)
  - JSAP: Intelligent audio plugin format for the Web Audio API (Jillings et al., 2016a)
  - An Intelligent audio workstation in the browser (Jillings and Stables, 2017c)
- **Chapter 4 Investigating Current Music Practices** With the data collection platform established, this chapter presents a series of data collection experiments. Engineers are given a set of multi-track recordings and are asked to produce a balance mix. This is a mix which uses only spatial, gain and routing information. The data is then analysed to extract quantitative information on early-stage mixing practices.
  - Investigating Music Production Using a Semantically Powered Digital Audio Workstation in the Browser (Jillings and Stables, 2017d)
- **Chapter 5 Instrument Grouping** From the study in Chapter 4 it is clear that grouping is an important stage of mixing to producing a good balance mix, however many engineers do not utilise grouping

functionality this early. Grouping is commonly used to help structure a session for easier navigation and processing. For example, it is very common to group all the drums together since a drum kit could be upwards of 10 individual tracks. A channel grouping and routing system is presented to automatically route tracks to groups. The system groups sources based on their instrument labels and provides suitable labels for the created groups.

  - Automatic channel routing using musical instrument linked data (Jillings and Stables, 2017a)

- **Chapter 6 Automatic Masking Minimisation** From the study in Chapter 4, we show that the aim of the mixing phase is to predominantly minimise masking between the tracks. To facilitate this, an automatic system which can minimise the masking between a set of unknown tracks is developed. Using auditory models, it is possible to calculate the amount of perceived masking between tracks. Evolutionary computing is then used to efficiently search the mix-space to find a suitable set of gain coefficients between the tracks which met the given cost functions. The mixes were evaluated against other automatic mixing systems to show if the system was able to effectively minimise masking and produce a better mix than the default starting positions.

  - Automatic masking reduction in balance mixes using evolutionary computing (Jillings and Stables, 2017b)

1.4 Scope

This study will primarily focus on western commercial music, using a wide variety of genres to ensure the work is validated. Each piece will use the same audio data set where possible to ensure validity across the studies. All music used is from freely available unprocessed multi-track recordings with a permissive licence for educational and research purposes (Senior, 2019). The participants were found using auditory mailing lists for the online participation's, however the laboratory controlled study would be made up of members of Birmingham City University, primarily by students of the School of Computing, Engineering and the Built Environment.

1.5 Outputs

The beneficiaries of the outputs from this thesis will be two fold. The first group are for individuals who may not know how to produce a mix to a production level from inexperience or lack of professional training to understand the actions being taken. By creating novel automated tools these individuals can improve the quality of their mixes whilst being able to observe what the decisions are being made. This can allow junior and amateur engineers to leverage powerful mixing tools without needing a deep knowledge of that tool to achieve their goal. The second group are for studio engineers who are time-limited in their work, the outputs can reduce time spent by automating decisions which would be made by the engineer anyway. This group will benefit from the time-save, allowing them to focus on the creative aspects of the mix and leaving the automation to the computer.

The first output is the discovery of the mixing practices used by engineers in Chapter 4. The system shows the intense number of actions required by engineers throughout the system to build up to a completed balance mix. The chapter also showed the decisions and goals used by the engineers to complete the task, such as to use groupings and what the spectral information shaped to be.

The second output is the automated instrument grouping system, which takes a set of tracks and suggests group structures for them. Instead of relying on features from the audio it uses linked meta-data from crowd sourced web data sets to build a grouping structure. This system has an advantage over other solutions because of its flexible nature to be used with different knowledge basis, continuous improvements to the data set being used and not requiring computationally expensive feature extraction or processing on the client device.

The third output is an automatic mixing tool using mask minimisation as its goal. This goal was discovered from Chapter 4 and was used as the requirement for the mixing tool. Because of the search-like nature, a solver is used and its algorithm presented and tested against other possible solutions. This provides a solution for mix exploration which can be rapidly developed for mixing goals.

From this thesis a new web based Digital Audio workstation was developed and deployed. This was used as the basis for a funded spin-out from the University, Semantic Audio Labs Ltd and was branded as Faders.io. This site holds the realised outputs from this thesis, as well as outputs from Dr. Ryan Stables, Dr. Sean Enderby and Dr. Brecht De Man. To date the company has raised investment and output patents, listed below, in support of the work formulated in part through this PhD.

The following is a list of published materials made during the PhD.

- Js-xtract: A real-time audio feature extraction library for the web (Jillings et al., 2016b)
- JSAP: A Plugin Standard for the Web Audio API with Intelligent Functionality (Jillings et al., 2016d)
- JSAP: Intelligent audio plugin format for the Web Audio API (Jillings et al., 2016a)
- Audio processing chain recommendation (Stasis et al., 2017a)
- An Intelligent Audio Plugin Framework for the Web Audio API (Jillings et al., 2017)
- Investigating Music Production Using a Semantically Powered Digital Audio Workstation in the Browser (Jillings and Stables, 2017d)
- An Intelligent audio workstation in the browser (Jillings and Stables, 2017c)
- Automatic channel routing using musical instrument linked data (Jillings and Stables, 2017a)
- Audio processing chain recommendation using semantic cues (Stasis et al., 2017b)
- Investigation into the effects of subjective test interface choice on the validity of results (Jillings et al., 2018)
- Patent WO2021069630A1 Digital Audio Workstation (Jillings et al., 2019)
- Patent GB2595456A Collaboration system (Jillings et al., 2020)

Chapter 2

Literature Review

Music production has undergone multiple technological revolutions since the first recordings used phonographs (Burgess, 2014, p. 16). Disruptive technologies have modified the roles and responsibilities of the artists, engineers and producers within the studio. Whilst in previous large format recording houses there would be a team of engineers, the ability to mix, edit, and record using faster and smaller technologies has caused the roles between the personnel to blur. The digital age has reduced the workload, with previously laborious tasks such as session recall, now being instantaneous (Burgess, 2014, pp. 101).

The technological revolution has sparked a new field of audio technology, intelligent music production (Reiss, 2011). Systems have been developed for a range of the mixing process from volume Perez-Gonzalez and Reiss (2009), panning (Mansbridge et al., 2012b), time alignment (Hockman et al., 2008) and audio effects (Ma et al., 2013, 2015; Stasis et al., 2016). These systems are generally based upon taking over an aspect of the mixing process from the engineer, either to save time for the engineer by reducing the workload or to provide an improved starting point. These are often based upon numerical models or instructions from texts which are conflicting in their instructions.

This chapter will give a background into the balance mix and what supporting literature exists from experts and engineers on how this should be performed. Then the chapter will explore the tools of the studio, from current technology to assistive and automatic or intelligent systems which have been developed. This will give a grounding into the various roles that have been born into the industry and how they have evolved with technological changes. The chapter will then explore the methods of data collection that could be used to support the future chapters work, as well as common themes such as masking, machine learning and web technologies.

2.1 The Balance Mix

It is known that different engineers working on the same mix will produce different finished mixes, due to their approach or personal styles (Ronan et al., 2015b; De Man and Reiss, 2017). Traditionally, engineers are taught to start with a 'balance mix' or 'rough mix' (Izhaki, 2012, p. 35). This is defined as a basic mix, using minimal

processing, aimed at organising the session for further work (Izhaki, 2012). John Leckie states the aim of a balance mix is to 'simply hear what you've got' (Senior, 2019, p. 132), with the aim to make each track equally audible. The mix is called a balance mix since it is aimed at balancing the tracks, ensuring things are panned appropriately and that the levels are at the rough position. This mix is unprocessed, only utilising the volume, panning and routing controls to define the layout of the song (Senior, 2019). Typically this process focuses on making all sources intelligible, and making the stereo image sound appropriate for the genre of music (Izhaki, 2012). Once the balance stage is complete, engineers can move on to improving the mix for artistic and aesthetic purposes (Senior, 2019, p. 269).

There are multiple texts which present a standard process for mixing a set of tracks into a produced mix, all with varying steps. Engineers are encouraged to start first by auditioning the session before doing any other mixing process Izhaki (2012). Engineers should also make a rough reference mix to help guide them through the process and to focus on the broader changes early on Izhaki (2012). Timing, tuning and editing should be completed before any mixing should take place Senior (2019). Then volume controls and noise removal should also be corrected (Owsinski, 2017, p. 43). This can be done, for example, by adding high-pass filters to tracks that do not have any reasonable energy in the low frequencies (Senior, 2019, p.139). The logic is to remove any unintentional rumble or noise from a signal which could interfere with the low-frequency content of the mix. Multiple sources also suggest creating control structures at an early stage in the mixing process This involves send busses for side-chain processing, laying out the tracks in the appropriate order and even colour-coding or applying other forms of quick identification (Owsinski, 2017, pp. 49-51). Which passage of the arrangement to start with is also up for debate, with suggestions that engineers should focus on the chorus first, then the verses, followed by additional components (Senior, 2019, p. 132).

There is little consensus on the approaches that an engineer could take, without any information or data on whether these approaches are effective or well-used. Most sources agree that a balance mix is an important step to take, and that this should comprise of minimal processing to be effective Izhaki (2012). The balance mix should have minimal processing in the beginning, except corrective processing such as high-pass filtering of tracks which should have no low energy content (Senior, 2019, p.139) The sources do give an overview of a treatment pipeline, saying engineers should start with adjusting the level positions, pan positions, adding processors and finally automation (Izhaki, 2012). Although other sources state the engineer should start with corrective processing, the pan balance instead of the fader positions (Senior, 2019, p. 132). This is the reverse of the pipeline proposed by Izhaki (2012) and Owsinski (2017) which proposes a coarse-to-fine mixing process starting with faders, pan and then adding processing after. There is agreement that engineers start with larger, coarse actions before refining down to fine-tuned parameter adjustments as the mix nears completion.

Interviews with several mixing engineers expose two predominant approaches: top down and bottom up (Owsinski, 2017, p. 69). The top down approach is performed by starting with the most important track isolated and then introducing additional sources individually (Izhaki, 2012). This means starting with a lead instrument, such as the vocals or the primary instrument in the mix and then adding in the next track to build up the mix. One advantage of this approach is that the lead is always the focus and will be prominent in the

mix. The alternative bottom up approach has the engineer start with a sub-mix such as a rhythm section, and then will add in the harmonic tracks, finishing with the lead (Owsinski, 2017, p. 69). This approach has the engineer build upon a bed of tracks and can have them mix with more of the activity at once.

Music production can be mathematically described as a search task to optimise a set of parameters which best fit a performance criteria detailing an acoustically pleasing mix (Terrell et al., 2014). Each time the engineer makes a mixing decision and changes the mixing parameter, they will change the state of the mix and thus alter the perception of the mix. This can be represented by giving each track a set of specific parameters to form part of the mix-down process. Each track receives a set of controls, usually through a mixing console or a DAW, to modify the incoming audio. All of the control vectors could then be said to describe the mix state, making it completely reproducible. An engineer will listen to the audio and make certain adjustments based on their criteria for how the audio should be presented to the listener. This process almost exactly mimics the process of a machine learning algorithm, to modify and evaluate, repeating until a solution is found (Terrell et al., 2014).

$$y[n] = \sum_{m=0}^{M} (g_m[n]x_m[n]) \tag{2.1}$$

Equation 2.1 gives the generic description of a single track mixing environment of M number of input tracks (Terrell et al., 2014). Each input track x_m is multiplied with a gain coefficient g_m and then summed with the other tracks to give an output vector y. Most DAWs start the mix at the unity, with everything set to unity: pan is set to centre, the fader is at 0dB gain and all tracks feed the master bus directly. In this case $g = [1.0, 1.0, ..., 1.0]$ such that the output is just the direct summation of the input tracks. The role of the engineer during the mixing stage is to modify this vector such that the mix quality is improved (Terrell et al., 2014). This starting position itself has also been shown to have an impact on the the mixing process, as varying the initial gain structures often produces variation in the final structure (Wilson and Fazenda, 2015b). Before creating any algorithms or functions to optimise the mixing process, data needs to be obtained explaining how engineers approach a mixing task.

Analysis into musical content has mostly been focused on the final mix. This is due to the relatively easy access of acquiring end consumer content through CD archives or media libraries (Pestana et al., 2013). It has shown a trend in production changes over time, with newer production mixes usually having higher bass levels in their mix than earlier recordings, with a peak spectral energy being experienced near the 100Hz mark (Pestana et al., 2013).

2.1.1 Mixing Studies

The methods and processes for performing the balance mix are found extensively in literature on music production, however there is little in the form of studies on how the balance mix is performed and its impact on the mix. It is known that engineers have a large part to play in the outcome of the mix when given the same set of tools to produce the song (De Man et al., 2015). And that there are cultural differences between engineer, due to their education or other factors, which have a measurable impact on the outcome of the song

(Pras et al., 2018). However there is not a large amount of information into exactly how an engineer solves the mixing task, such as which controls are used more than others and how the culmination of actions impacts the performance of the produced mix.

The practices from literature and interviews were evaluated to show if certain commonly-held statements by engineers are actually true or false (Pestana et al., 2014). This study showed that ideas that all tracks should be of an equal loudness are fundamentally flawed, although this is in the final mix and there is no perception that this would not be a starting point. In fact, when asked engineers showed that only 1-10% of their mixes actually exhibited equal loudness across all sources. Other common factors for engineers to work towards includes ensuring the left and right track have equal energy. 928 commercial mixes were evaluated for their stereo characteristics and showed that the energy difference between the left and right channels rarely exceeded 0.8dB. The study concluded that masking is the most important factor an engineer has to deal with, which affected over a quarter of all the mixing aspects the study processed,

A study into the mixing environments used by mix engineers showed that there is a large degree of variability in what the engineers liked, with engineers themselves choosing different qualities that suited their style (Tervo et al., 2014). Few things were agreed upon except that lower reverberation time T_{60} generally was more likely to be preferred over other spaces, but this correlation was not exceptionally strong. Likewise the clarity of the room also plaid an important role.

A similar study looked into the performance of an engineer in the studio from a discussion stand point, rather than an objective measurement of what the engineers would do (Anthony, 2018). This study interviewed several engineers to ask their perception on the role of the studio as an instrument and part of the performance of the piece. From this study, a few key themes were identified that were important or relevant to engineers. Firstly that engineers do not need the latest equipment or technology, but will use a system they feel comfortable working on. This will translate into learning techniques and processes unique to that system, giving an engineer a style that is synonymous to them. This shows the importance in new technology to assist and aid as opposed to control and black-box the design process. The study also showed that mix engineers need to have the system set up as complete as possible prior to the mixing phase occurring, which will ensure the mix engineer stays in the correct frame of mind throughout the mixing phase and not have to correct items or reconfigure items.

One study into mixing engineer performance asked a very appropriate question "Do mixing engineers hear the same thing?" (Bitzer et al., 2008). The task required twenty-two mixing engineers to process a file by sweeping through the track with a high-Q filter. This type of filter will create a large resonance at the given frequency. The engineers would then report the most perceptually salient frequencies. Their study showed that engineers were in agreement on wide band sources, but for narrow-band the agreement drops to 50% or less for each file. Whilst the study did not prove the engineers disagree on what they hear, mostly because of the difficulties in translating the filter parameters to salient frequencies, it did not prove that they do agree. The filter translation occurs because the filters can still be quite broad, so in regions with higher-bandwidths, it is possible that engineers are talking about the same frequency because it is still getting significantly boosted, or cut, by the filter.

The level preference of balance engineers was studied in a limited paper to show how much 'expert listeners' vary their preference to level over time (King et al., 2010). Whilst the study was aimed primarily at showing variance of engineers during listening trials, the variance will also show that an engineer's perception will change over time when they are performing any critical listening task, including production tasks. The study had engineers perform a level task, by balancing the level of a stereo track of a solo instrument with a pre-mixed backing track. This task would simulate a real-world production task, where the engineer would possibly be completing the final task of the balance mix. Each engineer repeated the task eight times for three different genres, giving 24 trials per engineer.

A week later the engineers repeated the same task again. The study showed that there was a degree of variability between the levels chosen by the engineers, but the most significant factor which drove that variance was experience. If an engineer was in-experienced they would show a high degree of variability between the two studies. More experienced engineers tended to be consistent between the study trials. This shows two interesting insights as well. Firstly that the balance mix task itself is again not an agreed upon system, with a high range of options chosen between the engineers. And secondly that the experience of the engineer has an impact on the repeatability of the results in the mix.

Given the amount of conflicting information in texts on what engineers should do in a mixing environment, one study proposed a recommendation system aimed at aiding engineers select a suitable workflow for the audio mixing task (Sauer et al., 2013). Their system was aimed primarily at educational and junior engineers who may be less experienced or familiar with the mixing processes. The system relied upon understanding the terminology and syntax of audio engineers when communicating with artists or producers who use more semantic and descriptive terms. A similar system was designed for specific plugins to take semantic terms and match it to plugin parameters (Stables et al., 2014). The study gained its data from engineers during studio sessions to observe their processes as well as interviews with engineers afterwards. From this a knowledge model was constructed which could be used to query with a set of natural language terms to give engineers a suitable approach or solution to a problem. Whilst the novel artefact is deeply impressive, and possibly even more so with more recent developments in natural language processing, the system was still limited to a narrow scope of problem solving for plugin control. Again, the study and interviews of how the engineers actually solved the problem was not discussed or published.

The tools used by the engineer could impact on the performance of the mix engineer to complete the mixing task (Wu et al., 2019). A single song was mixed by an engineer in GarageBand using iOS and Logic Pro X in a professional studio. The rationale was that the GarageBand on iOS was readily available to consumers whilst the professional level software in a studio should elicit a better performance. The performance of the two mixes was evaluated by performing a subjective listening test on twenty individuals, ten with mixing experience and ten without. The results overwhelmingly showed the professional system scored better than the commercial system, although there are issues which question the validity of the results.

Firstly, only one song was produced by one engineer which could lead to learning bias as the engineer becomes familiar with the task. It was not made clear which mix was produced first, although both were done in the

same studio. The second problem is the questioning used during the listening test. When the question asked "Which version sounds like it was done in a studio" the 20 respondents said 30% the iOS, 30% the pro and 40% both. This shows that whilst the listeners had a preference, it does not actually show that one had better quality than the other.

Music mixing is often reduced to a problem of minimising masking and interference with important tracks from backing tracks (Izhaki, 2012). A study looked into how engineers would prefer to solve this mix problem using three masking reduction techniques (Wakefield and Dewey, 2015). From this, three well used masking reduction techniques were studied: mirrored EQ (Mynett et al., 2010), frequency spectrum sharing and stereo panning (Mansbridge et al., 2012a). Mirrored EQ involves inverting the EQ of a priority track on a sub-priority track, such that the salient frequencies of the primary track are not just boosted but also removed from the interfering tracks as well. Spectrum sharing involves low pass filtering one instrument and high-pass filtering another, causing the spectrum to be split although this would have limited usage on certain tracks. Stereo panning involves panning two instruments which are competing for the same spectrum away from each other, which has shown to improve perception in able-hearing listeners. The engineers were all given a blind control surface, where a single slider controlled the underlying property. Each engineer mixed three two-track mixes which contained common pairs of conflicting instruments, such as bass guitar and kick drum or rhythm and lead electric guitars. The engineers were then asked to rate the performance of the tool being used using a slider with the Likert scale. The study showed that engineers tended to prefer panning as the most favoured tool having the highest satisfaction scores for solving the masking problem. This study could also be biased since panned stereo music generally scores higher satisfaction scores than mono music. This seems apparent given the mono mirrored EQ and frequency sharing tasks scored similar satisfaction scores.

2.2 Music Production Workflow

The production process can be defined in four stages: Recording and Overdubbing, Mix-down, Mastering and Publishing (Huber and Runstein, 2005, p. 10). The starting point of any production workflow is the recording process or tracking (Eargle, 2002, p. 326). An artist or group would usually go to a studio and record their performance onto a form of storage. Previously this meant recording onto tape or other analog formats, which introduced artefacts and colouration on their own (Eargle, 2002, p. 155).

Recording before editing technology was available, used to be one shot. So the recording engineer (or Director) was the only engineer involved as the recording phase was the only step involved (Hepworth-Sawyer and Golding, 2011, p. 16). As technologies and techniques developed, it became possible to split up the recording process into multiple segments. Initially this was through practices called over-dubbing, where individual instruments or sections were recorded onto the tape at different times, effectively layering the individual signals on top of each other. This meant 'Recording' engineers were crucial in the early days of commercial music production, but their role evolved into two distinct phases: Recording and Post-Production.

After the recording phase, the engineer begins the editing and production phases. Here, the engineer may take the different audio assets and re-construct them. To do this, multiple tape machines would be used to create

copies of individual sections to duplicate in other parts of the piece. Sections of the tape can also be cut out and pasted over other sections, effectively moving the fragment in the timeline of the piece (Eargle, 2002, p. 339). For example, if the vocalist made a mistake in one chorus, the engineer could copy another chorus to that place. This form of editing is obviously very destructive, and each copy or move introduces artefacts into the final product.

The mixing engineer will then take the edited tape reel and, using outboard analog hardware, down-mix (sum) the individual tracks into a single audio asset on a Master tape (Hepworth-Sawyer, 2009, pp. 54). During this process, the engineer will use the outboard processing equipment for either corrective or creative reasons. Corrective reasons would be, for example, removing the low frequencies from a guitar microphone which may be picking up bleed from the bass guitar. Creative reasons would be boosting the high frequencies of a piano to make it more prominent, or using a gated-reverb on a snare to increase the perceived impact. This Master tape is given to a Mastering engineer who will apply processing techniques to convert it from the Studio-grade tape to be ready for the various consumer-grade products, accounting for their individual characteristics (Ojanen et al., 2015).

So for one piece, it has to pass through 4 independent processes: Recording, Editing, Mixing and Mastering. Each of these roles are often conducted by different engineers with slightly different skill sets. Quite often a Recording engineer, then a Production or Studio engineer for the Editing and Mixing phase, followed by a Mastering engineer for the final checklist.

Today, the lines are blurred significantly due to the digitisation of the studio. Recording is completely transparent with the ability to record into a section of the timeline without interrupting or interfering with any other process. For example, the engineer may record the drums first, then the guitars and finally the vocalist with the ability to immediately jump through the song, dropping in the assets in-place. By recording in this way, the isolation between all the instruments is maximised since there is very little bleed between the two. It is also immediately comparable to previous performances and the other performers, so it is easier to identify if re-recording is required. Likewise, editing, recording and composition are blurred together since digital synthesisers and other post-recording tools have developed for the digital age.

The mixing phase is often entirely digitised today, with the Digital Audio Workstation software handling all of the processing and summation of the audio. Each track is sent through a set of digital audio processors to alter their sound. These often mimic the traditional outboard gear, but some can be completely creative and only exist in the digital domain.

2.3 Roles in the Studio

Music production is a complex industry with many significant personnel and roles required (Hracs, 2012). The two most obvious personnel in the studio are engineers and artists (Huber and Runstein, 2005, 16-18). These are two distinct groups of individuals with defined roles in the studio environment. However, there is a wider range of people involved in the whole production pipeline. Figure 2.1 shows how many different groups of people are involved in the production, distribution and promotion (Hracs, 2012).

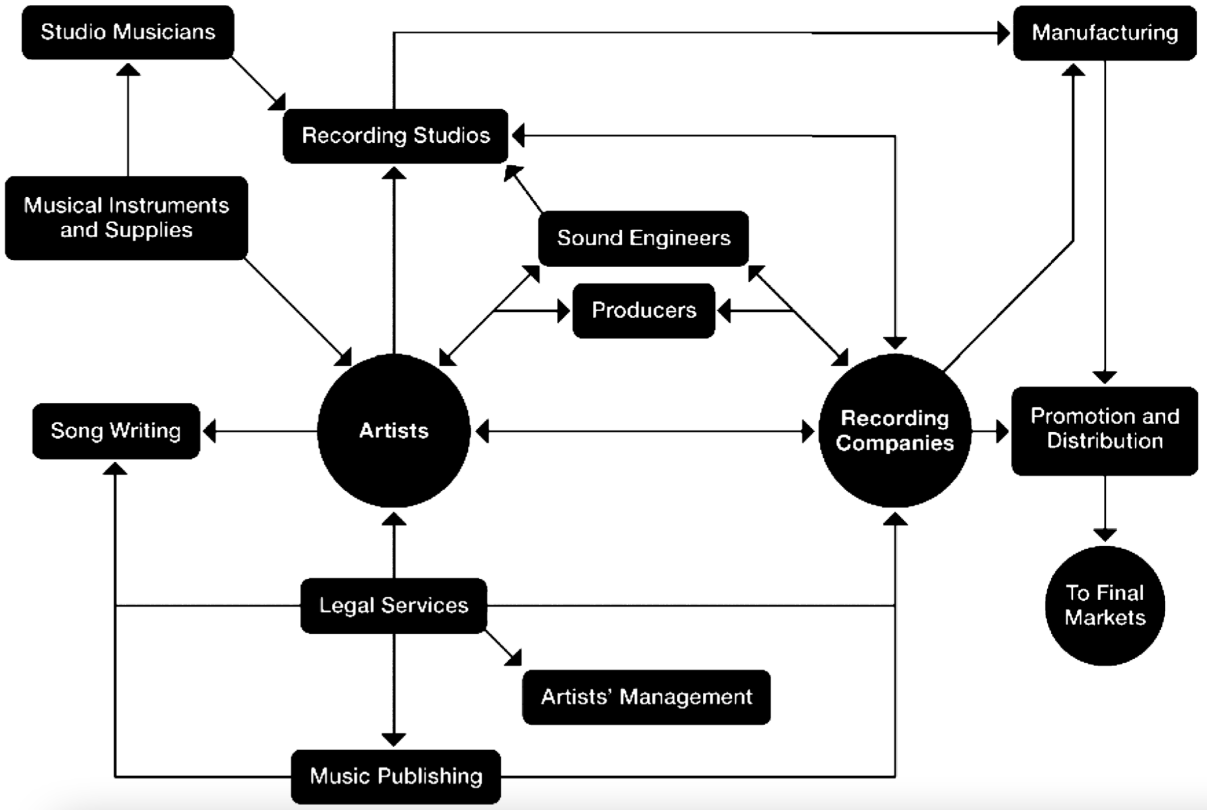


Figure 2.1: The traditional model of services in the music industry (Hracs, 2012)

The foundation of any musical recording comes from the artist's talent and composition skills. If the performance of the artist is not satisfactory, then further processing would be required to correct these mistakes. A strong performer improves the quality of the recording process as it moves through the production stages. The recording project itself may also need other performers, also known as studio or session musicians (Hepworth-Sawyer and Golding, 2011, p. 10). These performers may provide instrument playing skills the artist does not posses, and can perform the rest of the composition. Session musicians are usually professional and may be contracted by the studio or recording house.

The producer will often be a representative from a recording house, or an other vested interest (Eargle, 2002, 290-291). The producer is the medium between the artists/performers, engineers and the record label who is ultimately paying for the work. Hracs (2012) shows this in their traditional model of the studio environment, where producers are placed in between the Artist and Record Label roles. Their role is to reflect the recording house's artistic view as well as the financial and organisational aspects (Hepworth-Sawyer and Golding, 2011, p. 10). This includes working with the artist and engineer to create a recording plan, working out if any equipment, spaces or musicians would need to be contracted, and keeping the whole production workflow on time.

The engineer provides the knowledge and skill to operate the studio equipment both from a technical view and artistic view (Eargle, 2002, 291). The role of the engineer is to convert the artistic instructions from the artist and producers to technical instructions. The engineer will direct the recording session based on a prior discussion with the artistic team, deciding with the producers to re-take a section or not. They will choose

Figure 2.2: Three common Digital Audio Workstations in use today. From left to right: Apple Logic Pro X, Avid Pro Tools 12 and Ableton Live 9.

the microphones, acoustic spaces to use and the timescale. The engineering crew will also set up the physical aspects such as routing, setting appropriate recording levels and configuring equipment.

In larger studio houses, the engineers may be assisted by junior engineers (Eargle, 2002, 292). These juniors will quite often do the heavy lifting on the recording preparation, which is a time intensive task to complete. Juniors will take the instructions from the recording or session engineer on the microphone selection, placement, routing and equipment for the recording session. The junior engineer will then act on these instructions to ensure the senior engineer can focus on the recording session. A junior engineer would also be required to 'save' the session state by recording the equipment state, such as faders, pan dials and patch bays, in the studio so it can be replicated later. Before the days of recall systems (Burgess, 2014, pp. 101), engineers would spend hours in a studio just resetting everything so the mixing engineer could come and perform the mix phase (Hepworth-Sawyer and Golding, 2011, pp. 231).

Historically, the studio would be populated with many people to take control of various aspects of the recording process, with engineers having multiple technicians or junior engineers allowing them to delegate tasks (Hepworth-Sawyer and Golding, 2011, pp. 19-21). Over time, the producer and engineer roles have become more collaborative, as both learn each others craft. Engineers often spend a large amount of time working with artists, and part of their skill is to manage artists expectations and work with them to achieve their vision. Conversely, the producers are becoming more knowledgeable of equipment and practices and can better work with engineers on the technical limitations they may encounter. In more recent times, due to the progression away from recording in large and dedicated studio spaces and the modernisation of recording techniques, the concept of a 'bedroom producer' has emerged (Hepworth-Sawyer and Golding, 2011, pp. 19). These are artist-engineer hybrids, who use the significantly lower costs of music production tools to self-produce their own music (Bell, 2014). This blurs the boundaries of the production flow, constantly composing, recording, editing, mixing and evaluating their work. This is a by-product of learning using the non-destructive, flexible tools available, and driven by the wider democratisation of the music production industry.

2.4 Digital Audio Workstations

A **Digital Audio Workstation** (DAW) is the software package used to compose, edit, mix, and master audio content (Eargle, 2002, 201). In a standard DAW, the signal flows through channel strips, synonymous with

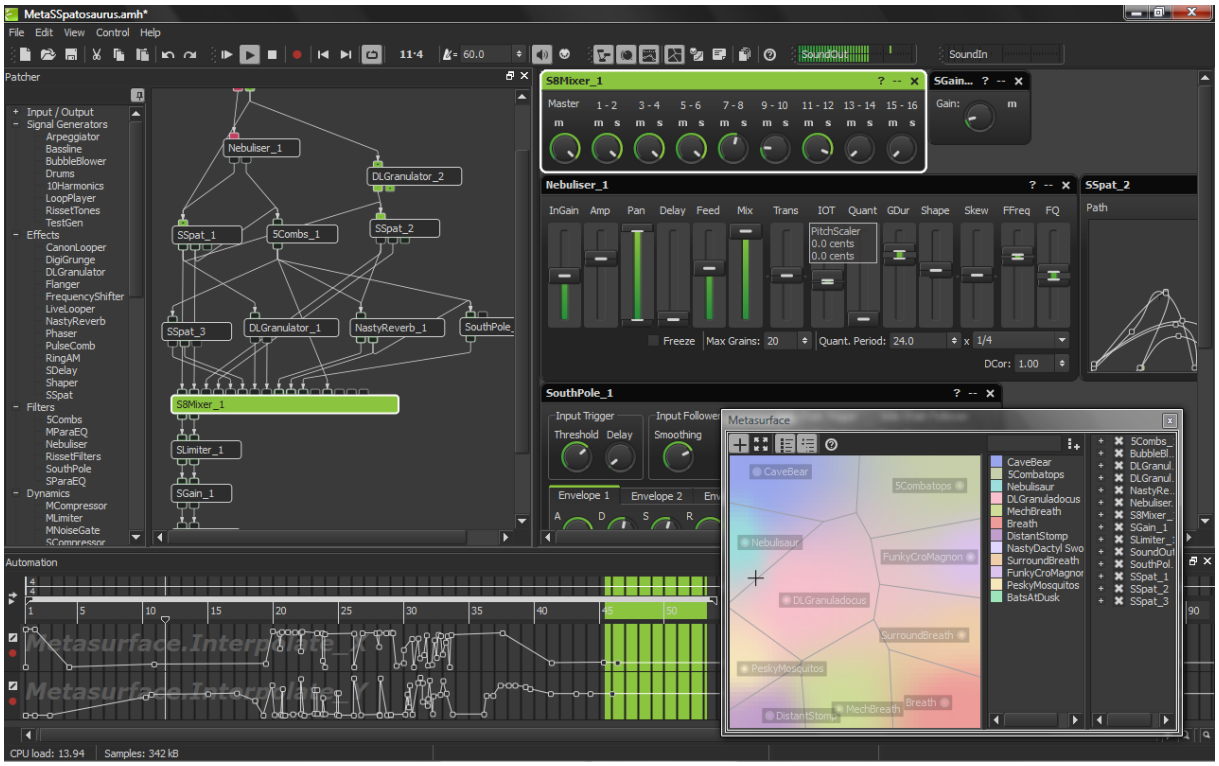
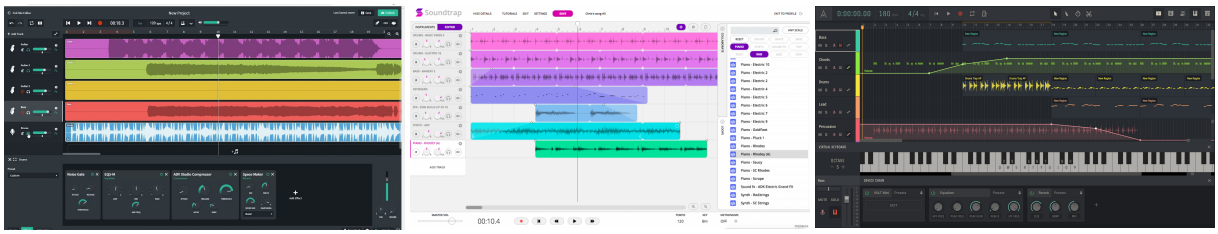


Figure 2.3: AudioMulch user interface, showing the mixing interface and non-traditional routing interface (AudioMulch)

large-format mixing desks (Marrington et al., 2017). It generally has a linear timeline, where audio clips can be arranged across multiple tracks over time analogous to the concept of tape machines, where the playback system will progress through the timeline rendering the audio as it occurs. The signal is routed through a serial bank of digital audio effects, then through pan, fader and output stages, inheriting a large amount of the behaviours from traditional studio workflows using hardware (Marrington et al., 2017). As an example, most DAWs include a mixer view, with an skeuomorphic channel strip, complete with bottom LED faders (Marrington et al., 2017). In figure 2.2, Logic Pro X shows a complete mixer along the bottom of the screen. All of the example DAWs shown in figure 2.2 also have a timeline view.

Non-linear processing pathways and complex routing is where digital systems can overtake traditional studio hardware. This is because the signal can be copied, processed and routed using simple mouse and keyboard interactions. This would still be possible in the physical studio using outboard gear, but it would be time-consuming to set up and would be limited to the amount of patch pathways available in the studio. Digital platforms enable this form of flexibility without the physical limitations. Traditional *effects-in-line* interface, following the skeuomorphic design principles are not universal, with AudioMulch (AudioMulch) being an example. This interface, shown in figure 2.3 shows alternative routing methods which are normally not possible in a traditional DAW interface, but more closely resemble what a studio environment could achieve.

Recently there has been a shift from desktop based audio production software to online production suites. These can exist either as hybrid solutions, where the software is still local but synchronises changes to an



(a) Bandlab (b) Soundtrap (c) Amped Studio

Figure 2.4: Three popular web-based DAWS from three vendors

online database (Stickland et al., 2018), or as fully web-first solutions (Lind and MacPherson, 2017). Web based DAWs can provide the functionality of dedicated desktop software, but leverage the possibilities of the web as a platform through peer-to-peer communications and anywhere access (Lind and MacPherson, 2017). These DAWs usually focus far more on collaboration and ease-of-use over outright functionality (Stickland et al., 2019). Figure 2.4 shows three commercial online DAWs: BandLab, Sountrap and Amped Studio. Each of these platforms focus on simplifying the production workflow significantly for end users. It is immediately apparent their user interface is cleaner compared to the traditional studios in figure 2.2. Most do away with the mixing interface entirely and have a single view.

2.5 Automatic and Intelligent Music Production

A wide range of systems for automatic mixing have been developed to assist engineers at various stages of the production life cycle (De Man et al., 2017). These include systems for panning (Mansbridge et al., 2012a), equalisation (Perez-Gonzalez and Reiss, 2009) or audio editing (Montecchio and Cont, 2011). Out of these systems, the vast majority have focused on Fader and volume control (De Man et al., 2017; Mansbridge et al., 2012b; De Man and Reiss, 2013a). Even for a simple mix problem, where only the faders are used to control the volume, this is still an n-dimensional problem, where n is the number of tracks (Terrell et al., 2014). Because each adjustment on any one track has a perceptual impact on the resulting mix, this becomes a complex problem to optimise as the mixing process becomes iterative (Izhaki, 2012). The engineer will often make adjustments to the audio stream based on a mixing decision, then evaluate that decision each time. Automating the parts mixing process will allow engineers to focus their time and energy on the creative aspects of mixing, rather than corrective aspects.

These automatic mixing systems can be broadly divided into two formats: real-time and offline. Automatic mixing systems will take the audio signals, analyse them, and return control signals to the digital audio workstation (Reiss, 2011). A real-time system will use some form of signal analysis such as feature extraction, where real-time signal flow information from the incoming audio frames is used as the input to an algorithm which outputs a control signal (Reiss, 2011). This allows the system to respond to incoming audio events and is highly suited to environments where the audio stream may not be available ahead of time, such as in broadcast environments. It is also suited to mixing processes which can be simplified to a known process or a set of defined rules (De Man and Reiss, 2013a).

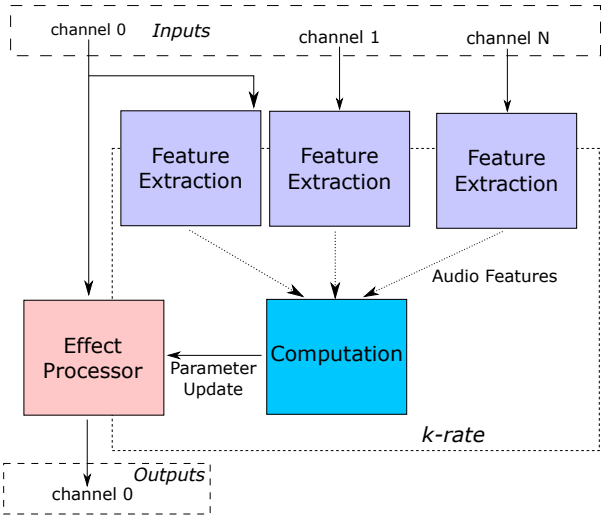


Figure 2.5: Block diagram of the general Cross-Adaptive Processor for audio (Reiss, 2011).

Offline automatic mixing systems allow for the analysis of the entire signal rather than just a set of historical frames. This gives algorithms the ability to prepare for future audio events and set control parameters accordingly, but it limits the number of potential applications. In this environment, algorithmic processing can employ machine learning techniques such as neural networks (Stasis et al., 2016). This is because the system can examine the entire audio file allowing it to optimise best to the signals for the task being automated. One such example would be time-alignment, were the entire audio signal needs to be warped and stretched to align two signals of differing and varying tempos (Hockman et al., 2008).

The term 'Adaptive Audio Effects' is used to specify an audio processor which manipulates its own parameters based on the incoming audio signal and a set of predefined rules (Verfaille et al., 2006). A processor which takes the incoming audio solely from its own stream for processing is called auto-adaptive (Reiss, 2011). The processor extracts features or other high-level information from the stream to generate a control signal to update the parameter it is targeted to automate. For example, a Dynamic Range Compressor can be considered an auto-adaptive effect, as it uses the incoming audio frames to extract loudness information using a peak or RMS calculation. This information is sent through a gain computer to determine the desired amount of gain reduction or boost to apply to the signal.

An effect is cross-adaptive when it takes in audio or features from neighbouring tracks or sources in the session to control its own plugin parameters, an overview of which is shown in Figure 2.5. Reiss (2011). In a multi-track environment, such as a DAW, as plugin could access each incoming audio track. Each track is processed using feature extraction to create a set of descriptive semantic values, such as loudness, tonality or skewness. These also convert the signal from audio-rate or a-rate into a lower sampling rate or k-rate. This is achieved by the incoming feature taking a window of the input streams, such as a chunk of 1024 samples and producing one scalar variable. In this example, k-rate would be 1,024th the audio sampling rate. These k-rate signals are then passed into a computation function which emits parameter updates to the desired effects processor. A commonly used example of this form of effect in dynamic range compression when using a side-chained input, such that the rate of compression on one track is dictated by the loudness function applied to a different track.

This allows a plugin to take multiple inputs and control them together with minimal user input, such as for automatic panning (Mansbridge et al., 2012a). An early example of a complete cross-adaptive audio processor is Dugan's automatic microphone mixer for conferences (Dugan, 1975). The device was designed to assist engineers when handling multiple microphones for several speakers, such as for a panel of speakers or during a conference. Since only one member would be speaking at a time, it would be beneficial for only the current speaking microphone to be active and all others muted. This would improve the signal quality by removing any noise or feedback collected by the other microphones, which would have no desirable content. The system instead used a gate-style device, whereby if a microphone feed went above a given threshold it would become active and disable the other microphone feeds. This had a significant improvement on the quality of the audio stream without putting excessive strain on the audio engineer. This system was improved upon by allowing the system to analyse the inputs then applying gain dynamically, allowing for use in a wider range of applications, such as for teleconferencing Julstrom and Tichy (1983).

2.5.1 Model based mixing

More recently the automatic mixing space for level controls has shifted towards using higher level features such as loudness level (Mansbridge et al., 2012b). These systems would try to make sure the relative loudness level between each track in a mix was equalised to the total mix by calculating the LUFS of each track using the EBU R128 loudness recommended filter (International Telecommunication Union, 2011). It uses ballistics to dampen the impact of the control signals, so short and sharp sounds would not be boosted unnecessarily. This metric has been shown to be a suitable measurement for the perceptual loudness of a sound source, but its aim is for broadcast production. This means it may not be directly suitable for streams and sources directly from the recording stage of the mix process. Adapted loudness models have been used in more recent publications, although the premise and processes are similar (Wichern et al., 2015; Fenton, 2018).

Auditory models have also been used for automatic mixing in real-time, which demonstrates that the models provide an acceptable metric for mixing; however, the computational complexity of the models rules them out of most real-time applications (Ward et al., 2012). This approach is one of the few implementations of automatic mixing which aims to use a model of human hearing. This was also done for real-time loudness control for broadcast environments using a real-time implementation of the loudness models (Lund, 2005).

The process has also been used for other parts of the mixing process, not just fader control. Some of the most prominent are panning (Mansbridge et al., 2012a) and equalisation (Perez-Gonzalez and Reiss, 2009). One system in particular uses spectral features for equaliser controls with the aim to balance the spectral energy of the signal (Perez-Gonzalez and Reiss, 2009). Their results show listeners are able to easily discern between mono and stereo mixes, and when presented with both, tend to prefer stereo versions of the same stimuli. The system also outperforms non-expert mixers, but does not out-perform an 'expert' mix engineer. This indicates a suitable sweet-spot for Intelligent Music Production systems, where it can advise and guide the engineer to make better decisions at a faster rate. The system works by passing each incoming stream through a filter bank to perform spectral decomposition in order to find the most dominant frequency for each incoming stream.

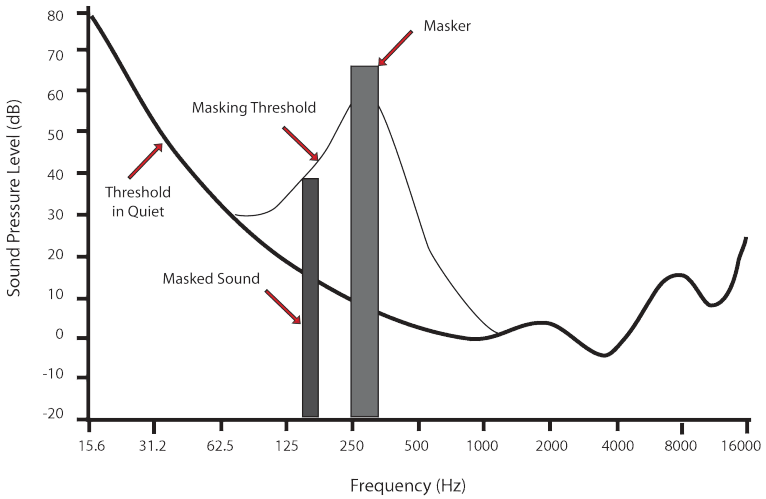


Figure 2.6: Auditory masking, showing the threshold curve for human hearing. When a dominant sound source occurs, the masker, it raises the threshold of neighbouring frequencies, causing a quieter signal to be masked if it is below this elevated threshold.

Once a dominant frequency range has been identified, the audio streams are panned according to a set of pre-defined rules.

The first mixing rule described by (Perez-Gonzalez and Reiss, 2009) is based on the assumption that all bass and low frequency content should be kept as central as possible. This is implemented such that any signal with a dominant frequency of 200Hz or lower has the same gain applied to the left and right channels. The second rule then tries to space each incoming stream, with similar energies, at equidistant sides of the panning space. So if two signals have the same dominant signal, they should end up hard left and right, scaled by the frequency of the dominant band. This means as the frequency gets higher, the more extreme the panning. As more signals are added, the search algorithm gets more complex to implement and ensure equidistant.

This system approximates the affect of internal auditory masking, by trying to ensure two similar acoustic sources are not playing through the same ear (Wakefield and Dewey, 2015). This is because the masking phenomenon is tightly related to a single ear with overlapping frequencies. Therefore if the conflicting signals are being played to the separate ear, less masking occurs. An auto-panning system utilising auditory models for masking instead of the rules-based approach was developed Tom et al. (2019). Using the same filter banks as the MPEG standards to identify masked frequencies, the system would then pan based purely on the impact of the masking performance. The system was also informed with more rules based on the perceptual model, with an emphasis on de-masking the mid-band frequencies. This system iterates over time, indicating that panning as a real-time model is more complex, and applying the processing offline or performing pre-analysis would improve these systems even further.

Auditory Masking

Auditory masking is a well researched phenomenon whereby an audio signal will prevent the auditory system from perceiving other frequencies in the signal (Greenwood, 1961; Wegel and Lane, 1924). The human ear bins

the frequency information into discrete receptors, when these are activated by an incoming audio stimuli it can raise the threshold of neighbouring bands. This means a stimuli that would have been heard may no longer be strong enough to trigger a response, effectively masking the signal from perception. The human ear has a known sensitivity to different audio signals, meaning the auditory system will perceive frequency information at different thresholds. The human ear does not perceive the loudness of audio signals with a flat frequency response, but in fact the response is curved (Fletcher and Munson, 1933). Furthermore, this loudness curve changes for different listening levels, meaning the relationship between two frequencies at 0dB may not be the same relationship at 10dB. Their experiment required participants to listen to a 1kHz tone at a given sound pressure level, and then adjust the amplitude of a second tone until the listener perceived them to match. This discovery led to the publication of the Fletcher-Munsen curves. These curves shows that there is increased sensitivity around the 2kHz to 4kHz range, and lower sensitivity to lower frequencies. As the sound pressure level approaches 100dBSPL, the curves mostly flatten out.

Following the equal loudness contours, Fletcher (1940) showed the impact of auditory masking. This phenomenon is caused by the dynamic sensitivity of the human auditory system to alter as the sound pressure level changes. This is supplemented by the auditory system only taking a subset of auditory information in bands. These bands each take a different range of frequencies. Each of the bands can change its sensitivity to incoming auditory stimuli based on the loudness perceived. This means that if there is a loud signal of similar frequency to another, it can mask this quieter signal so it is not even perceived by the listener.

Since it's documentation, multiple models have tried to simulate the effect of auditory masking. A model was presented to calculate the loudness perceived by a listener from any given audio source (Zwicker and Scharf, 1965). This incorporated masking patterns to provide a more realistic measurement. The masking patterns were simulated by using Bark bands to estimate the different audio receptor bands in the human auditory system. This model was updated when it was shown that auditory masking thresholds have a tail effect, whereby the masking effect is persistent even after the masker has finished (Zwicker, 1965) . This shows the auditory system takes time to lower the perceptual sensitivity to loud audio sources, giving time for continued masking after the high SPL event has passed. Masking models have been created to enable the calculation of the perceived masking between two signals (Gilkey and Robinson, 1986; Moore et al., 1997). These have severe computational cost but there are suitable compromises to reduce this cost (Ward et al., 2013).

$$I = kx^2 \tag{2.2}$$

$$I = \frac{1}{T} k \int_0^T x^2 \left(t\right)\, dt \tag{2.3}$$

$$L_I = 10 \log_1 0 \left(\frac{I}{I_{ref}} \right) \tag{2.4}$$

The Glassberg and Moore model will simulate the natural filtering of the outer, middle and inner ear to provide an excitation model of the sound over time and frequency (Moore et al., 1997). The model outputs multiple measurements for experienced audio stimuli to summarise the excitation pattern of the human auditory system. The first stage of the model calculates the transfer of energy through the outer and middle ear into the cochlea (inner ear) (Moore et al., 1997). This is represented by a fixed linear filter with frequency dependent gain. This gain curve is designed to match the filtering of the physical ear and follows the Fletcher Munsen curves of sensitivity (Fletcher and Munson, 1933).

The output of the function gives the intensity of the loudness at a specific frequency in dB. Loudness I is represented as a perceived intensity of a sound pressure wave x, and can be expressed in terms of sound pressure squared against the acoustic impedance of the air k as in Equation 2.2 (Simpson et al., 2013). To calculate for a given pressure wave over time, the pressure is integrated to simulate the integration effect of the auditory system, described in Equation 2.3 (Simpson et al., 2013). Intensity can then be described as a ratio with respect to some reference level, usually a level of human hearing in decibels. This is known as the intensity level L_I in Equation 2.4. By passing L_I through the auditory filter, L_{I1} is calculated. This can then be converted into an excitation level L_E by normalising to a reference 1kHz sinusoidal signal at 0dB SPL.

$$ERB = 24.7\,(0.00437f + 1) \tag{2.5}$$

$$n = 21.4 \log_{10}(0.00437 f_c + 1) \tag{2.6}$$

$$f_c = \frac{10^{\left(\frac{n}{21.4}\right)} - 1}{0.00437} \tag{2.7}$$

With the excitation level L_E calculated, the next stage is to pass the signal through a set of filter banks to calculate the energy of the auditory bands. This is done by passing the excitation response through a set of Equivalent Rectangular Bandwidth (ERB) spaced auditory filters. The bands are calculated in such a way that nearby frequencies, which fall in the same band, will be combined together to calculate the energy in that band. The ERB calculation gives a mapping from frequency f (Hz) to the ERB (Hz) in equation 2.5 (Moore et al., 1997). This can be converted to equation 2.6 to calculate the ERB band n (Moore et al., 1997). The equation 2.6 can be re-ordered to equation 2.7 so that the centre frequency f_c can be calculated for the ERB bands. These are calculated for the ERB intervals which match the known range of the basilar membrane of 50Hz to 15kHz (Moore et al., 1997).

$$w(g) = (1 + pg)\exp^{-pg} \, g = \frac{|f - f_c|}{f_c} p = \frac{4 f_c}{ERB} \tag{2.8}$$

$$g = \frac{|f - f_c|}{f_c} \tag{2.9}$$

$$p = \frac{4f_c}{ERB} \tag{2.10}$$

$$w(g) = \begin{cases} (1 + p_l g) \exp^{-p_l g}, & f \leqslant f_c \\ (1 + p_u g) \exp^{-p_u g}, & f > f_c \end{cases} \tag{2.11}$$

These filters are generally applied using a rounded-exponential or 'roex' filter $w(g)$ as defined in equation 2.8, where g is the normalised distance from the centre frequency f_c, shown in equation 2.9, and p (equation 2.10) is a constant related to $4f_c$ over the ERB value in 2.5 (Patterson et al., 1982). This is the definition for a symmetrical auditory filter, however the auditory filter should be asymmetrical (Moore et al., 1997). Therefore the filter is broken down into two cases as shown in equation 2.11 for the lower and upper bands depending on whether f is above or below the centre frequency f_c.

$$E(n) = w(g(n))L_E \tag{2.12}$$

$$N_{masking} = \frac{N\left(E_t + E_m\right)}{N\left(E_t\right) + N\left(E_m\right)} \tag{2.13}$$

For the given ERB number band n the excitation pattern can be extracted, which defines the output from each the ERB-spaced auditory filters. This can be defined as Equation 2.12 (Simpson et al., 2013). This gives the excitation pattern which can be converted into a specific loudness measure for the n-th auditory filter. With the excitation patterns calculated it is possible to then look at the masking effects. Two excitation patterns are calculated for the above measure, E_t the target and E_m the masker. The basic principle is to compare the sum of the loudness of each pattern alone against the linear sum of the two loudness patterns, as shown in Equation 2.13 (Simpson et al., 2013).

$$N_{masking}(t) = \frac{N\left(E_t(t) + E_m(t)\right)}{N\left(E_t(t)\right) + N\left(E_m(t)\right)} \tag{2.14}$$

By extending these calculations over time-varying sounds, it is possible to show the excitation pattern changes as time progresses. The principle is the same as above, except that each step needs to have a time factor applied, and for the time to be integrated at the end to get a final number, should this be the desired output. With the temporal pattern extracted as well, the masking of the term. Equation 2.14 gives the temporal masking effect definition, where time is now a factor. By taking the integral of the output it would be possible to get a masking energy calculation as a scalar value.

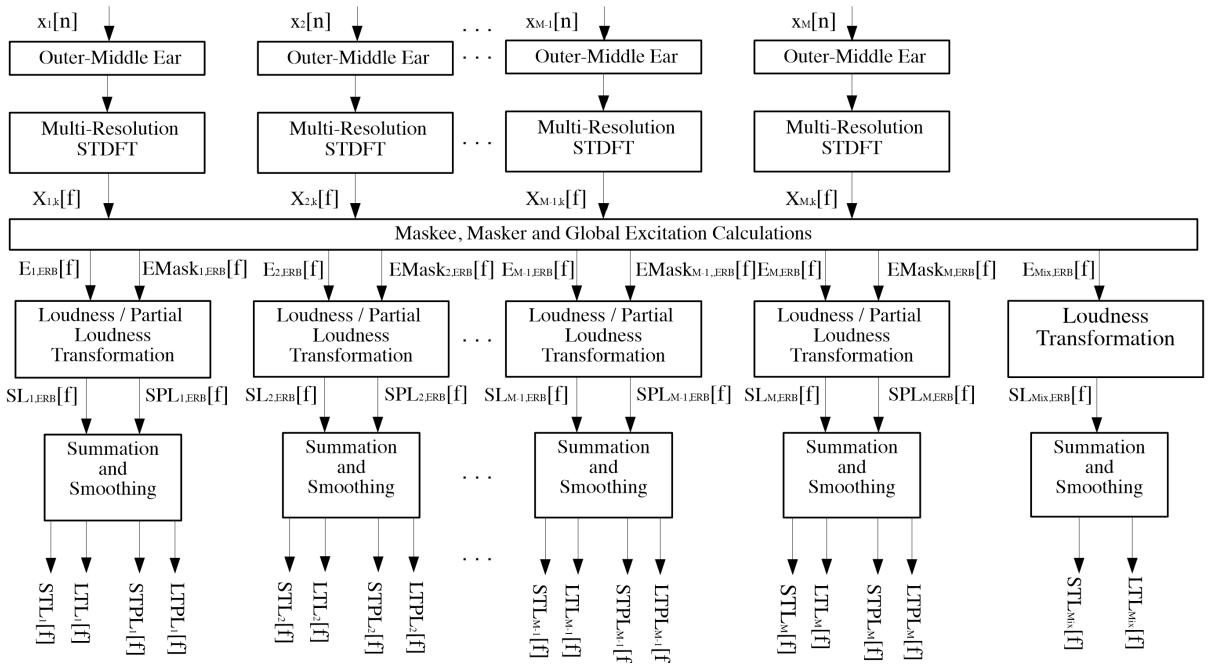


Figure 2.7: Diagram of the loudness model developed by Ward et al. (2012) showing the output for each track given after processing all incoming audio signals.

Audio coding technologies have used auditory masking models to remove redundant audio information that would otherwise not be perceived by the end listener (Brandenburg and Bosi, 1997). The MPEG-Layer III coding standard used this to its advantage to remove information which would most likely never be perceived by the end listener. As auditory models have improved, so have the accuracy and performance of the audio coding standards.

(Ward et al., 2012) used an auditory masking model to create a real-time, automatic-mixing process based on the loudness models by (Moore et al., 1997), shown in 2.7. This was also done for real-time loudness control for broadcast environments (Lund, 2005). This system would take the incoming audio streams and calculate the total and specific loudness for each track. This meant each track not only affected the final mix, but its impact on other tracks through masking could be evaluated. Whilst no formal evaluation of the mixing performance of this model was performed, it showed significant improvement in the ability to adapt to noisy, broadband signals compared with steady-state signals. This should lead to significantly increased intelligibility of the audio content compared to systems which do not account for the masking effects.

Using the loudness toolbox developed by Ward et al. (2012) enables the analysis of the partial loudness and specific loudness, using the Glassberg-Moore model, for a given audio signal. This can be used to show how much of a signal is masked in the final mix, called the masked to unmasked ratio (Aichinger et al., 2011). This ratio gives a score of 0 if all of the signal is masked, and 1 if all of the signal is present. The ratio also responds to the amount of energy present in each band, so if a band is masked but it has low energy, it will not have a significant negative effect on the score. Likewise, an important band with high energy being masked will have a significant impact on the score.

$$MUR[\%] = \frac{max\left(N_{masked}, 0.003\right)}{max\left(N_{unmasked}, 0.003\right)} \tag{2.15}$$

This formula in equation 2.15 takes the total loudness of the masked signal, using the partial loudness from an auditory model, and divides it by the specific loudness. At each given band, a score of 0 would indicate there was no signal present in the partial loudness output (the output filtered by masking effects). Conversely a score of 1 would represent that the bands had the same perceptual energy. As an example, if a mix has three instruments: Guitar, Drums and Bass, and the the goal is to determine the amount the guitar is masked by, then the denominator is the sum of all three instruments and numerator is the partial loudness of the guitar. To prevent situations where a division by 0 is possible, the two are bounded to 0.003 sones, the threshold of human hearing (Aichinger et al., 2011).

2.5.2 Machine Learning

Instead of using heuristic algorithms (Mansbridge et al., 2012b; Perez-Gonzalez and Reiss, 2009) or rule-based systems (De Man and Reiss, 2013a), machine learning can be used instead. This uses a model-based approach instead of a event based approach, an example is Wilson and Fazenda (2017) which used Monte-Carlo simulations to probe the gain and pan positions to find suitable mixing parameters which best matched the mix to a given target mix. Generally, the system operates the same way by taking blocks of audio, extracting features and matching to a target. The difference is the system will learn the features it believes are most important, rather than being set by the researcher. A similar methodology was used by Eichas et al. (2017) to find suitable parameters for a guitar amplifier.

Machine learning models are often used in intelligent music production as an alternative to empirical modelling. With machine learning, generalised models like neural networks are used, and the latent properties of the network are then used to make decisions on unseen data. Three major forms of learning methodologies are commonly used: supervised, semi-supervised and unsupervised (Ayodele, 2010).

Supervised Learning

Supervised learning is when the computer is required to approximate the behaviour of an unknown function, which maps an input vector to an output vector. It achieves this by comparing a set of known input-output combinations against the current approximation function, and adjusts it until the function successfully maps onto this function. Algorithms can be quite simple, in the form of linear regression models which aim to create a set of weights for a given vectored inputs to map to a given output (Murphy, 2012, p. 217). The system then allows a new input vector to be passed through to predict a suitable output state.

Supervised learning methods work when there is an unknown mapping between dependent and independent variables. Linear regression or least squares models form one such methodology for extracting this mapping function, so long as there is a linear relationship between the dependent (output) variable and the vector of independent (input) variables. Equation 2.16 shows the linear regression problem, where x is the independent variable. The task is to find a suitable parameter vector, β, such that it can suitably predict the dependent variable y.

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i = \mathbf{X}\beta + \epsilon \tag{2.16}$$

Finding suitable coefficient values requires an iterative approach to training the model. The job of the training in this case is to minimise the error between the predicted and the actual outputs. This is done by individually evaluating the error of a given parameter vector. Given a candidate vector for β as $\beta\prime$, the model can be tested by measuring the distance between the observed output and the predicted output. This distance between the observed data point and the hyperplane given by $x^T\beta\prime$ is called the residual. By calculating the sum of these errors, a measurement of the overall fit for the given parameter vector $\beta\prime$ can be calculate.

$$S(\beta) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} X_{ij}\beta_i \right)^2 = |\mathbf{y} - \mathbf{X}\beta|^2 \tag{2.17}$$

Equation 2.17 gives the Error Sum of Squares to measure the overall accuracy of the parameter vector. The difference between the observed and predicted outputs is taken and then squared. This squaring makes negative values positive, ensuring the distance is taken, as well as penalising higher error counts.

One method for estimating the the parameter coefficient β is to use Ordinary Least Squares. The equation 2.18 gives a prediction for β by taking the Gram matrix ($N = X^{\mathbf{T}}X$) against the moment matrix ($X^{\mathbf{T}}y$). By re-arranging equation 2.18 to 2.19, the estimator for β can be found. This method can only be used in scenarios where strict properties of $\mathbf{x}$ can be applied, with the primary requirement being the linear independence of the p column of $\mathbf{x}$. If this is not true, then the Gram matrix would have no inverse and therefore this model will not work.

$$\left(X^{\mathbf{T}}X\right)\hat{\beta} = X^{\mathbf{T}}y \tag{2.18}$$

$$\hat{\beta} = \left(X^{\mathbf{T}}X\right)^{-1} X^{\mathbf{T}}y \tag{2.19}$$

Linear regression models allow the estimation of a linear relationship between a set of independent variables to be discovered. In audio, a common use case is for de-noising of the audio signal. Noise is a random component or error added to the signal through the recording, production or distribution processes. It can be removed by decomposing the signal into a set of vectors containing the discrete cosine transform of the signal (Févotte et al., 2007). The non-linear related parts of the signal are extracted through this system and are exposed through the residual component, containing the noise of the signal.

However there are situations where a linear relationship cannot be uncovered. In this case, neural networks provide a suitable method for uncovering the relationship between the data (Dayhoff, 1990). A neural network is made up of layers of neurons, each neuron takes as its inputs the previous layer and outputs to the next layer (Dayhoff, 1990). Simple neurons will receive a binary input vector and output a binary emission, but more

complex neurons will have memory and non-linear functions themselves. This makes neural networks extremely powerful, but highly expensive to train as each neuron is trained as part of the network.

Neural networks have become more prevalent as computing power using general purpose GPU's has considerably reduced training times. Neural networks are traditionally classification models, aimed at taking a set of inputs and to output classification that they best represent (Dayhoff, 1990). Instrument classification is a well researched field of work, with multiple systems being developed (Gómez et al., 2018; Newton and Smith, 2012). More recently, drum transcription has also been developed to detect the onset of drum hits from single mix downs, where traditional transient detection with a multi-mic setup is not available (Stables et al., 2016). With transformation networks and more advanced neural network designs, they can output more complex items than just binary classifications. This gives way to applications in whole mix optimisation where the algorithm not only learns how to mix but can suggest multiple mix parameters from a set of raw incoming audio tracks (Steinmetz et al., 2021). The performance of such systems is heavily dependent on the training data, and results in the same problem of defining what is a good target mix. Without access to thousands of unprocessed raw audio mixes, this goal is always going to have high variability of results (Steinmetz et al., 2021). Transformation neural networks also allows an automatic mixing system where the output is not a set of parameter controls but the mixed audio file. This would make it possible to pass in an audio file and a target file to match the 'style' on to, usually for drum timings. This can automatically alter the performance of the piece to match a different style, though this is approaching automatic editing techniques (Tomczak et al., 2018)

Unsupervised Learning

Unsupervised learning methods instead provide a solution to a hypothesis when there is no training data available. These usually include clustering, where the boundaries between the clusters are unknown. Anomaly detection is also a common use case, whereby a system learns what is a normal event or sequence of events to identify anomalous data in a system, without having to experience the anomalous data. Cluster analysis was first used by Driver and Kroeber (1932) to determine relationships between Polynesian populations and to classify groups of cultures which shared common traits.

One of the most simple methods for cluster analysis is K-means, with the algorithm standardised by Lloyd (1982). This equation operates on two steps, the assignment step and update step. Given k number of clusters to find, the cluster centroids are first initialised randomly in the parameter space. The system then iterates through each data point, and assigns them to a given cluster by assigning them to the cluster centroid with the smallest distance. The distance measurement is usually the least squared Euclidean distance. Once the clusters have assigned members, the cluster centroid is updated to be the mean position of the assigned observations. In this case, unsupervised learning will allow a rudimentary classifying algorithm to be established as a new, unknown data point, can be assigned to a cluster quickly and therefore classified. This algorithm has its limitations, with the number of clusters to identify having to be known and the risk of over-fitting the data points.

Another form of unsupervised learning are solvers. This is a collection of search algorithms, which can either work on given data sources and structures, such as web page indexes, but they can also search virtual

parameterised spaces. These algorithms are useful in environments where the search space itself is not known and cannot be predicted or extracted by the system, such as through feature extraction, because the space itself is non-deterministic.

Early forms of solvers are brute-force, exhaustive search, or naive solvers. Brute-force solvers operate by testing every possible combination of the proposed system until a valid solution is found, or no solution. The brute force algorithm works best on binary solutions, where either the solution passes a test or fails, which have many possible forms of solutions. One such problem where brute-force works is the Eight Queens problem (Rouse Ball, 1960, pp. 165–171). This problem asks to find a solution to place eight queens on a chess board such that no queen can attack another. Because the number of positions on an 8x8 chess board is over 4.4 billion, but only 92 possible solutions, the process itself is quite computationally expensive. But applying the rules of chess into the game, such as simple rules such as no queen on the same row or column, it is possible to quickly iterate through a set of suitable positions starting with one queen at a time until a solution is found. However, there is no metric or cost solution which states how close the solution is, only that it is right or wrong. The problem is still a search problem in that the suitable positions of the queen are unknown and therefore must be suitably searched.

$$a_{n+1} = a_n - \gamma \bigtriangledown F(a_n) \tag{2.20}$$

Gradient Descent, is one of the early methods for finding the solution to an unknown solver space (Cauchy et al., 1847). The algorithm works by iteration to walk towards a local minimum. Equation 2.20 gives the mathematical definition of the iterative process. To find $a_n + 1$ given the current location of a_n, the system should maximally reduce the system by moving in the direction of the steepest gradient. For a learning rate γ being both real and positive, then the next step is some multiple towards that minimum. This is only possible in the situations where the solution is not only known but differentiable.

An important component in all machine learning processes is to have a representative cost function. This function takes the input vector and should return an accurate description of the output in a numerical form, as to how suitable the solution is for solving the problem. These functions are typically defined as the absolute difference between some observed values in the data set, and an optimised hypothesis. A simple example of a cost function in audio processing might be the loudness level of a given audio track in response to altering the inputs of a dynamic range compressor. It would be possible to write a cost function which takes the input as the parameters for the dynamic range compressor and the given audio file, and return a number stating how close the result is to the target loudness level. A well defined cost-function will improve the performance of the test function as it will allow the function to accurately assess the performance of the given solution.

2.5.3 Evolutionary Computing

Genetic Algorithms (GA) or Evolutionary Computing (EC) is a form of artificial intelligence based upon biological concepts, such as mutation, mating and generations (Back, 1996). Genetic algorithms have been used for a wide range of applications in machine learning, include antenna design (Lohn et al., 2004), line balancing

in factories (Watanabe et al., 1995), and as a training method for neural networks (Maniezzo, 1994). They are most effectively used for complex design processes where traditional machine learning approaches are not suitable (Katoch et al., 2021). Instead of a single starting point, multiple are taken at once across the entire space. The number of samples is referred to as the population size, and the larger the population size the more samples are taken. Each sample member, referred to as a chromosome, is then passed through a cost function to determine the suitability of the solution. This cost often needs to be converted to a fitness score, ranking from 0 to 1, where 1 is the most optimal solution. Once each chromosome has been given a fitness score, they are sorted by this score and the top set of chromosomes are kept. The method of this selection stage is discussed in 2.5.3.

Once the best performing chromosomes are selected, the next iteration of chromosomes must be made. This is done by crossover, and is discussed in 2.5.3. An important step of the search process is mutation, where individual chromosomes may experience a randomisation event, analogous to biological mutations when copying DNA, this is discussed in 2.5.3 as well.

Selection

When all the chromosomes in a generation have been given their cost function, the next phase is to select the bet performing solutions such that the next generation can be populated from them (Katoch et al., 2021). The first parameter is to determine how many chromosomes to take forward for each stage as 'parents' to create the next generation. This ratio affects the learning rate of the system, a high rate means there is little variation on each iteration with the majority of the chromosomes being made up of previous generations. Too low and the generations may not converge quickly as each generation holds multiple newly randomised entries.

Several methods exist for selecting the chromosomes from the population. The established ones are roulette wheel, rank, tournament and truncation (Schmitt, 2001). Roulette Wheel selection operates by giving each chromosome a relative number between 0 and 1 relative to its fitness performance (Lipowski and Lipowska, 2012). Equation 2.21 shows how the i-th chromosome h_i is given a probability p_i by taking the fitness function result $f(h_i)$ over the sum of all the fitness results for the generation.

$$p_i = \frac{f(h_i)}{\sum_{j=0}^{N} f(h_j)} \tag{2.21}$$

A chromosome which performed well will be given a larger range than a chromosome which performed poorly. If all chromosomes performed equally, then each would have the same range. They are then sorted together such that the first chromosome has the range from 0 to p_0, and the last would have p_i to 1. A random number is then generated and whichever chromosome's number range contains it is selected. This quick to process but does mean the same chromosome can be selected multiple times.

Rank selection is derived from the Roulette selection, but instead of the fitness function result being used directly, the rank of the chromosomes is used instead. This solves a problem where a small but high performing

set of chromosomes are selected multiple times leading to convergence too quickly, which may be in a local minima. By using the rank order over the rank sum, it allows each chromosome to have a more uniform range.

Tournament selection (Brindle, 1981) selects two random chromosomes from the results. If first selects individuals from the population at random to fill a given tournament size. This can be two or more individuals. The best individual from the tournament is selected with the probability p as defined equation 2.21. The tournament stage is repeated until the desired number of chromosomes are selected. If the tournament size is one, then it is akin to random selection of the population. Larger tournaments also tend towards truncated selection, as the best performing chromosomes will always be picked.

The truncation method of selection is the crudest form and only preserves the top performing chromosomes to create the new generation. Whilst it is simplistic in that the chromosomes only have to be ranked and then the top few used. However a combination of truncation selection and the randomised selection methods can be used. Elitism selection is an improved version of the Roulette wheel selection above. Rather than having a new population made up entirely of new members, it preserves the top performing chromosomes to propagate into the next generation. This speeds up convergence of the algorithm whilst ensuring that a false convergence on a minima is reduced by preserving previous population solutions (Purshouse and Fleming, 2002).

Of these selection methodologies, it is shown that rank based and elitism based methods perform well in various use cases (Razali et al., 2011; Purshouse and Fleming, 2002; Koljonen and Alander, 2006; Ahn and Ramakrishna, 2003). There are no texts defining a suitable level of k chromosomes to select from the population amount for each re-population, as the ideal optimal level is also dependent on the type of cost function and how quickly it can converge onto a solution. These are supported by a higher mutation rate which has been shown to improve the ability of the system to break out of local minima when convergence begins (Wilke et al., 2001).

Crossover

The crossover function combines a pair chromosomes (*parents*) and creates a new pair of chromosomes (*children* or *descendants*) which inherit traits (genes) from both parents. Multiple methods exist for crossover, with common ones being single-point, two-point or k-point, uniform, partially matched, order, precedence preserving crossover, shuffle, reduced surrogate and cycle (Razali et al., 2011).

Not all of these are suitable for every type of chromosome variant, for example Partially matched crossover (Goldberg and Lingle, 2014) is used where permutation preservation is required. In the Travelling Salesman Problem, the salesman must visit multiple sites in the least amount of time. A genetic algorithm chromosome may then be populated by the visitation as a permutation set, for instance $h_i = (1, 2, 3, 4)$ representing the order of visitation. An invalid chromosome might be $h_i = (4, 4, 1, 2)$ where the salesman visits city 4 twice and never city 3. Permutation based crossovers are built to ensure preservation of these rules.

Single-point crossover is the most straightforward to implement. Two chromosomes are selected using the selection criteria, and a random crossover point α is then selected where $1 \leqslant \alpha < N$ (N is the size of the chromosome). The two chromosomes swap their values after this point and the new chromosomes added

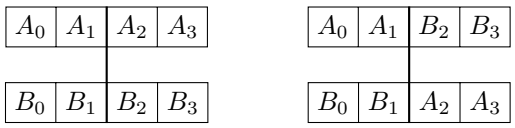


Figure 2.8: The single-point crossover method for integer chromosomes. The two parent chromosomes are split at a random point, in this example the second index. The four partials are then swapped to create two new chromosomes.

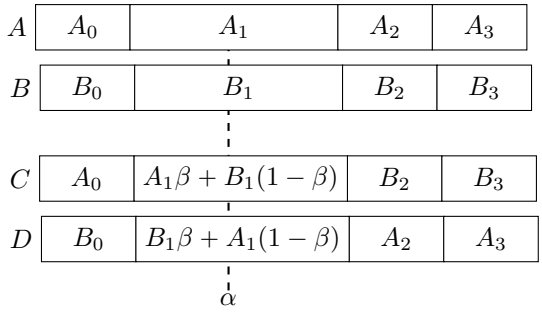


Figure 2.9: The single-point crossover method for floating point numbers. Instead of a direct split at the cross over point, the point index is selected and then the two values are merged.

to the next generation. In the case of integer chromosomes the swap point is between the two boundaries, depicted in Figure 2.8.

The continuous number algorithm allows a chromosome to be encoded as a vector of real numbers instead of a binary format (Haupt and Haupt, 2004). The process is very similar, except during the crossover phase where two parent chromosomes are combined. The crossover point splits the chromosome at the index given, but the indexed gene is blended between the two chromosomes using a β value, where β is a random number between 0 and 1. This is shown in Figure 2.9, and in equation 2.22.

$$C[j] = \begin{cases} A[j], & \text{if } \alpha < j \\ B[j], & \text{if } \alpha > j \\ A[j]\beta + B[j](1-\beta), & \text{if } \alpha = j \end{cases}$$

$$D[j] = \begin{cases} B[j], & \text{if } \alpha < j \\ A[j], & \text{if } \alpha > j \\ B[j]\beta + A[j](1-\beta), & \text{if } \alpha = j \end{cases} \tag{2.22}$$

Two-point and k-point crossovers operate in the same way, but have multiple split points, sharing the data multiple times between the two parent chromosomes. This can facilitate faster searching as the chromosomes change more on each iteration, but is only suitable for larger chromosome sizes where multiple split points can be made.

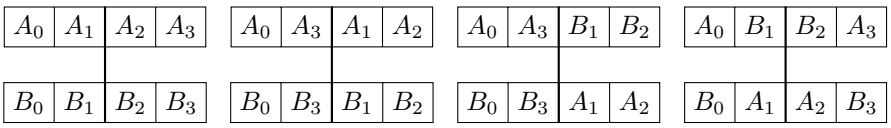


Figure 2.10: The shuffle crossover method (Caruana et al., 1989).

Uniform crossover can be used in scenarios where the chromosomes cannot be split up. It operates by taking the two chromosomes and randomly swapping individual genes between the two. The result is suitable for larger datasets and can facilitate recombination if permutations are needed to be preserved, but can result in less diverse solutions where the chromosomes are similar to each other.

A final methodology is the Shuffle method, which aims to reduce the inherent bias in other solutions (Caruana et al., 1989). This method randomises the order of the parent chromosomes before performing the crossover using the same shuffle index. The chromosomes are then combined using the single point crossover to create two shuffle children. The children are then un-shuffled using the same shuffle index as the parents to create the new chromosomes. The four stages are depicted in figure 2.10.

Both children are then added to the next population set C_{next}. This process repeats until $|C_{next}| = |C|$, the next set is fully populated.

Mutation

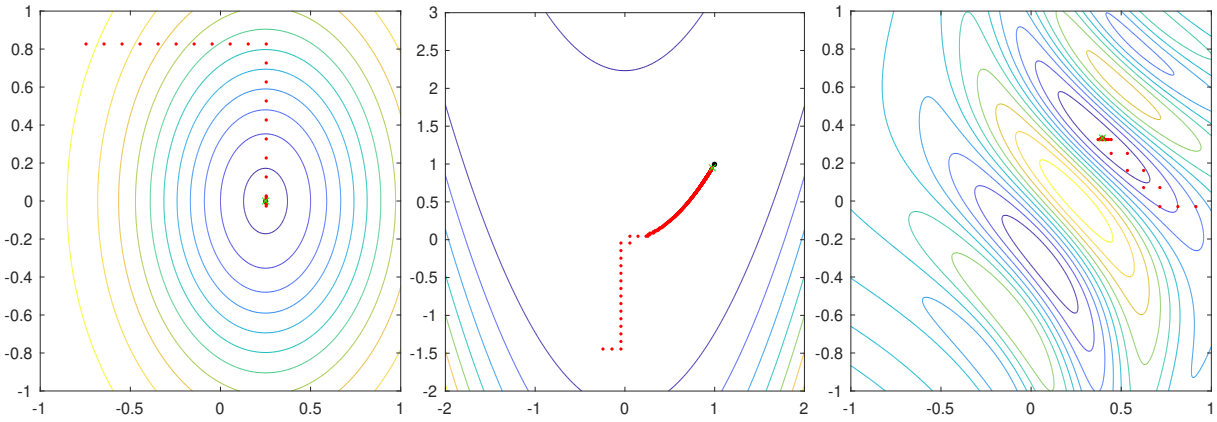
Mutation is a method for applying randomisation to the repopulation phase. Mutation is when errors are introduced in the copying and replicating of genes in biological reproduction. For the genetic algorithms, the same phenomenon can be used. It helps to maintain genetic diversity between populations and ensures that genes do not become identical or too narrow in scope (convergence).

String algorithms have displacement mutation, where a portion of the chromosome (string) is moved in place Katoch et al. (2021), amongst others. Number based chromosomes can use a randomisation effect, by selecting a gene of the chromosome to randomise (Booker et al., 1989). When a crossover occurs, there is a random probability for a mutation to occur. If a crossover does occur, then a random gene is selected to mutate.

The amount of mutation is bounded by the number range as a normal or uniform distribution. Mutations can be the entire range, or can be a small divergence from the current value. In number mutations the probability that a mutation occurs is called the mutation rate. The higher the rate the more likely a mutation will occur, with 1 meaning every child chromosome has a mutation.

2.5.4 Machine Learning Methodologies

To test the performance of the models, test functions have been made to extract their real-world performance as a search algorithm. These cost functions are purely mathematical exercises to observe how many iterations and operations and search function takes to find the known global minima. The most straight-forward type of cost function is a convex cost function with a well-defined, single global minima. This implies a function which, as the Euclidean distance increases from the minima it continues to rise. An example of this is given in Equation 2.23.



(a) Convex Eq 2.23 (b) Rosenbrock Eq 2.24 (c) Multi-Minima function Eq 2.25

Figure 2.11: Example of gradient descent in action. The red points mark the sampled regions, showing the point where it changes direction to find the global minima for three cost functions, with the minima marked with green crosses and global minima a black dot.

More complex problems may be encountered in a cost function such as the Rosenbrock function in equation 2.24. It does this by proposing a test cost function with a trench like solution (Rosenbrock, 1960). Gradient descent and other cost minimisation functions will quickly converge on the trench, but then get stuck in large iteration counts as it slowly traverses the trench towards the slightly smaller minima. The final test function is a multiple-convex function, derived from the simple-convex, where there a multiple local-minima which could obscure the global minima. Equation 2.25 gives the equation used throughout this section.

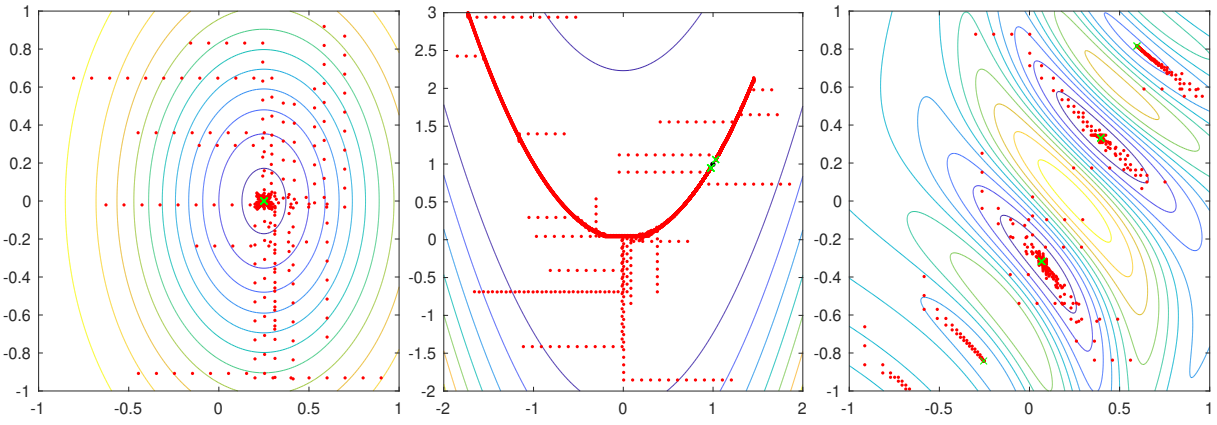
$$f(x_1, x_2) = \exp^{x_1 - 2x_1^2 - x_2^2} \tag{2.23}$$

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \tag{2.24}$$

$$f(x_1, x_2) = \exp^{x_1 - 2x_1^2 - x_2^2} \sin(6(x_1 + x_2 + (x_1 x_2)^2)) \tag{2.25}$$

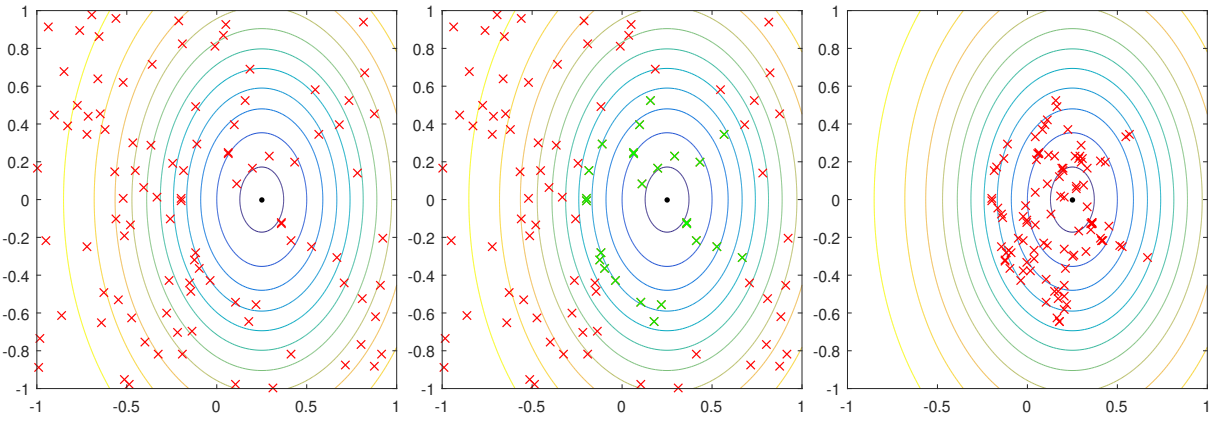
Figure 2.11 shows the gradient descent algorithm in action. As it iterates, it takes uniform samples, until it reaches the point where it would start to rise again. It then changes direction to follow the new minimisation curve. In this case, the descent was able to find the minima in 296 samples, whilst the brute-force method to draw the contours took 10,000 samples. It is clear the gradient descent in this simplified example is very efficient.

In more realistic cost functions, the gradient descent tests highlights two important issues. When using the Rosenbrock function (centre plot) to simulate cost curves with large, flat minima, the gradient descent starts to skate across the space causing it to over-sample the search space. In this case, it took the algorithm 2,783 sample points to converge near the global minima (black point of the figure). This also shows the problem



(a) Convex Eq 2.23 (b) Rosenbrock Eq 2.24 (c) Multi-Minima function Eq 2.25

Figure 2.12: The three test cost functions running for twenty iterations, with the minima marked with green crosses and global minima a black dot



(a) (b) (c)

Figure 2.13: One generation of genetic algorithm on the test single minima cost function.

when multi-minima spaces are considered, whereby a search space could have local-minima which provide solutions but are not the global solution. In this case the starting point for the gradient descent starts where it cannot get to the global minima because it would require it to ascend. As such it can get trapped into the local minima.

The solution to both of these problems is the same: to run multiple tests and take the best solution. The number of times to run is not easy to establish, but by running the same above tests with 20 repetitions the scores do improve over brute-force. Figure 2.12 shows the results of running each 20 times. This gives a solution for all three after 1,845 steps for the simple cost function, 37,056 for the Rosenbrock and 1,705 for the multi-minima. The Rosenbrock failure is well documented as a failure case for the gradient descent as it's minima is less defined. But for the others this proposes a significant improvement over the 10,000 step brute force which can still miss the global minima.

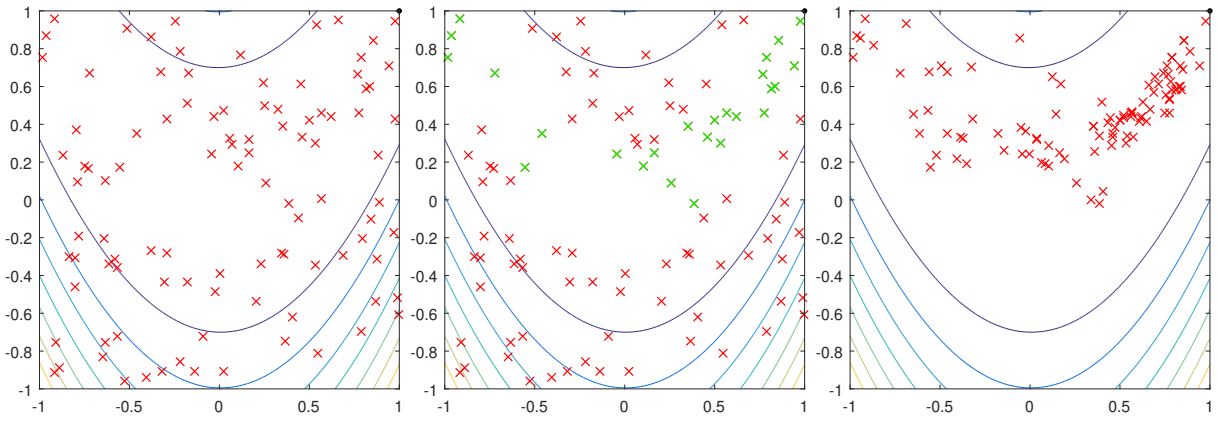


Figure 2.14: One generation of genetic algorithm on the test Rosenbrock cost function.

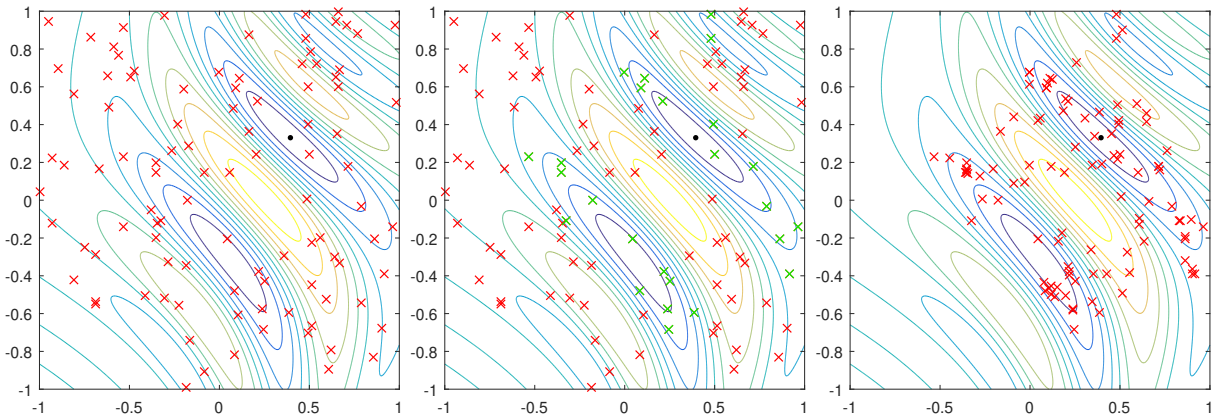


Figure 2.15: A single run of the complex cost function using the genetic algorithm really shows the ability for the genetic algorithm to reject spaces, concentrating straight onto the two main minima points.

Performance of Evolutionary Computing

For the three test fitness functions, it is clear to see how the evolutionary process differs from the gradient descent, although the end goal is similar. Figure 2.13 through 2.15 show how the process operates for one iteration. The first plots show the original randomised population, then after costs and sorting the best quarter of chromosomes are selected, highlighted in green in the second plot. These are then used to create a new generation shown in the third plot.

For the cost functions with a single minima (shown in Figure 2.13), the Genetic Algorithm is able to quickly optimise the function. For the Rosenbrock (Figure 2.14), the second generation is entirely contained in the large flat minima having already rejected the external spaces. And lastly, for cost functions with multiple complex minima (shown in Figure 2.15), the Genetic Algorithm quickly rejects the external spaces focusing on the two prominent minima points. This last plot shows the generational makeup very well as there are points contained between the two main groups, caused by the blending of the two numbers to create a new chromosome. In these examples the population size was 100, with a parent forwarding ratio of 25% (25 chromosomes carried forward as parents) and no mutation rate. The simple minima converged after 11 generations (850 sample points), Rosenbrock after 10 generations (775 sample point) and the complex multiple-minim after 10 generations (775 sample points). This shows a massive improvement over brute-force calculations (each taking 10,000 sample points as a grid) and the traditional gradient descent methodology.

A neural network requires a defined set of input-output vectors to optimise an algorithm for a given task, whilst the mixing process would require to operate on any given number of individual audio stimuli and provide and suitable output point per stream. Secondly, a neural network requires examples of solutions to known problems in order to learn how to achieve a similar solution for a future unseen problem. In audio mixing, this would require definitions of 'a good mix' which is in itself subjective. Since the solution is to find a good potential mix state from a given N-dimensional mix-space, the neural network methodology seems ill-suited. Therefore, the mixing problem is more suited to a search-style algorithm, where a space is traversed until a suitable solution is discovered.

2.5.5 Interface Design

Intelligent Music Production techniques have also been used to augment digital interfaces. Digital audio workstations in general have mimicked their analog predecessors. This provides familiarity to engineers making the transition initially, but the design practice has remained the same ever since. This is not necessarily the most effective method for producing a music production interface. Cartwright et al. (2014) provided a new 2-dimensional (2-d) space instead for engineers to use. The space maps from 2-d co-ordinates onto gain and equalisation parameters for all the incoming signals. The interface allows the user to quickly navigate a broad range of mixes, since moving one parameter on a 2-d space with a human interface device is very easy. But adjusting several effects and gains takes more time.

The results from the study, shown in Figure 2.16, indicate that engineers agree that the system enables them to explore the mix space further than if they were using the traditional DAW interface. When using the abstracted

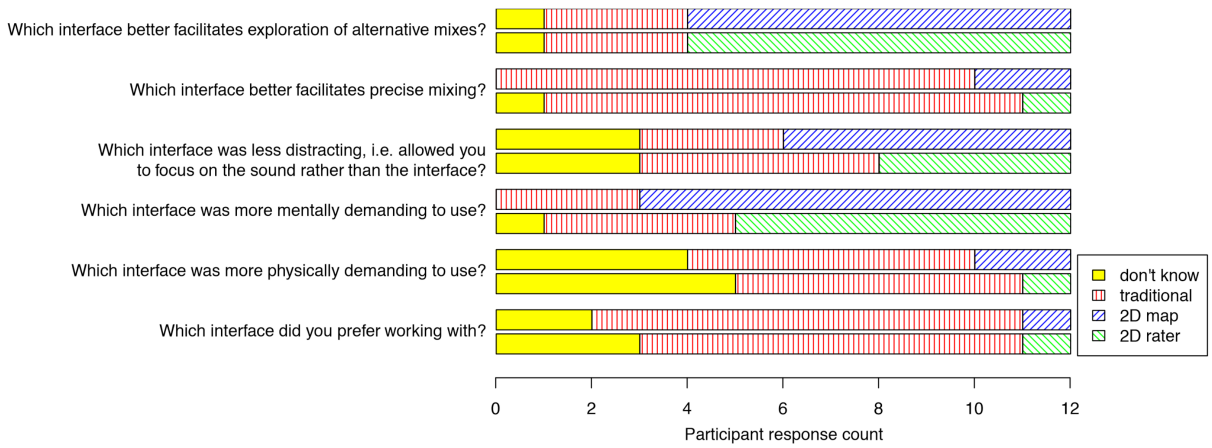


Figure 2.16: Results of the survey of the MIXPLORATION study by Cartwright et al. (2014)

system, engineers reported they did not have as much control over the mix, confirmed by the low scores for 'precise mixing' and high 'mentally demanding' score. The participating engineers concluded they preferred the traditional interface. This shows that new interfaces need to be complementary to the mixing process rather than replacing. And a system that has this kind of mapping functionality would be a powerful addition to a traditional DAW.

Line 6 made the commercial M20D which attempted to build a digital mixer for bands to use without the need for professional front of house live engineers White (2012). This product aimed to abstract the traditional interface in favour of semantic control. One example is a screen which mapped four semantic descriptors of what the sound should be onto a 2-d touch screen, allowing the performer to quickly navigate to the control space which best matches their desired artistic decision.

Other control surfaces have also been investigated, such as the Stage Metaphor (Gelineck et al., 2013). This interface works by mapping the volume along the y-axis and pan on the x-axis of a 2-d representation. Each audio source is added to the stage view, allowing for easy representation of the audio system and efficient control. This view has a compromise on information delivered to the user, but again enables the user to be more efficient. De Man et al. (2018) took this further, and showed that a polar representation of the stage is better for the end user to understand. This could be due to the polar plot being more directly related to the pan positions of the tracks, whilst the flat 2-d map can make control counter-intuitive.

2.6 Data collection methodologies

Data collection is the bedrock for performing accurate qualitative research (Gill et al., 2008). As opposed to quantitative data, where the experiment can be measured using statistical measures, certain actions are inherently subjective and require a human to provide a response to the problem. Throughout audio specifically, qualitative research has been used extensively for audio codec design (Eberlein et al., 1993) or model performance evaluation (Moffat and Reiss, 2018) to name just two such uses. This section gives a brief overview of several different methods for data collection that could have been employed in this study.

2.6.1 Quantifying engineer actions

To be able to quantify what an engineer does to a mix, the data showing what they do needs to be collected. Important factors on the design of the data collection need to be taken into account to ensure that any bias is minimised as much as possible, to ensure the outcome is as reliable as possible. This sub-section will go over a few methods of data collection that could be used in such a study.

Surveys and Interviews

Data collection methodologies range in their scope depending on the type of data that needs to be collected. Traditional data collection used interviews and surveys (Gill et al., 2008). These are done by sending a form of questions to the target subjects which they can respond to the researcher with. Traditionally, these might have been sent via post depending on the target audiences needed to respond. For example, surveys are often used in social sciences or market testing environments by sending the survey to a subset of potential users to then extrapolate by the known number of users in a population (Worcester, 1996). More recently these surveys are done online, using email, targeted advertising or other methods to question a large population of users (Granello and Wheaton, 2004). Surveys in audio data collection is not often used except informally as interviews (Pestana et al., 2014). One such example was investigating the order of audio curriculum in colleges and universities (Leonard, 2020).

Surveys can also take the form of user experience or user effort measures. NASA developed a form of survey measurement called the Task Load Index (Hart and Staveland, 1988). The subject would perform a task using the system being evaluated and then be asked a series of questions. The responses to these questions, which are defined by the Task Load Index, would then be converted into several metrics.

- Task difficulty
- Time pressure
- Performance
- Mental & sensory effort
- Physical effort
- Frustration level
- Fatigue
- Activity Type
- Overall Workload

Each of these metrics would allow the developer of a system to compare if the new methodology was improving the workload of a user, but able to quantify the areas where the performance was coming from.

The main advantage of surveys over other methods of data collection is the low barrier to entry to obtain the data and the ability to make it as accessible as possible. A survey performed in 2009 on video game players gathered feedback from 10,000 subjects across a wide range of regions and devices (Goodwin, 2009). The

breadth of this survey enabled the researchers to show what was most important in audio for video game players using real-world data, whilst the distributed nature means the researcher has less control. To obtain enough people, non completions need to be expected and understood (Jillings et al., 2018). Likewise the lack of control means individuals may not be able to ask questions if they do not understand the question, adding noise to the feedback. This can be solved through the use of focus groups and direct interviews. Whilst these raise the quality per subject they also require more time from the researcher to collect the data (Gill et al., 2008). While with modern surveys, the data collection and processing is almost automatic (Granello and Wheaton, 2004). Further advances in technology also makes interviewing less arduous as before, with mobile devices capable of video-conferencing (Leemann et al., 2020).

To gather the kind of detailed data needed from audio engineers to understand how the mixing process evolves over time, surveys would not provide a deep enough understanding and therefore not useful for this form of study. Surveys can be used to help qualify the actions being performed, although this would break the double-blind condition as the system would need a way to know which users did what to be able to contact them in the future.

Recording of Subjects

Similar to interviewing, recording of subjects allows the researcher to keep an auditory or visual record of what the subject did given a certain task. Recording the audio of a subject is similar to an interview except that the researcher or operator of the study should not be involved in the study itself. This means the subject might be recorded performing the task or process (VanDam et al., 2016). Once recording has taken place the audio and video needs to be transcribed into useful data for further research, a process which is very laborious. A 45 minute recording may take an experienced transcriber 8 hours to convert into a useful format (Sutton and Austin, 2015).

Video recording is also a very powerful form of data collection. In fields such as smart cities, tracking vehicles using deployed cameras can give high quality information on the traffic flows through a road network (Bas et al., 2007). Recording of subjects playing video games is also used extensively in video game development to understand how players react to changes in game play over time (Moll et al., 2020). Video recording is not extensively used in audio research as a tool to measure how the engineers approach a mix. For this purpose, of understanding how an engineer approaches and completes the mixing problem, this could be used as a potential method of high-quality data gathering. The subject could be placed in a controlled environment, such as a recording studio, and given several mixing tasks with audio-visual recordings of the event taking place. The camera system could pick up cues from the subject that would otherwise be difficult to obtain, such as eye tracking (Holmqvist et al., 2011) or hand tracking across a mixing console (Wang and Popović, 2009). This would generate a large amount of data to be annotated and processed from raw video footage into event tables suitable for processing.

Session File Analysis

Engineers often mix 'in the box', that means inside a software based audio workstation or DAW (Eargle, 2002, 201). The mix parameters are all stored inside a file specific to that DAW which holds the entire session state, such as routing, plugins used, asset positions and sometimes a limited undo history. This provides a very rich set of data from which engineer actions could be extracted from. Studies have previously used the session files alone to understand certain aspects of the mixing processes, such as Ronan et al. (2015b) and De Man and Reiss (2013a). In these studies the task was to understand what the final mixing environment looked like, how did engineers set up their routing structures and what processing was performed. Ronan et al. (2015b) focused this as the starting point for understanding how tracks were grouped together for example. This process is useful but it only provides the end of the task, not how the engineer arrived at that final mixing stage. Therefore this form of analysis is limited in itself as well and not suitable as a data collection strategy.

2.6.2 Perceptual Listening Tests

A commonly used method for evaluating the outcomes of music production and audio tools is to use a perceptual listening test (Bech and Zacharov, 2006, pp. 4-6). Perceptual evaluation of audio using listening tests, is a powerful way to assess anything from audio codec quality (Hines et al., 2014), or realism of sound synthesis (Moffat et al., 2019) to the performance of automated music production (De Man and Reiss, 2013a). A listening test can be used to gather research data about an audio signal, but is only appropriate if the answer to the following questions are all no (Bech and Zacharov, 2006).

- Can a measurement of the physical signal provide sufficient information?
- Does a direct measurement of the perceived audio feature exist?
- Can a suitable predictive model of the perceived audio feature be identified?

The common factor for all of the studies using listening tests is they are trying to evaluate the perception of the end user. For example an audio codec can quantify how efficient it is at compressing the data compared to raw Pulse-Code Modulated audio, but the physical measurement cannot state whether the removal of information is perceived or detrimental to the system (International Telecommunication Union, 1996). Likewise, there is no measurement for the audio codec artefact that is introduced to the signal flow. Acoustic models do exist to simulate the process of the human auditory system to convert the audio stream into high-level features for analysis (International Telecommunication Union, 1997; Moore et al., 1997). These models aim to extract loudness or masking measurements from the audio signal (Bech and Zacharov, 2006, pp. 276-279) They cannot tell you whether sound A is perceptually similar to sound B. Therefore a listening test would be suitable for this form of study.

Listening tests are powerful in their ability to study a subject and report a large amount of data per person. However there is a high degree of variability in the responses that can be gathered. There are four variability components which can impact the measurement reported by each user (Bech and Zacharov, 2006, p. 143).

- **Permanent**. The physical attributes of the test, such as any hearing impairments of the subject, the quality of the loudspeaker used or other constant impacts.
- **Treatment effect**. The influence of any variables under test.
- **Internal variability**. Any variation if the subjects physiology and psychology during the study.
- **External variability**. Any changes in the external environment during the study.

All of these variations have an impact on the final result and the ability for a test to manage these will change the quality of the results. However it does show that listening tests are, inherently, variable and therefore requires many participants, along with a strong statistical analysis and subject filtering, to create a reliable conclusion.

Testing Standards

Early listening tests in music production were performed to primarily gather information on the performance of a new audio processor, such as an amplifier or audio effects unit.To test the performance of the hardware devices, black box units were developed to allow the listener to switch between two sources: one from the original or baseline unit and the other from the new processor (Grey, 1977; Lipshitz and Vanderkooy, 1981; Clark, 1982). The test was named 'AB', since it compared product 'A' with product 'B', whereby a subject is presented the original and altered audio signal and asked to select which is better. The subject (listener) does not know whether they are listening to the new or original processor, but the test conductor does. This form of test is called a blind test.

Since these listening tests were conducted in laboratory conditions, the subject and conductor were in the same room, therefore the subject could talk to the conductor could give feedback. This could introduce bias into the test results as the conductor knows the state of the system, and unintentionally bias their feedback towards a particular state, thereby contaminating the results. To reduce the risk of biasing the results the test should be conducted double-blind, where neither the subject nor the conductor know the currently playing system. This required complex circuitry to achieve with relays and timers to reduce all number of potential cues which may give hints to the subject (Clark, 1982). To remove all possible bias, the conductor should not be in the same room as the subject.

Due to the inherent effects of the acoustic space in which the test is conducted, a number of systems pay close attention to the environment under test. To reduce the potential bias, the acoustic impact of a room should be mitigated as much as possible. This can be achieved by outlining an 'ideal' room response, which should match the environment in which the content would be naturally consumed. Toole (1982) states since the majority of tests are for consumer products, then "most listening tests should be done in rooms whose essential acoustical parameters are similar to those of a typical domestic room". Due to the variety of listening environments that could be considered, several standards were developed which aim to define the ideal listening room for subjective listening tests (ITU-R, 1990, 2015).

With an increase in quality and performance of computer playback systems, more advanced interfaces were introduced. Early systems took the AB test and digitised the interface. This made it simple to have a truly

double blind system as computers could pick a suitably random number to determine which of the two sources will be presented as A or B. New interfaces were then introduced to obtain more information from subjects. The AB test simply returns which one the user selected, but does not convey any amount of preference, nor whether they were forced to make a selection. A simple progression on this model was to take the AB method and allow the users to provide a rating or score. Parizet and Nosulenko (1999) achieved this by showing the user five possible options they could select from: "A++", "A+", "A=B", "B+" and "B++". The task was to identify the loudest source, the crucial option being able to say two things are equally loud.

Standardised parametric listening tests were then developed to improve the reliability of the tests. ITU-R BS.1534 (ITU-R, 2015) introduced the Multi-Stimulus test with Hidden Reference and Anchor (MUSHRA). It was developed specifically for evaluating small differences in audio codec performances as an alternative to ITU-R BS.1116 (ITU-R, 2015) which was unsuitable for discriminating between small differences (Soulodre and Lavoie, 1999). Rather than selecting which source was desirable, the subject would give each audio sample a rating from 0 to 100. One important aspect of this interface was the requirement to add an anchor and a reference to the pool of evaluated content. The reference is the unprocessed, original content and should score 100, or be very highly ranked. No evaluated content should be higher than the reference since they should be processed accordingly. The anchor is the reference signal with a low-pass filter at 3.5 kHz applied to purposefully degrade the content. The anchor ensures the subject uses the full scale, and provides a frame of reference for the lower bound. Other types of anchors are specified to customise the test to the application under examination.

Subjective Listening Test Design

One important aspect of user listening tests is to remove bias from the testing procedures. This is achieved by randomising the page order of the tests to ensure that there is no training bias. As the subject works through the listening test, they become more trained to the task which means that each page is not being evaluated as clinically as the others. To reduce this bias, the pages are randomised so that each page is in a different position or combination for each listener. To aid the training phase, and ensure the subject is able to understand the question being asked, a small training task is placed at the start of the test. This training task is designed to mimic the test in the question, but not be overly arduous as to take a long time or effort.

The type of interface to use is itself an important question. Multiple test interfaces exist to gather multiple forms of data on the subject, and each has their strengths and weaknesses. Little has been done to evaluate the direct comparison between test interface choice and its impact on the results of the test. Most interfaces fall into one of two categories: continuous and discrete. A continuous test interface is one which has a scale the subject can use freely, such a slider. A discrete test interface is one where the subject has to make a choice, such as an A/B or Ordinal scale.

To evaluate the choice of interface types to use, a study was conducted to find the effects of subjective test interface choice on the results (Jillings et al., 2018). The study looked to evaluate if the interface design choice itself would have an impact on the performance of the subject in the listening test. Two different tests were

proposed, each done using the MUSHRA interface standard and the A/B pairwise standard. The detailed analysis and conclusions are presented in the Appendix.

In MUSHRA, the task is try and notice differences in the audio quality and if there are any artefacts which are perceptible to the subject (Reiss, 2016). The MUSHRA standard defines the listening test to be a set of sliders to be presented to the user for each trial of the listening test (International Telecommunication Union, 2011). The MUSHRA standard was initially created to look for the differences when coding standards, such as MPEG Layer-III was being developed, started to be used more commonly. Each trial contains all the stimuli under test as a set of vertical sliders, with a scale range of 0-100. The trial is augmented with a known reference, which is the gold truth of the test, a hidden reference which is the same file as the known reference and at least one anchor. The anchors must be deliberately degraded to show a sub-optimal In the coding studies, the reference would have been the PCM Lossless encoded source file and the trial has multiple forms of coding techniques to compare against. If the listener could not tell the two apart, they would give it a perfect score and therefore that encoding method would be of a very high quality. The anchors would be low-passed at 3.5kHz to provide a deliberately degraded source which should perform worse than the stimuli under trial. To remove further bias in the trial, the order of the stimuli are randomised and their starting values are randomised. The output of the trial gives each stimuli a score, which when combined with multiple participants can be easily sorted into a rank.

The AB or pairwise trial takes a set of stimuli and compares them to each other. The stimuli all need to be compared to each other, meaning if there are 5 test stimuli to process, a total of 20 pairwise tests will be needed to complete all comparisons. Each pairing could be reviewed multiple times. The output of a pairwise is a little more complicated to process, where the results of each pair just indicates which one was preferred over the other pair combination. This can be converted into a score by distance calculations between the relationships of each instance, however this can be non-trivial if the results become circular.

2.6.3 Lab vs Web

Early test studies used physical hardware, custom built platforms were then used for digital listening test procedures (Grey, 1977; Lipshitz and Vanderkooy, 1981). Web technologies now provide a more open and accessible environment for deploying large scale studies, through traditional surveys to more specialised environments. Deploying perceptual studies on the web enables researchers to access a larger pool of participants for an insignificant increase in resources. In lab based studies, physical spaces would need to be prepared and research time spent monitoring test participants, collecting data and so on. This can place a significant time-load on researchers performing these tests to ensure each participant provides accurate data in a controlled environment. This leads to smaller sample sizes, with high quality results.

By deploying these tests on a distributed network, such as the World Wide Web, a large number of participants can perform the test simultaneously. This also allows multiple geographical locations to be sampled as well, which would have previously bee inaccessible. This creates efficiency, and increases the size of the available sample population. The trade-off in this scenario is the removal of ‘laboratory conditions’ where the researcher can control the study environment.

In some cases the ecological validity of the familiar listening test environment and the high degree of volunteering may be an advantage (Reips, 2002) The researcher loses the ability to control the acoustics of the environment along with the hardware used and potentially the participants attention span. The subject could also misinterpret instructions they have read or interact with the material during the test (Reips, 2002). Therefore it is of vital importance that the study is suited to tests which are resilient to these factors, or to provide sufficient controls to filter subjects out afterwards.

Data collection is not just through subjective listening studies, but can also be used for data annotation (Cartwright et al., 2017). In other fields, using the public through 'crowd-sourcing' has been used to great effect, by distributing large complex tasks across a group of subjects instead of a handful of researchers. This can be done for preparation or conversion of data from one format to another, a process which could be very laborious (Barbier et al., 2012). The study on music annotations showed that for crowd-sourced, whilst there is a higher variance in the data, the biggest factor was the presentation of the work. Comparing annotation accuracy between the visualisation presented showed that, even in un-controlled environments, with the right task the process can be as accurate as laboratory conditions.

These can be mitigated through suitable interface designs, training phases and screening of subjects. Studies have shown there is no demonstrated difference in reliability between laboratory conditions and distributed tests (Schoeffler et al., 2013; Cartwright et al., 2016). With proper laboratory controls applied to their listening studies, both proved that results could be obtained which were of the same quality as each other. The noise of the uncontrolled data collection needs to be considered and mitigated. In one experiment, the reaction time between audio stimuli and the human response was compared between laboratory conditions and through a mobile phone app. The phone app showed far greater variation in the response times given, not least because of the variation in device accuracy, but also playback latency reporting (Nagy et al., 2016).

2.7 Enabling Technologies

From the previous section, it is clear that a traditional data collection method is not suitable to understand how the engineer completes a mixing problem. A system needs to be developed which can gather the data in a machine-ready format for further processing of what the engineer was doing at each stage. Surveys, interviews and session analysis are not enough to obtain the fine grained data required to explore this problem. Likewise video and audio recording of what engineers are doing would require a large amount of time commitment from the researchers to conduct the controlled study and then transcribe the data into a format ready for analysis and processing.

The system should be distributed to as many participants as possible. Whilst this will mean more noise in the data being collected, it should be possible to still obtain the right information by having a greater number of participants, as discussed above. This section goes over the enabling technologies to allow for such a form of data collection to be developed.

2.7.1 Music and the Internet

Even before the launch of the world-wide-web in 1995, audio over networks was already a burgeoning industry. With the launch of Audio CD's in 1982, commercial ownership of digital audio was fast replacing traditional Vinyl records. Figure 2.17 shows that in 1984, CD only accounted for 0.5% of the total revenue ($44.7M). By 1990, it accounted for 45.8% of the revenue with cassette being only slightly ahead at 46.0%.

Early internet adoption was slow, utilising narrow-band technology such as Dial-up and ISDN. This meant most personal connections were 56kbps to 256kbps. Given the Red-Book Audio CD standard (16-bits per channel, 44.1kHz sample rate) meant real-time audio needed 1.411Mbps, real-time internet streaming was not suitable to uncompressed PCM. Advances in audio compression technology and internet line speeds allowed producer Phil Ramone to record Duets and Duets 2 with Frank Sinatra, where two session vocalists transmitted their performances, live to the studio, over ISDN (Burgess, 2014, p. 122). Despite being transmitted using a lossy format, Ramone still commented the transmission was "excellent quality" (Burgess, 2014, p. 42-49). And in 1995, the first live broadcast of radio over the internet was used, using the RealNetworks prototype to the RealPlayer.

Despite the internet launching in 1995 to the masses, digital revenue did not play a role in the industry until 2004, where music downloads accounted for 1.5% of global revenue, compared to 92.7% for Audio CDs. This is partly due to the nature of the internet that time, with media streaming being very time consuming over 56k dial-up and ISDN links. This made private residential access limited until the early 2000's where 31% of developed world had access, rising to 62% by 2007, making it more commercially viable. The development of the MPEG Layer-III standard also provided a flexible coding standard for efficient data storage and playback enabling consumer grade digital playback (Fraunhofer IIS, 2021). Asynchronous Digital Subscriber Lines (ADSL) introduced in 1998 gave broadband connections to premises for the first time, sparking an explosion in download speeds from 8Mbps to Gigabit connections today. Now, digital media accounts for over 90% of the U.S. audio market (RIAA, 2020).

Over time the internet has also become more available to a greater reach of the population. In 2015, 11% of the population of the United Kingdom had been estimated to have never used the internet Office for National Statistics (2015). By 2020, this figure had dropped to 6.3% (Office for National Statistics, 2020). Likewise, users of over 75 years old in the United Kingdom increased over the 5 year period, rising from only 33% of persons aged 75 and over having used the internet (Office for National Statistics, 2015). After 5 years this number had increased significantly to 54% of persons aged 75 and over having used the internet (Office for National Statistics, 2020). The demographics also show marked improvements, in 2001 over 143 million Americans had used the internet, 54% of the population at the time (Granello and Wheaton, 2004). Most of these users were White (87.2%), Male (66.4%) and married (47.6%) (Granello and Wheaton, 2004). The 2020 report by the ONS showed that, whilst in the United Kingdom 88% of users where from a White ethnic background, the level of proportion of users from other ethnic groups accessing the internet rose from 86.6% participation in 2013 to 95.7% (Office for National Statistics, 2020).

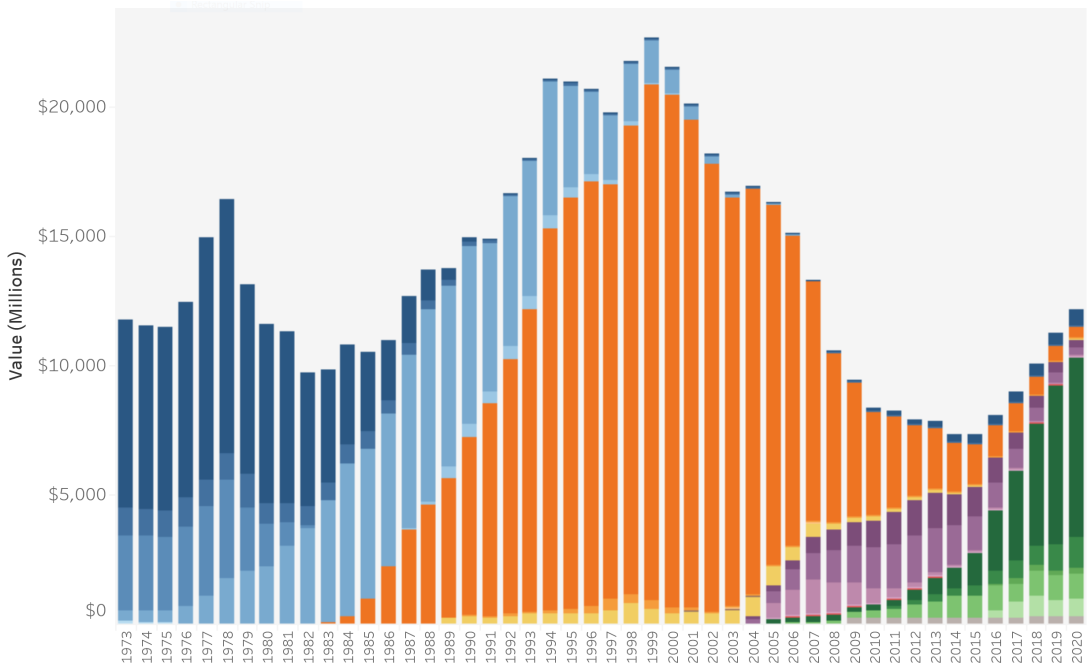


Figure 2.17: Revenue generation in millions USD of various formats in the U.S. market (RIAA, 2020)

Digital audio on the web not only plays an important role for consumers of music, but also content delivery. With radio live streaming over the web (Berry, 2020), to music on demand streaming services (Kreitz and Niemela, 2010) and now to user interactive content (Carson, 2021). Web technologies have also evolved to make audio content delivery more seamless (Yan et al., 2012). With the introduction of HTML5 in 2008, the first native audio tag was standardised, allowing website designers to embed native audio playback in the browser without the need for third party software such as Flash or RealPlayer (Yan et al., 2012). Then the Web Audio API enabled native real-time manipulation of audio streams (World Wide Web Consortium, 2018). This kick-started several new-age projects which would otherwise have been complex and cumbersome to deploy to the web. This includes real-time transmission and synchronisation of audio between performers (Gover et al., 2021), a system which would have required bespoke software and transmission protocols prior to the collection of new web stands.

2.7.2 Web Audio API

The World Wide Web Consortium introduced the Web Audio API to enable browsers to have a common standard for audio manipulation in the browser (World Wide Web Consortium, 2018). The API exposes into the JavaScript language in Browsers a fully fledged audio routing, processing and scheduling agent. Instead of writing audio Digital Signal Processing code in JavaScript, the browser sends calls to the API to create a chain of processing which can then execute using high speed, compiled code. The standard supports real-time manipulation of low-level audio samples using defined processing nodes for performing common tasks, such as gain, filtering and compression. Custom processing can be implemented through audio worklets, which use parallel threaded web-workers to execute the code (Choi, 2018).

The API is exposed to JavaScript as set of functions to control the creation and management of these processing blocks, called **Nodes** (World Wide Web Consortium, 2018). Each node performs a specific Digital Signal Processing function, exposing controls through parameters which can be manipulated. The nodes can accept an audio stream as an input or generate an audio stream if it is a source node. Likewise, the nodes can forward the processed stream onto other nodes. Several DSP blocks are specified in the standard which all major browsers implement and their behaviours are defined in the standard to ensure the same processing occurs on all browser variants (World Wide Web Consortium, 2018). In 2022, all major browser vendors support the Web Audio API across all their major browser brands, with over 95.67% of global internet users connecting with a supporting client (Can I Use, 2022).

2.7.3 Web Audio Evaluation Toolbox

Tools have been developed to aid researchers in building and use distributed listening tests (Kraft and Zölzer, 2014; Schoeffler et al., 2015). These focused on building MUSHRA tests for near-laboratory conditions to end users and were often strictly built for these standards. The Web Audio Evaluation Tool (WAET) was built as an open-source tool to help researchers quickly build subjective listening tests which uses the browser as a method to present the testing environment to the user. The tool was specifically designed to remove the need to rebuilding the listening test environment every time, by allowing simple changes using configuration files. WAET operates entirely on the client side, utilising the HTML5 Web Audio API (World Wide Web Consortium, 2018), supported by most major web browsers (Can I Use, 2022).

The WAET system is made up of three distinct parts: test creator, test running and analysis. The test creator is a stand-alone HTML page, meaning it can be run without a running web server. This page can be used to build a new test from scratch, or modify an existing test. The creator exposes all of the options to the user to create a powerful testing platform, whilst also providing pre-defined templates to confirm to various testing standards, as shown in Table 2.1. Some of the extra capabilities of the system are included below (Jillings et al., 2016c).

- **Survey options**: Before the test a survey can be shown to the subject. This includes options for text, number, radio and checkbox style data collection systems. Text and videos can also be presented to the subject with minimum duration timers to ensure the subject has read each entry. The survey is also dynamic allowing for options to be skipped if they are not relevant based on the responses from previous surveys.
- **Looped playback**: Repeat current stimuli from the start when end is reached, until the stop playback or the page is submitted.
- **Comments**: Under the main interface, each fragment on a page can be given a text box allowing the subject to enter in any information they think is relevant to the researcher. Specific survey-like questions can also be added with radio, checkbox and text entry options.
- **Synchronised playback**: Each fragment can be played synchronously, allowing for smooth transitions between the changes using the **cross fade** option.

- **Fixed Sample rate**: The web audio API automatically resamples files if they do not match the system reported sample rate. This can stop the test if the system sample rate does not match the desired sample rate.
- **Randomise page order**: Unintentional bias can be introduced into the order of the pages presented to the subject, therefore WAET can randomise the page order. It also supports fixed page order, such as making sure a training page is placed at the start.
- **Randomise stimuli order**: Same as randomising the page order, the fragments can also be randomised on the page as well.
- **Require playback**: Require that each fragment has been played at least once, partly or fully.
- **Require Moving**: Only for continuous scales, this requires that each stimuli is moved at least once before submitting.
- **Require comments**: Require the subject to enter text into each comment box before continuing.
- **Repeat pages**: Pages can be repeated a number of times, each repeat is stored as a separate entry.
- **Flexibile Pages**: Tests can be scrolled, whereby the subject can go back or forwards through the collection of test pages.
- **Scale usage**: The subject can be prevented from submitting their results until at least one fragment is below and above a given set of scale ranges.
- **Hidden anchors and references**: The stimuli can be labelled as an anchor or reference, and the subject can be prevented from submitting if they place a stimuli below the anchors or above the reference.

The test runner can be placed on any Apache or NGINX style web server which hosts static pages. The URL to the desired test XML file from the test creator is an XML file, which is stored alongside the rest of the test, is passed to the test runner page as a URL option. This allows one single deployment to support multiple tests. The runner loads the test definition XML which holds the information such as which interface to load, the URL of the various stimuli and any other options or constraints that need to be presented or enforced on the subject. At each stage, a results XML is generated and returned back to the server for storage. This XML file is updated at periodic moments, such as the completion of a page or survey entry, to ensure the server always has the most up to date information.

Once the test result has been saved, the data can be processed using a set of python scripts.

- **Score Parser**: Converts the XML pages of the completed results and extracts each page of information into a CSV file, containing the rating given for each subject for each fragment on each page.
- **Score Plotter**: Takes the Comma Separated Value files from the parser and generates labelled box plots of the results.
- **Survey Parser**: Extracts the survey responses from each of the page surveys and test surveys into a CSV file per survey, along with the duration of time spent on each survey option.

Toolbox	**HULTI-GEN**	**APE**	**BeaqleJS**	**MUSHRAM**	**WAET**
Language	MAX	MATLAB	JavaScript	MATLAB	JavaScript
Remote Capable	No	No	**Yes**	No	**Yes**
Pairwise/AB	**Yes**	No	No	No	**Yes**
ABX Test	**Yes**	No	No	No	**Yes**
MUSHRA	**Yes**	No	**Yes**	**Yes**	**Yes**
APE	No	**Yes**	No	No	**Yes**
Rank Scale	**Yes**	No	No	No	**Yes**
Likert Scale	**Yes**	No	No	No	**Yes**
ABC	**Yes**	No	No	No	**Yes**
Bipolar Scale	**Yes**	No	No	No	**Yes**
Absolute Category Rating	**Yes**	No	No	No	**Yes**
Degradation Category Rating	**Yes**	No	No	No	**Yes**
Comparison Category Rating	**Yes**	No	No	No	**Yes**
Continuous Impairment Scale	**Yes**	No	No	No	**Yes**

Table 2.1: A Comparison of the features of the Web Audio Evaluation toolbox against several alternatives: Hulti-Gen (Gribben and Lee, 2015), APE (De Man and Reiss, 2014), BeaqleJS (Kraft and Zölzer, 2014) and MUSHRAM (Vincent et al., 2006).

- **Comment Parser**: If the listening test has on-page comment boxes for the subject to enter in to, this parser will extract them into a CSV for further analysis.
- **Timeline View**: If the listening tracking option is turned on, then each time a fragment starts and stops being listened to is logged. These are then plotted on a continuous timeline.
- **Timeline View Movement**: If the movement tracking option is turned on, then each adjustment made by the subject on a continuous slider interface is stored along with the relative time from the start of the page. These are plotted for each subject showing how the movements occurred over time, along with when the subject was listening to each fragment.
- **Evaluation Stats**: Extracts the performance of the evaluation including how many evaluations were made, how many pages were completed, how long each page took and how long the presented order of pages took.
- **Generate Report**: Runs several of the above extractors and creates a LaTeX and PDF file in a report ready format, allowing researchers to quickly extract formatted tables and plots to include in their documents.

Unlike most previous subjective listening test frameworks, WAET was built to support several test standards, including MUSHRA, in a modular framework. Table 2.2 gives an overview of some of the more common testing standards. A continuous interface is one which provides an axis or slider allowing the user to provide a range of

Interface	Reference	Type
MUSHRA	ITU-R (2015)	Continuous
APE	De Man and Reiss (2014)	Continuous
AB	Lipshitz and Vanderkooy (1981)	Discrete
ABC/HR	ITU-R (2015)	Continuous
Bipolar	N/A	Continuous
ABX	Clark (1982)	Discrete
Pairwise	David (1963)	Discrete/Continuous

Table 2.2: The various interfaces and testing standards that the Web Audio Evaluation Toolbox can support (Jillings et al., 2015).

Figure 2.18: Four common test standards in WAET. Clockwise from top left: Traditional AB test ITU-R BS.1116 (ITU-R, 2015), MUSHRA ITU-R BS.1534 (ITU-R, 2015), APE (De Man and Reiss, 2014) and Ordinal test

Study	Reference
SampleRNN: An unconditional end-to-end neural audio generation model	Mehri et al. (2016)
Perceptual evaluation of synthesized sound effects	Moffat and Reiss (2018)
Phase vocoder done right	Pruuvsa and Holighaus (2017)
Creating real-time aeroacoustic sound effects using physically informed models	Selfridge et al. (2018)
Audio time stretching using fuzzy classification of spectral bins	Damskägg and Välimäki (2017)
Deep learning for black-box modeling of audio effects	Ramírez et al. (2020)
Modal synthesis of weapon sounds	Mengual et al. (2016)
Sound synthesis of objects swinging through air using physical models	Selfridge et al. (2017b)
Perceptual evaluation of source separation for remixing music	Wierstorf et al. (2017)
The impact of compressor ballistics on the perceived style of music	Bromham et al. (2018)
Physically derived sound synthesis model of a propeller	Selfridge et al. (2017a)
Exploring object-based content adaptation for mobile audio	Walton et al. (2018)
What the future brings: Investigating the impact of lookahead for incremental neural TTS	Stephenson et al. (2020)
Noise annoyance in urban life: the citizen as a key point of the directives	Labairu-Trenchs et al. (2018)
Machine Learning Multitrack Gain Mixing of Drums	Moffat and Sandler (2019)
An Investigation into the Relationship Between the Subjective Descriptor Aggressive and the Universal Audio 1176 FET Compressor	Moore and Wakefield (2017)
A unified neural architecture for instrumental audio tasks	Spratley et al. (2019)
Modelling musical similarity for drum patterns: A perceptual evaluation	Bruford et al. (2019)
Investigation of metrics for assessing human response to drone noise	Torija and Nicholls (2022)
Multilevel annoyance modelling of short environmental sound recordings	Orga et al. (2021)

Table 2.3: A subsection of the 126 citations for WAET when it was actively used in the study as a way to evaluate or investigate.

values, based upon the listening test being conducted and the scales being used. Figure 2.18 shows four of these listening test interfaces included in WAET running in a browser. New listening test standards can be added by building a new interface package, or including the required parameters into the included test building software.

The interface supports multiple scale standards, as shown in Table 2.1. A continuous interface allows for many comparisons between multiple different stimuli, along with reference stimuli, hidden reference and hidden anchors. These are used to help remove bias on the scales, by giving the subject stimuli which is better or worse than the stimuli under test. For example, the listening tests conducted for evaluating the performance of codecs in voice transmission have a hidden reference which is the unaltered master recording, and two anchors with deliberate filtering to make the speech unintelligible (International Telecommunication Union, 1996). This means the subjects are forced to give an honest ranking of not just how close the processed samples are to the reference, but how much better they are than the anchors.

The WAET instance is controlled through a configuration document which defines the test parameters including the interface type, audio samples to use and any survey questions. The system collects data not just on the final ranking given, but also how the subjects performed. This includes time taken to complete the page, movements done, start and stop times when auditioning stimuli and any errors or messages shown to the subject. Conditions can also be set to enforce the subject to meet certain conditions, such as using the full range of the axis given, playing every stimuli and moving every stimuli, where appropriate.

Each participant that loads the study generates a results file which contains these metrics from the subject. Both the specification and result documents are stored as XML format, which is readily parsed by most languages as well as being of a suitable structure for direct editing. The tool also includes a test creation page which provides a more user-friendly method for building the test definition files.

This tool was developed to reduce the need for repeated creation of listening tests from scratch. Since its release, it has been cited 126 times. A subset of these academic outputs where WAET was used as a primary form of data gathering or evaluation is given in Table 2.3. It also demonstrates that web based data collection can yield meaningful results with careful filtering of subjects.

2.8 Conclusion

The balance mix is a well understood phase of the mixing stages, occurring right at the start of the mixing process as a way for the engineer to organise and familiarise themselves with the recordings (Izhaki, 2012). But with conflicting arguments and differing views on what the goal should be by supporting texts. With certain individuals banning grouping structures, whilst others support the use of corrective effects at this stage.

There has been very little objective studies into how an engineer approaches, completes or performs in the studio when producing a song. Previous studies on how engineers approach the mix are either focused on general qualities of the mix (Pestana et al., 2013), or they focus on small aspects of the mix (Wakefield and Dewey, 2015; King et al., 2010) or looked into creating tools to help educate engineers (Sauer et al., 2013;

Stables et al., 2014). Therefore to understand how best to assist an engineer in the mixing phase, the first question to ask is how does an engineer complete a mix. No single resource or combination of texts gives a clear picture from which further systems should be based upon. Automatic mixing systems that currently exist are based upon these prior conflicting rules, and generally are focused on niche areas of mixing technologies, often taking away control from the engineers themselves. Therefore it is clear that a study which could capture not just the engineers actions but also the quality of those actions on the mixing process would be valuable. To support this the data collection method must be a low-impact form of collection such that it is as close as possible to a real-world scenario, so that engineers perform as they would normally.

Chapter 3

Data Collection

The process of releasing a piece of music is made up of several key steps from composing, recording, editing, production (mixing and mastering) and distribution (Owsinski, 2017). All of these steps have defined goals to complete before moving on to the next stage. During the mixing phase, an engineer will take a set of recorded tracks and convert them into a single rendered performance, based upon the instructions from the producers and artists. Editing and production are often entwined today because their roles are performed in the same domain using a Digital Audio Workstation (Putnam, 1980). There is little academic research into the performance of engineers in the mix, as it is highly subjective and contains numerous steps to complete.

From Chapter 2 it is known that one of the first tasks to complete a mixing process is to complete the balance Mix, and how this crucial step is performed at the start of the mixing process sets up the foundation for the rest of the project. Existing literature and research into how this stage of the mixing process is performed is not well understood, with mixing analysis being performed either on the final processed mix (Wilson and Fazenda, 2015a) or examining specific attributes of the mixing process (Wakefield and Dewey, 2015). This chapter introduces two novel data collection systems to obtain the required information directly from engineers.

3.1 JSAP - Audio Plugins for the Web

Previous studies have used smaller audio software as data collection tools. The SAFE project was set up to capture data from engineers using their plugins in their mixes (Stables et al., 2014). These plugins collected audio features, parameter settings and the semantic terms, which could then be used to create new interface designs (Stasis et al., 2016).

Because of their influence from hardware mixing consoles and studio practices, most DAWs are still restrictive in how they operate (Constantinou, 2019). Any third-party processing must happen inside a track's signal flow, therefore it is very difficult to build automatic systems for DAWs. Most plugin formats only accept a single track's signal flow, similarly again to the process of outboard gear, which only receives the signal as playback occurs. Therefore, to implement cross-adaptive processes (Verfaille et al., 2006), the DAW must send multiple

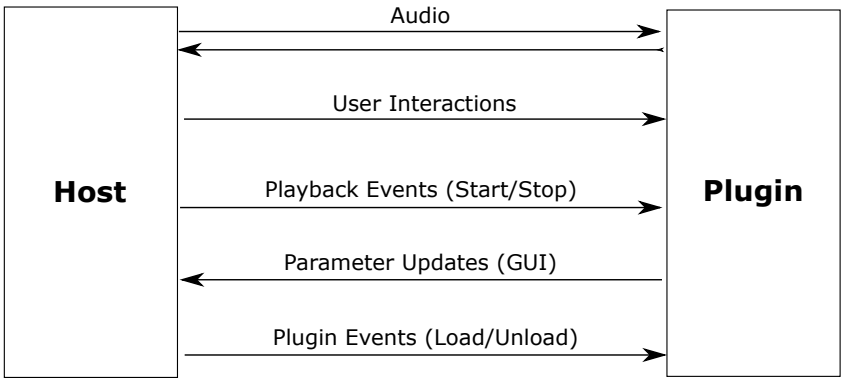


Figure 3.1: Visual layout of the communications between the plugin object and its host

streams to the same plugin instance. This requires DAWs to support either non-traditional effects routing or multi-track routing matrices.

Native audio plugins, such as RTAS or VST, cannot exist in the DAW as browsers will not execute arbitrary user binary code. Although cross-compiling through projects such as Web Assembly does allow for near-native executions (Haas et al., 2017). Several frameworks have been developed to aid audio programmers to build custom audio processing chains in the web. Tone.js (Mann, 2015) expands the number of base processing nodes by creating Web Audio-like objects. These objects are tightly bound to the existing specification, making them very easy to deploy and use. The framework does not specify any graphical user interface standard for plugins. Web Audio API Extension (WAAX) Choi and Berger (2013) and Web Audio Modules (WAM) Kleimola and Larkin (2015) both define methods for building plugins with graphical user interfaces. They draw upon the principles of traditional desktop standards to build a processor and editor environment. Web Audio API Extension (WAAX) Choi and Berger (2013) requires developers to use their own WAAX units for processing, which is not ideal for 3rd party development as it limits the available effects to those which can be built using their pre-defined nodes. The Web Audio Modules (WAM) project Kleimola and Larkin (2015) is more open, as they provide a wrapper for VST compiled code to be converted into JavaScript. Efforts to unify these disparate standards into one open framework have been attempted before (Buffa et al., 2018).

All of these proposals overlooked a significant aspect for developers of audio processing tools, plugins and their end users. None of these systems took the host requirements into account. This makes any standard difficult for end users and consumers to use, since they may not be interchangeable or may require specific communications and hooks to integrate. A traditional Audio Plugin is a self-contained environment for processing discrete audio frames with a host interface. The host serves audio frames and handles the lower-rate communications for parameter controls, playback events and user interfacing, depicted in Figure 3.1. A plugin in the browser should behave the same way and therefore the standard should be able to sanitise these communications, facilitating the development for third-parties.

Figure 3.2 shows the JSAP project scope and relationship of the classes being used. This is an empirically developed plugin framework, which we designed to specifically support intelligent audio production research on the web. Each plugin itself manages its own graphical user interface to present to the user. They also manage

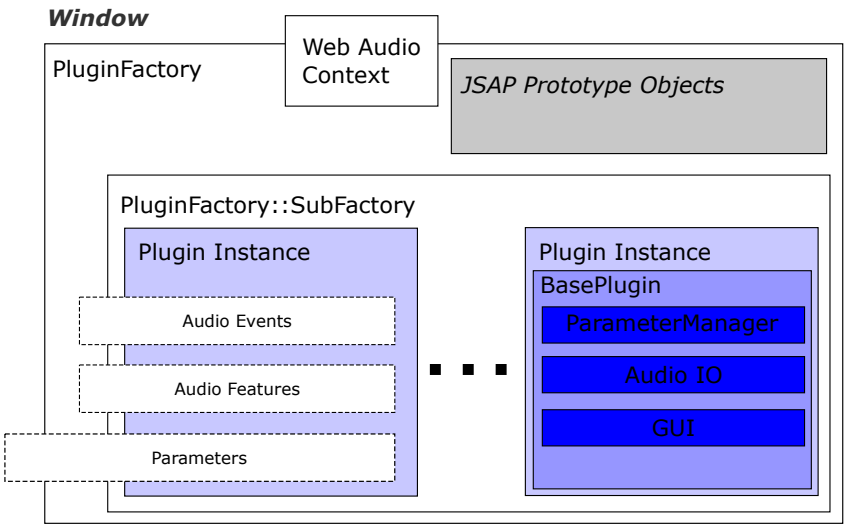


Figure 3.2: Class structure of the JSAP standard, showing how objects can be inherited or controlled.

the parameters of the audio effect, allowing a standardised method for control. Audio processing is performed through the Web Audio API nodes, allowing developers to choose to either use the web audio building blocks or custom DSP code such as through Audio Worklets (Choi, 2018).

The host is created by constructing the `PluginFactory`. This factory is given the prototype constructors for the plugins and allows it to build multiple copies of these plugins. The plugins are linked together using processing chains, two exist for MIDI and audio processing respectively. These handle the processing order between plugins, communication and common functionality such as plugin bypassing. The `PluginFactory` also manages the event flow and knowledge structure for the plugins. For example, when audio playback starts and stops, as this is considered an important event in many use cases. In the other standards, this is not directly supported, but JSAP allows the browser to call one function in `PluginFactory`, which in turn communicates this to all the plugins. By standardising the host interaction layer, program developers can abstract the plugin interfaces and have confidence that a conforming plugin will behave appropriately and correctly.

3.1.1 Audio Feature Extraction

One design choice of JSAP was to support auto and cross-adaptive audio effects (Reiss, 2011). These take features from other tracks in the mix session and extract features from these signals for processing. Audio feature extraction describes a collection of algorithms which take an audio stream and extract higher-level descriptions and statistics. These are often far smaller than the original signal and are therefore easier to process. This is achieved by collecting frames of audio from other tracks to feed into a control signal. One such example of a common cross-adaptive audio effect is a side-chained dynamic range compressor (Giannoulis et al., 2012). Instead of the gain reduction being determined by the source signal, it is keyed from another signal and therefore the amount of gain reduction on the source signal is controlled by the loudness detected on the keyed channel. A communication system to allow plugins to request these features from other plugins was developed, to help reduce the possibility that multiple feature calculations would be operated which would

Feature	Firefox	Chrome	Safari	Edge
Tonality	0.791	0.712	0.411	0.277
Spectral Standard Deviation	1.022	1.094	0.415	0.334
Standard Deviation	0.819	0.627	0.377	0.248
ASDF	2,095	4,280	5,453	28,185
AMDF	2,319	4,252	2,476	3,102
Discrete Cosine Transform	56,395	184,412	50,618	142,830

Table 3.1: Feature extraction time in *ns* on up-to-date (July-2016) desktop browsers for 1024 samples.

Feature	iPhone 5	iPad 3rd Gen.	Nexus	Linx
Tonality	0.621	1.287	1.324	1.046
Spectral Standard Deviation	1.108	1.429	1.741	1.048
Standard Deviation	0.517	1.263	1.192	0.959
ASDF	7,246	18,779	9,096	45,319
AMDF	4,491	9,971	8,824	8,305
Discrete Cosine Transform	64,297	169,967	315,077	347,625

Table 3.2: Feature extraction time in *ns* on mobile browsers for 1024 samples.

be redundant. The developed tools would all need to have a form of the audio feature extraction tool kits, commonly found in other languages.

Moffat et al. (2015) gives an overview of several developed software libraries and compares the feature coverage of each. Their work shows that no libraries are complete in their feature coverage and that there is a significant divergence between the execution times of each, which may be related to the languages used. To facilitate the developing of a new library, it was decided to take an existing, open-source library and refactor the code for JavaScript. LibXtract meets these requirements: it is published on an open repository on GitHub under a permissive licence and is written in C language, which is easily portable to JavaScript (Bullock and UCEB Conservatoire, 2007). The new feature extraction library created is 'JS-Xtract', which is a lightweight JavaScript implementation designed to run seamlessly with JSAP (Jillings et al., 2016b).

The library was written to match the functionality of the LibXtract library as closely as possible, whilst using the features of the web languages to aid operation. The library has all the features in the LibXtract library as well as chroma feature extraction (Müller and Ewert, 2011) and vector manipulation functions, such as interlacing and de-interlacing. To demonstrate the performance of the library, feature extraction was performed on a 1024 point sine wave polluted with Gaussian white noise. The test was performed by creating a web page which would load the library on the target machine, collecting the information exposed through the Navigator API (Reid, 2015). This exposes the browser vendor, browser version and host operating system. The tests were repeated 3,000 times to ensure validity of the results across 41 unique browser-computer pairs.

(a) VST

(b) JSAP

Figure 3.3: The Parametric Equaliser from the SAFE project (Stables et al., 2014), VST on the left and JSAP on the right.

Millisecond accurate timestamps are obtained through the W3C High Resolution Time Level 2 API (World Wide Web Consortium, 2019). Upon each execution run, a timestamp is taken before and after each iteration, giving an execution time in nanoseconds. The average for the machine was returned to use as the score. The fastest times were for scalar feature extractions since they have the lowest operational counts. The three highest scoring functions all use matrix mathematics, requiring memory allocation to be performed as well as higher operational counts. Firefox showed the best overall performance, although the slowest for the scalar features. Chrome and Edge both showed unstable performance for the DCT calculations whilst Firefox and Safari proved consistent results for the vector features (slowest three).

The library is available as an open source project on GitHub, as well as through the Node Package Manager for integration into web development environments.

3.1.2 JSAP in research

The developed system was used in plugin recommendation studies (Stasis et al., 2017b,a). This was done by converting the VST built SAFE plugins into a set of Web Audio API plugins (Stables et al., 2014). The SAFE-EQ is shown in Figure 3.3 alongside the JSAP version. These operated identically to each other, including data collection and recall. A web page was then developed for the system to give engineers an audio file and a challenge, such as 'Make this sound warmer'. The engineers would then add these plugins into a chain along with their processing before submitting. The collected data was then converted into a recommendation system to propose the most suitable given plugin a set of input features and a target semantic descriptor.

3.2 Web-based DAW for collecting mixing parameters

The online listening test platform is useful for evaluating the output of a researched system, such as how realistic is a new synthesis model against previous systems (Moffat et al., 2019). To collect data on how the balance mix is created a specialised piece of software to gather the unique workflow is required. Previous examples of collecting data of mixing practices either required user surveys (Pestana et al., 2014), recording participants during their sessions or using derivative experiments which targeted specific cases (Wu et al., 2019). The system will need to meet certain design requirements and function as similarly to a normal Digital Audio Workstation.

3.2.1 Requirements

The design requirements for the collection platform were relatively simple. It should be:

- familiar to traditional DAW users immediately,
- feature all reasonable items required to complete the task of making the balance mix (multiple tracks, sends, faders),
- collect every user action performed and store it in a database,

The workstation was designed to be as familiar as possible for users of established platforms to lower the training time for participants. To aid this, two processing views were created: a timeline view which shows the tracks and content in time, and a mixer view which shows a skeuomorphic audio console with vertical channels (Marrington et al., 2017) These are the two most common UI configurations for most mainstream workstations, as shown in Figure 2.2. These two views should have a layout which is immediately recognisable for audio producers, thereby ensuring the data collected is not polluted by uncertainty on how to complete an action. The DAW also needs enough functionality to operate as a stand-alone piece of software, allowing users to be able to use the software as they would normally. Therefore, at a minimum, the following interactions would have to be supported in the environment to enable engineers to complete the mixing tasks (Marrington et al., 2017).

- Transport: Play, Pause, Stop
- Tracks: Audio Tracks and Busses
- Master Bus for output
- Volume and Panning on each track/bus
- Audio Regions with basic editing (trim and movement) and waveform
- Metering during playback
- Mute & Solo tracks

These actions are fairly fundamental for a DAW to be usable, although by no means a fully-fledged piece of software. These interactions should be enough for a user to be able to produce a balance mix, which is the

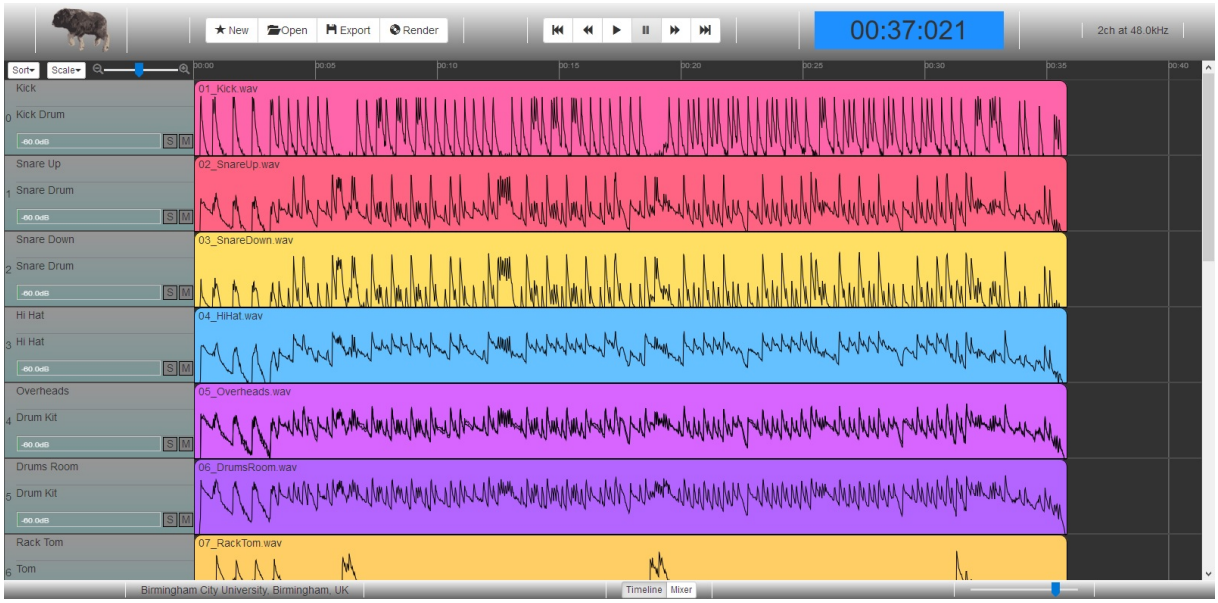


Figure 3.4: The empirically developed DAW presented in Firefox 51, showing the Timeline view

objective of this research. The DAW does not support MIDI tracks or the ability to export the final mix, which is beyond the scope of the study.

This tool needed to be functional and be readily distributed to multiple participants simultaneously to get as large a data set as possible to analyse Therefore a web-based platform would be the most straightforward to use, so participants would not need to install any specific software. The online nature would also mean all the user data could be transmitted in real-time back to a data collection server for storage. When developing the DAW, an emphasis was made on speed over audio processing quality, since it was more important to complete this phase and proceed on to the actual data collection phase. To this end, several packages were used to speed up deployment, including Bootstrap and AngularJS (Green and Seshadri, 2013). Both of these allow for easy creation of user interfaces which are standardised across browser delivery, whilst facilitating complex multi-view web pages by developing components to present to the user(Jadhav et al., 2015). Whilst in itself the development was an important part of the project, the previously discussed web technologies made it feasible to develop in a short time frame, including the Web Audio API (World Wide Web Consortium, 2018).

The developed data collection system is shown in Figure 3.4. This shows the similar design decisions against the commonly found examples of digital audio workstations shown in Figure 2.2. The DAW had two views commonly found in other platforms, a ‘timeline’ and a ‘mixer’ view (Marrington et al., 2017; Constantinou, 2019). The ‘timeline’ view, (Figure 3.4) provides a multi-track time-domain representation for the user. Each track is represented as a row, with its associated audio on that row in the ‘timeline’ (Constantinou, 2019). As time moves along during playback, the audio scrolls accordingly. By contrast the mixer view provides no time reference other than the session clock. The mixer view is used primarily for routing, effects processing and summation of tracks (the mixing phase). Users are able to add tracks, create sends and import audio.

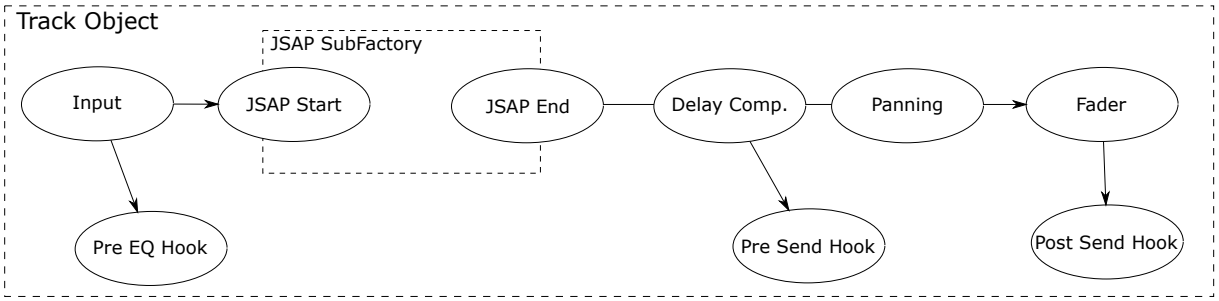


Figure 3.5: Structure of a track object in the DAW built using the Web Audio API.

3.2.2 Audio Engine and Routing

The audio workstation needs to function as a working audio production suite, therefore it needs to respond to user interactions such as volume adjustments and routing controls. This system is known as an Audio Engine, and facilitates the control of the audio rendering system. Since the web is being used the audio engine was built using the Web Audio API and JavaScript.

There are three bus types in the developed audio engine: audio, group and master. Each bus itself is a chain of Web Audio API nodes, connected together to perform specific functions and process the audio stream. Figure 3.5 shows a simplified structure of the web audio graph of an audio bus object in the developed audio engine. Depending on the bus types, the graph will have a different set of functions, for instance an input audio bus will only take audio information from files for playback, and the master back will not have any sends routing.

The audio bus type is the primary source for audio playback. It holds the audio regions, which contain the positioning information for the track in the mix. Each audio bus can hold multiple regions. It cannot accept any other form of audio input, hence its role as the source object. The audio regions themselves are simple audio buffers, which when playback is scheduled, will create BufferSourceNode objects to render the AudioBuffer into streams.

The group bus type is a routing framework which can be used for grouping or parallel processing of the audio tracks (Izhaki, 2012). It is a common task for DAWs to copy an audio signal from one track onto another, called 'send' routing (Huber and Runstein, 2005, p. 409). A send is used to copy a signal onto another bus for the purpose of adding parallel effects. This is different from an insert, where effects are placed onto the source bus directly. This is done by taking a 'send' from a source track onto a destination. The signal can be extracted from three points on a bus: pre-effect, pre-fader and post-fader. *Pre-effect* is done before the plugin processing processing chain, so is a direct, unprocessed copy of the input point to that bus. *Pre-fader* is taken after the plugin processing processing chain but before the panning and fader nodes, so has no spatial information applied. *Post-fader* is taken after the panning and fader points and is a copy of the output point of the bus.

A bus can also route its output to either the master bus or a group bus. This creates a form of grouping, since multiple tracks can be routed to a 'sub-mix' where the group bus behaves as a master (Izhaki, 2012). This

sums multiple input sources together, but its output can be routed elsewhere (e.g. to a further grouping bus or to the master).

The master bus is the final summation stage of the audio engine. This is a simplistic bus type, but it is important as the output from the master bus is sent to the Web Audio API audio context destination. This means the output of the master is routed to the default playback device from the browser and therefore played out through the user devices' audio sound card. The master does not have any send outputs, since it cannot be routed to another audio bus. Nor does the master have mute or solo functions, since it should always be active.

The DAW is constructed with the audio engine and associated user interface, allowing for engineers to operate and create a balance mix. As previous texts have stated, the balance mix needs to allow for volume and pan control over the individual audio tracks (Izhaki, 2012). Grouping and busses can also be made, as per advice from texts to engineers. Solo, muting and labelling are also supported to see how engineers approach the problem of mixing and what workflows are employed.

3.2.3 User Actions and Timings

When an engineer is operating the DAW, the actions they are performing are extracted and sent back to the data collection server, All user actions are then stored in the PostgreSQL database along with a timestamp for when the action took place. This level of granularity is not available in other DAWs without access to lower-level functions of the code base. This level of tracking is required to understand what the engineer is doing at any given point in time and how the mix decisions evolve.

HTML5 includes a set of input tags to standardise presentation across browsers, one such tag is the HTML5 slider (Casario et al., 2011). This tag allows a number range to be represented as a slider the user can interact with. This slider was used for the panning and volume controls on each track. When the user grabs the slider, the browser fires a *mousedown* event. Then as the user drags the slider, the browser fires a *oninput* event. This event is fired whenever the slider value has changed. When the user lets go the browser fires a *mouseup* event signalling the end of the interaction. Because the user could move the slider very rapidly a lot of events could be signalled at once. Therefore, only completed actions are stored, when a "mouseup" event occurs. This is a very powerful interaction to capture, since it enables the browser to know precisely when a user has released an element. This natural filtering allows two things to be assumed. Firstly, that the user has reached a decision, even if they change it very soon afterwards. They may then change their mind, but at that point they wanted to either audition the change, or change something else in relation to the decision. And secondly, that the action had completed and they might move on to another action. This is because it is impossible to click on two things at once in the user-interface. When clicking a button, the mouse up event is fired at the end, so the behaviour is not any different to knowing when the button is clicked.

The natural filtering that this applies makes it an ideal candidate for filtering the event stream, thereby pre-filtering the data before storage and analysis. Each action also happens at one point in time, so before transmitting to the database, the current Unix epoch is calculated and appended to the message. The

Session	ID	Variable	Data	Type	Time
15	14	session	15	load	2017-01-12 10:34:08.692796+00
15	15	transport	0	play	2017-01-12 10:35:13.937788+00
15	16	transport	4.589333333	stop	2017-01-12 10:35:18.529917+00
15	17	transport	0	play	2017-01-12 10:35:25.495475+00
15	18	transport	1.52	pause	2017-01-12 10:35:27.015178+00
15	19	transport	0	play	2017-01-12 10:35:28.428835+00
15	20	transport	3.248	pause	2017-01-12 10:35:30.153035+00
15	21	transport	0	play	2017-01-12 10:35:31.196964+00
15	22	transport	4.122666667	stop	2017-01-12 10:35:32.058362+00

Table 3.3: An excerpt from the Session History table of the PostgreSQL database during the development of the system.

database stores this time, allowing for comparison between actions from the start of the session. This enables frequency-of-event calculations, as well as interactions per minute and other useful metrics to be extracted.

3.3 Database storage

The data is collected from the system in real time into a relational database called PostgreSQL. The data is stored in predefined tables representing the system as a whole, to gather the data into a workable framework for analysis. When a subject loads a new session page, the session is stored in the database **session_info** table. This table holds the information such as session name, date and time created and a unique session id. The empty session is then loaded from a template, with tracks and their associated meta-data inserted into the **track_info** table. This holds the track name, instrument, type, routing information, mute and solo states, panning, volume and active states. As a starting point each track always routes to the master output, as is traditionally done in existing DAWs.

With these two tables of information, the mix state can be completely recovered as the engineer intended, just like a session file. The mixing structures can be rebuilt from each session, along with their panning and volume. To have the granularity of the data collected two further tables are needed.

The first is the **session_history** table, which holds the session level actions. Each action that is tracked in the system is sent, in real-time, to the server for storage. Each of the actions are stored in three data points per row: variable name, variable data and action type. The variable name represents which item of the session was altered, such as session loading, transport (play, pause and stop) and closing. The variable data represents the numerical state of the action. In this case, only the transport variable data is useful which reports the current session clock time in seconds, so it is possible to see which parts of the audio session the users were most interested in. The action type is a vowel based string to represent the session action. For instance, a track could either be added or removed, so the action type for a track action is ‘add’ or ‘remove’. For transport,

Session	Track	ID	Variable	Data	Type	Time
16	195	253	pan	0	set	2017-01-12 10:50:10.636069+00
16	196	254	volume	-3.333333487	set	2017-01-12 10:50:18.590316+00
16	197	255	volume	-4.958333791	set	2017-01-12 10:50:20.164578+00
16	198	256	pan	-22	set	2017-01-12 10:50:27.272535+00
16	198	257	volume	-3.708333695	set	2017-01-12 10:50:29.406886+00
16	199	258	volume	6.416666505	set	2017-01-12 10:50:32.537745+00
16	199	259	pan	3	set	2017-01-12 10:50:34.916963+00
16	197	260	pan	77	set	2017-01-12 10:50:37.222482+00
16	196	261	pan	72	set	2017-01-12 10:50:39.470574+00

Table 3.4: An excerpt from the Track History table of the PostgreSQL database during the development of the system.

as shown in Table 3.3, it represents whether the transport action is to start playing ('play') or pause or stop auditioning the session.

The track controls are also stored in **track_history**. This table holds every action an engineer makes on any of the tracks in the sessions. Because of the nature of track actions being more expansive than session actions, the number of data sets tracked is far higher. In the except in Table 3.4 a set of actions can be seen covering the manipulation of the several tracks volume and pan controls in a session. As with the session history in Table 3.3 the actions are stored in three stores: variable name, variable data and action type. The actions of pan and volume store the parameter controls, pan as -180 to +180 for hard left and hard right respectively, and volume as decibels. Mute and solo are also stored as Boolean 0 and 1 for 'Off' and 'On'.

Because the data can be queried by the unique track ID and its type, it will be possible to extract how a parameter has changed over time. Likewise the time stamps will allow for deeper interrogation of what actions occur when, such as what actions occur when the session is playing or not. Or what actions occur together or at similar stages of the mix. This level of details has not been explored and made available as a public data set before.

All of this data can be accessed as a query to give a real-time playback of the mixing decisions over time. Showing not only what the alterations to the mix state was, through panning and volume controls, but also how much of the system the engineer was listening to when the actions were performed, which state the system was in and at what time in the session these would be performed. This fine level of granularity will allow the collection of novel data on how engineers complete certain mixing tasks and challenges, including the balance mix.

3.4 Conclusion

Several data collection methodologies have been discussed and evaluated for data collection purposes. Surveys have been used to collect broad levels of information from many subjects quickly, however this would give the same data as had been discovered previously without being able to give more detail on the actions. Video recording and other observational systems could have been employed, to record what engineers are doing in a real studio under laboratory conditions. This would require extensive time commitments to transcribe the video content and actions into a usable structure for further research (Sutton and Austin, 2015). Therefore a novel set of data-collection tools were developed to obtain this unique data set from as many real-world engineers as possible.

The Web Audio API allows for low-level manipulation of audio content in the browser, previously reserved for closed-source development environments (World Wide Web Consortium, 2018). This step forward, along with improvements of web connectivity globally, allows for the development of a novel set of data collection tools. Using the Web Audio API a Digital Audio Workstation was developed for the browser. This tool allowed for most basic DAW actions to be undertaken, such as volume and panning, mute, solo, grouping and transport controls. Care and attention is needed to ensure the system developed is familiar to the engineers it is being presented too, such that they are not frustrated with the system or cannot use the system to give adequate feedback. The system recorded all user actions back to the remote data collection server over the web, to be stored in a PostgreSQL relational database as a set of time-based entries. These entries would allow for replaying of the mix from start to finish for further analysis, with each action recorded.

The Web Audio Evaluation Toolbox was developed to create a remote deplorable listening test environment in the browser (Jillings et al., 2015). This system enables a researcher to build and deploy a listening test quickly, whilst confirming to listening test standards such as MUSHRA (ITU-R, 2015). Perceptual listening tests are used extensively in audio to evaluate the performance of a system with real-world listeners, such as for sound synthesis (Selfridge et al., 2017a) or noise level evaluation (Labairu-Trenchs et al., 2018).

The mixes, along with their action data, can be examined in studies on how mix engineers approach and manage a session. Along with ability to rebuild the mix at any given point, the mixes themselves can be evaluated over time. The Web Audio Evaluation Toolbox can be used to compare the created mixes against each other. This will allow mixing practices which result in better mixes to be discovered as a ranking of each mix can be uncovered and used.

Given the data collection methodology is designed and developed, it can be used to gather the mixing information directly from engineers to understand the balance mix process. It will provide a unique level of detail that would have been previously prohibitive to obtain. The tool is used in Chapter 4 to gather the data for further analysis to answer the question on how engineers approach and complete the task of the balance mix. The Web Audio Evaluation Toolbox is used to verify the engineers mix against perceptual scores, so the decisions can be verified. The listening tests are also used to verify the balance mixing task in Chapter 6.

Chapter 4

An Investigation of Current Mixing Practices

In this chapter, we present a series of experiments which attempt to formalise the process of balance mixing. In section 2.1 an overview of popular texts is given, which discuss the importance of the balance mix and how the information for engineers can be conflicting. Section 4.1 presents the research methodology used to gather the data, using the tools described in Chapter 3. The results from the study are presented in Section 4.2 with more detailed analysis and discussion in Section 4.3.

4.1 Methodology

This study was designed to capture engineer operations when creating a balance mix using the web-based Digital Audio Workstation presented in Chapter 3. The platform is intended to replicate familiar control surfaces, minimising any learning time for engineers. This should make the experience of the participants as close to their real-world mixing tasks as possible. All user interactions are tracked through the system and logged into a database. This provides a time-series set of session data giving a granular insight into the mixing process. This will contribute to the relatively few data sets that currently only capture static session data (De Man et al., 2014b). The study aims to answer the following questions surrounding mixing performances.

- How do engineers approach a balance mixing session?
- What control structures are used in the session?
- How does the user interact with the graphical user interface?
- Are there any similarities in the final mixes?

The experiment was designed to simulate the balance mix stage of the production process. This is a specific mixing task performed after recording to prepare the session for the full mixing stage. After this point, engineers will perform minor corrections to the audio whilst starting to organise the sound stage of the mix. To simulate this process, the participants were limited to controlling a handful of parameters for each session:

Name	Artist	Genre	Number of Tracks
Left Blind	Hollow Ground	Death Metal	16
Queen's Light	Dino On The Loose	Jazz Fusion	17
Sleigh Ride	The Funny Valentines	Live Jazz	7
The English Actor	James Elder & Mark M Thompson	Indie Pop	17
I'm Alright	Angels In Amplifiers	Acoustic Rock	12

Table 4.1: Details of the five songs selected Mixing Secrets of a Small Studio site used in the study (Senior, 2019).

- Session Transport: start and stop playback
- Track Volume: the gain of the track
- Track Panning: the stereo position of the track
- Track Solo/Mute: the mute and solo of each track
- Create Busses: create new routing busses for tracks
- Create Sends: create routing to each bus from a source track

Five multi-tracks were presented to the user, obtained from the Mixing Secrets of a Small Studio site (Senior, 2019). The five songs chosen were used as they cover several genres without being too large, that the study would be onerous for the mix engineers to complete. Details of the songs are shown in Table 4.1. Each song was truncated to a 30 second excerpt from the chorus to reduce the duration of the experiment for end users, reducing the chance of listener fatigue. The chorus was chosen to ensure the most complex part of the mix was used, as opposed to the intro or a verse which, whilst valid, may be less challenging. The mixing parameters for five different songs were then initialised to two different starting positions, with each track given a random pan position and fader value. The two starting parameters mean when a new participant loads the session, they would load either the A or B starting position, allowing for analysis on whether the starting position has a noticeable impact at this stage, as shown in previous research studies (Wilson and Fazenda, 2015b). The test was performed online, with participants being obtained through audio research focused mailing lists.

Once all the mixes had been captured providing the mixing action data, a second data collection phase was performed to collect the subjective data on the mixes. The data was collected using the Web Audio Evaluation Toolbox introduced in section 2.7.3 (Jillings et al., 2015). Each subject performed the test remotely with instructions to perform the test in quiet conditions. To ensure that subjects could be rejected from the study, each subject filled out a pre-test survey to understand if they would be reliable participants. The questions presented were:

- Please select the option that best describes your listening environment: [*Studio, Personal Hi-Fi, Headphones*]
- Does your room have acoustic treatment?

Song	No. Tracks	Group	Participants	Avg. actions	Avg. duration (mm:ss)
I'm Alright	12	1	4	87.50	10:03.94
I'm Alright	12	2	2	86.00	05:18.67
Left Blind	16	1	1	125.00	07:00.56
Left Blind	16	2	2	175.50	30:58.36
Queens Light	16	1	5	126.60	59:17.89
Queens Light	16	2	5	173.20	18:52.22
Sleigh Ride	7	1	2	76.0	06:25.42
Sleigh Ride	7	2	1	51.0	03:06.31
The English Actor	17	1	1	115.00	06:28.94
The English Actor	17	2	2	139.00	31:11.79
Total (overall)	**353**		**35**	**3391**	**10:00:30.59**

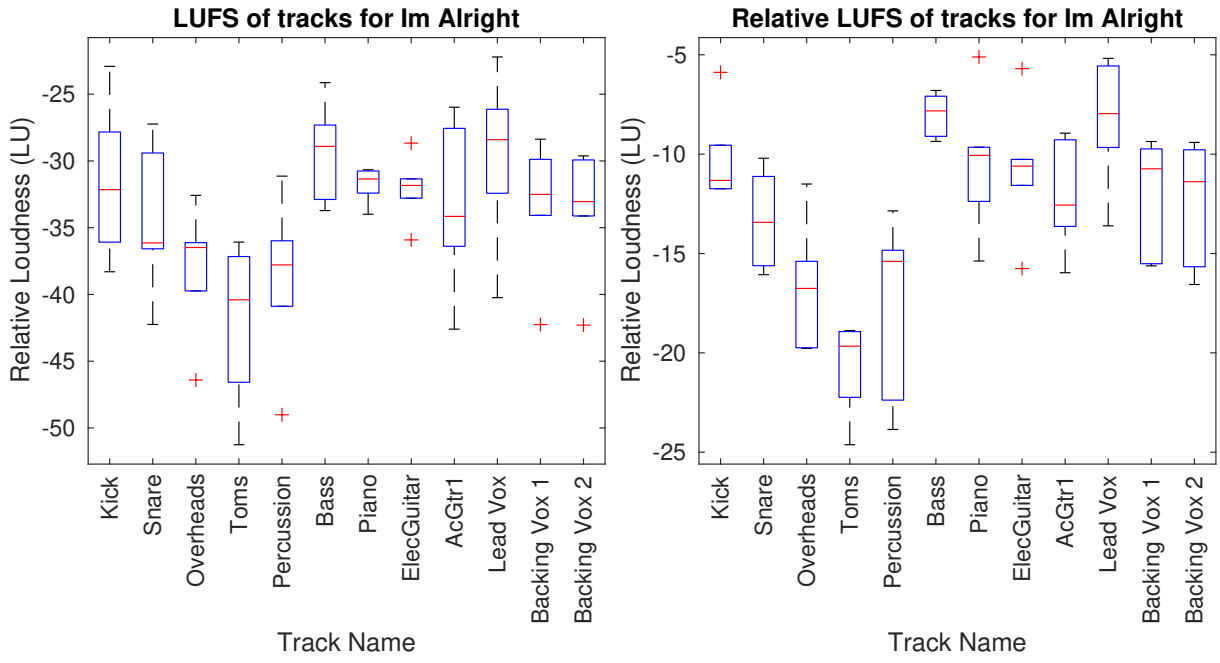
Table 4.2: The summary of the final sessions after filtering, with the total overall values in bold.

- Are your speakers full range?
- Please select the following to describe your headphones. In-ear,Circumaural, supra-aural, other, unknown.
- Please enter your experience in audio production (studio work) in years.
- Do you have a hearing impairment?
- Can you expand upon the hearing difficulties you have? (This question is not compulsory and can be skipped).
- Have you participated in listening studies before?
- Have you participated in this study before?

The subjects were presented with the listening test using the Audio Perceptual Evaluation (APE) interface (De Man and Reiss, 2014). The fragments were presented according to the mix they originated in. One page of fragments contained only Left Blind, another only Sleigh Ride. Each page also included the two randomised starting positions as presented to the original mix groups. This would provide a form of anchor, although no true anchor exists for this type of test, where it should be expected that each engineer would improve the quality of the given mix. The order of the test pages was randomised for each participant as well as the presentation of the fragments to reduce bias.

4.2 Results

Over the experiment there were 71 mix submissions using the online DAW. All web based experiments must be appropriately filtered to remove any noise or bias by poor candidates (Cartwright et al., 2016). To do this, we used the action count, as it roughly correlates with the amount of effort applied by the engineer. The threshold for the action count was set as double the track count (2 interactions per track). Each session took on average



(a) Loudness of each track across all mixes

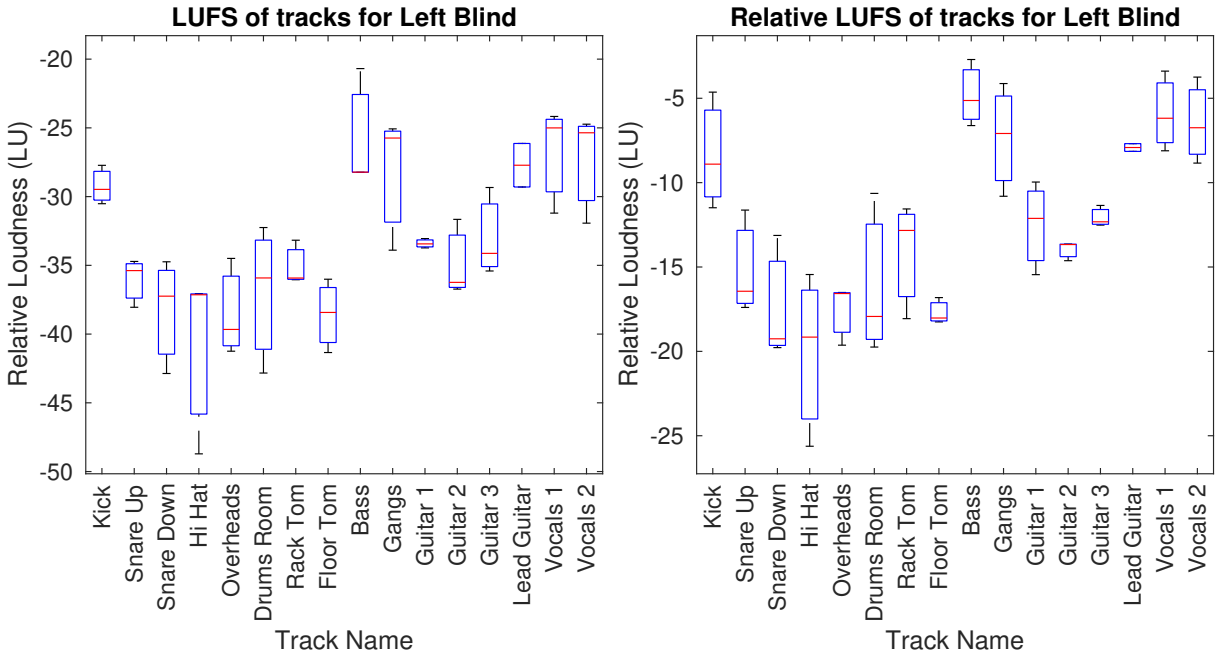
(b) Loudness normalised to the loudness of the final mix

Figure 4.1: Loudness score of each track in **Im Alright** in LUFS and Relative Loudness in LU.

between 8 and 16 minutes to complete with 100 to 150 actions per session. The final 35 mixes after filtering are shown in Table 4.2.

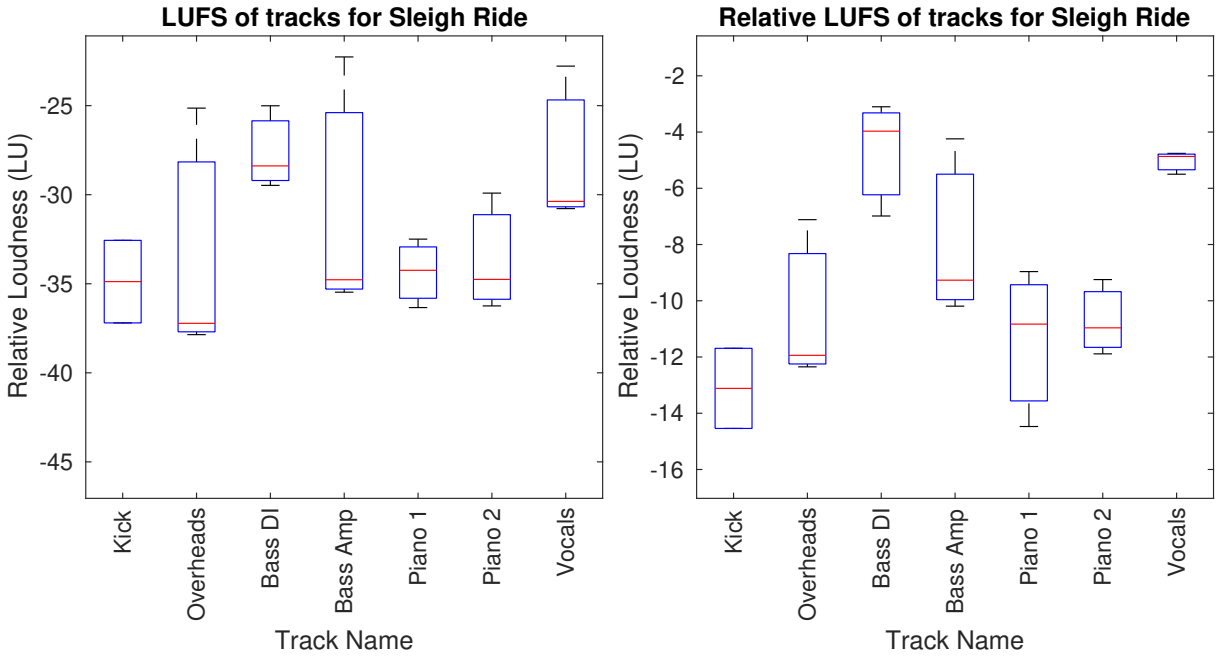
The most important aspect of the mixing process is the relationship between the different tracks, namely their perceptual loudness. The loudness of the track will impact its perceived importance to the engineer, as louder tracks generally sound more prominent in the mix. To illustrate the average gain structures used by participants, Figures 4.1 through 4.5 show the relative loudness measurements for each track against the overall session loudness. This is calculated using the ITU Recommendation 1770 (International Telecommunication Union, 2011) to obtain a LUFS value for each track. Whilst the LUFS calculation is designed primarily for broadcast material it performs better than RMS in perceptual listening tests, making it a suitable candidate for broad and varied music sources. Whilst there are criticisms on LUFS being used for narrowband sources, there is no single measurement or correction which conclusively proves to improve on all situations. The use of LUFS also provides consistency with previous studies exploring the same phenomenon (Wilson and Fazenda, 2015a).

LUFS is calculated by passing the audio through a filter which very crudely estimates the loudness curves of the human ear, and is based upon perceptual listening tests. Then, the audio RMS levels are calculated and framed to form a gate. All frames which have an energy level of less than -70DB LKFS (absolute loudness) are dropped from the calculation, and then an average loudness level is taken. From the remaining average loudness -10dB forms the adaptive loudness gate. All frames which are above this loudness gate can pass through and the remaining frame average gives the loudness level in LUFS, which is a dB unit. Because it is related to RMS levels, a change in the fader levels in dB directly maps to a change in the LUFS level in dB. The figures show two plots for each session, one in LUFS and the other in LU. The LUFS measurement is the



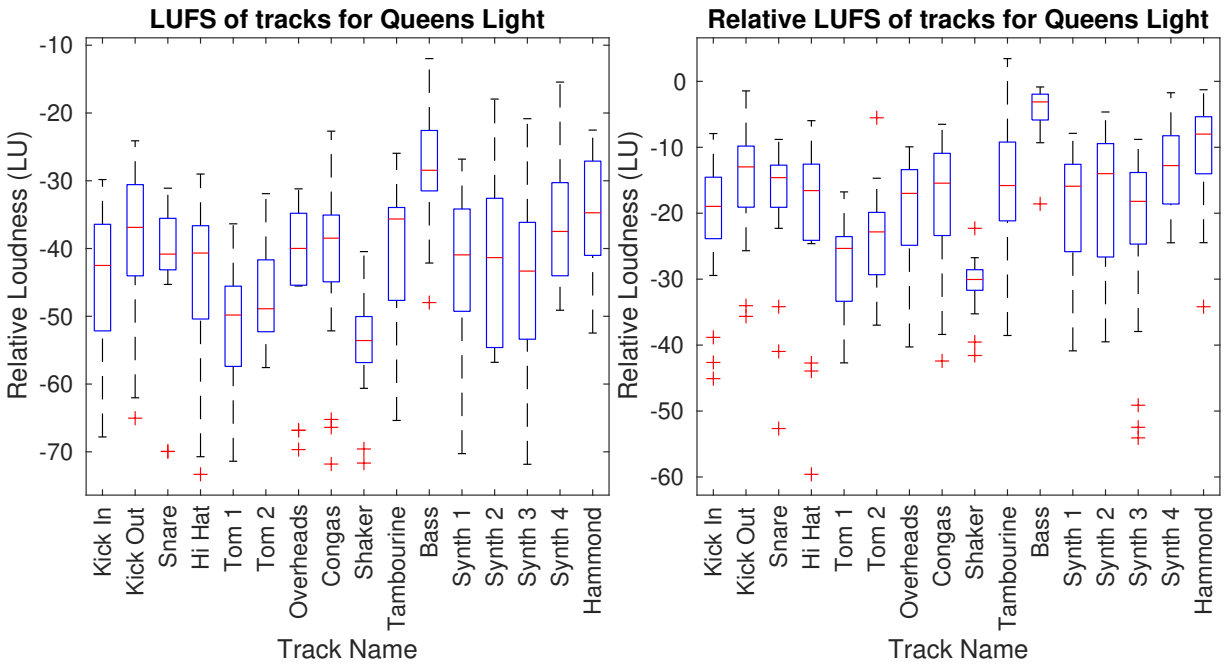
(a) Loudness of each track across all mixes (b) Loudness normalised to the loudness of the final mix

Figure 4.2: Loudness score of each track in **Left Blind** in LUFS and Relative Loudness in LU.



(a) Loudness of each track across all mixes (b) Loudness normalised to the loudness of the final mix

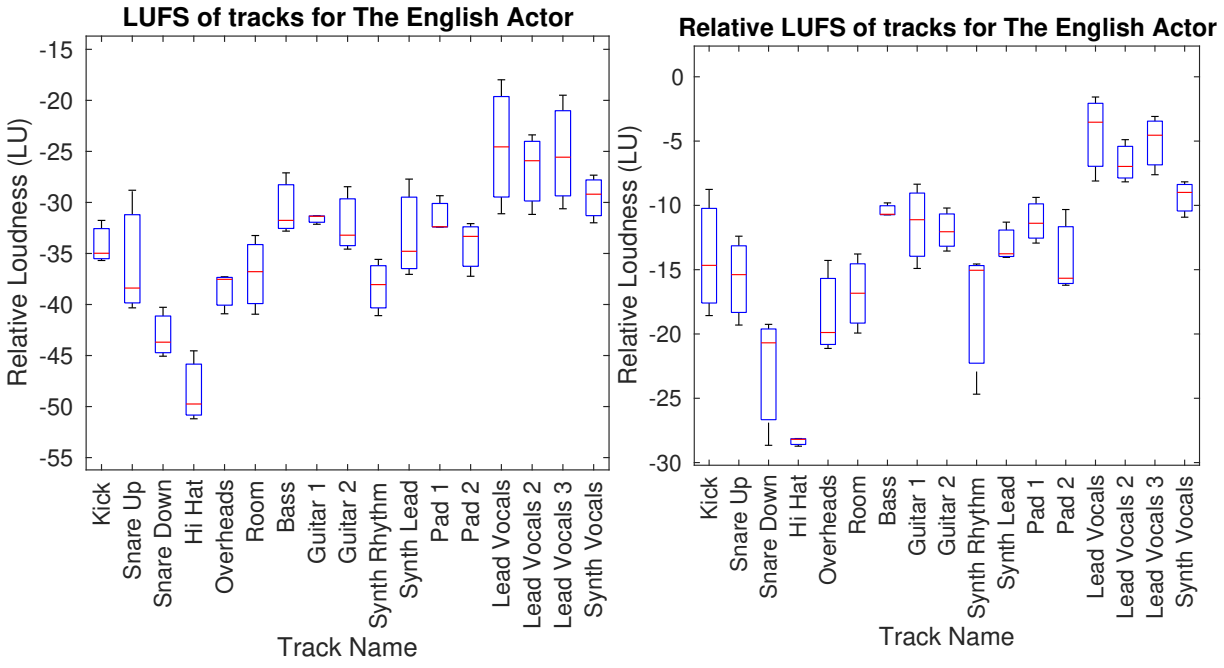
Figure 4.3: Loudness score of each track in **Sleigh Ride** in LUFS and Relative Loudness in LU.



(a) Loudness of each track across all mixes

(b) Loudness normalised to the loudness of the final mix

Figure 4.4: Loudness score of each track in **Queen's Light** in LUFS and Relative Loudness in LU.



(a) Loudness of each track across all mixes

(b) Loudness normalised to the loudness of the final mix

Figure 4.5: Loudness score of each track in **The English Actor** in LUFS and Relative Loudness in LU.

Listening Environment	Number
Headphones	37
Studio	14
Personal Hi-Fi	7

Table 4.3: Listening Environments as reported by the subjects in the listening test.

Headphone Type	Number
Circumaural (Closed)	19
Supra-Aural	5
Circumaural (Open)	4
In-ears	8
Fitted In-ears	1

Table 4.4: Headphone types as reported by the subjects in the listening test.

direct loudness value of that track. The LU is the relative loudness and is the difference between the track loudness and the overall mix loudness.

4.2.1 Listening Test

A total of 61 individuals participated in the study over a two week period. Table 4.3 shows the listening environments as reported by the subjects. The majority of the subjects were using headphones to conduct the listening test. These headphone types were broken down into 5 different categories that the participants could use, with the most common being closed-backed Circumaural headphones, as shown in Table 4.4. This shows most of the participants who were using headphones were using high-quality equipment, since Circumaural and Supra-Aural are more desirable. The subjects who performed a test using a Studio or Personal Hi-Fi were asked if their room was treated acoustically. Table 4.5 shows that 13 said yes, but 10 said no. This is not a reason for exclusion since treatment is not always necessary or helpful. Of the 23 participants using speakers, 16 declared using full range speakers.

11 participants reported existing hearing conditions which may influence the test. Alongside asking if they had a known condition, they were asked to give details. Most of these were for mild symptoms, such as tinnitus, however some were more extreme which could interfere with the critical listening study. The four most severe were removed, who declared having partial / complete deafness in one or both ears. 53 of the

Treated Room	Number
Yes	13
No	10

Table 4.5: Room treatment for those in a studio or Hi-Fi environment, as reported by the subjects in the listening test.

Speakers Full Range	Number
Yes	16
No	7

Table 4.6: Number of subjects using full-range speakers, as reported by the subjects in the listening test.

Hearing Impairment	Number
Yes	11
No	49

Table 4.7: Number of subjects which reported having a hearing impairment.

Listening Test Experience	Number
Yes	53
No	7

Table 4.8: Number of subjects which reported having participated in a listening test previously.

Years Production Experience	Number
Less than 1 year	14
1-2 years	7
2-3 years	5
3-4 years	3
4-5 years	7
5+ years	24

Table 4.9: Years of music production experience, as reported by the subjects in the listening test.

Page Index	Average Duration	Number of Subjects	Page Errors
1	3 min 53 s	32	7
2	4 min 3 s	29	11
3	2 min 45 s	29	6
4	3 min 3 s	29	5
5	5 min 41 s	29	14
6	2 min 20 s	29	3

Table 4.10: The average duration per page as presented to the subject during the listening test.

Page Name	Average Duration	Number of Subjects	Number of Fragments	Page Errors
Queen's Light (1)	5 min 28 s	29 subjects	12 fragments	7
Queen's Light (2)	6 min 14 s	29 subjects	12 fragments	11
I'm Alright	3 min 32 s	29 subjects	8 fragments	3
Left Blind	2 min 27 s	29 subjects	5 fragments	6
Sleigh Ride	1 min 49 s	30 subjects	5 fragments	9
The English Actor	2 min 25 s	31 subjects	5 fragments	10

Table 4.11: The average duration per page per the testing configuration to the subject during the listening test, along with the number of fragments.

participants stated they had been in previous listening study's before. After filtering of the participants, a total of 55 participants remained. Of the 55 participants, a total of 177 pages and 1,378 fragment evaluations were performed. This shows on average each participant completed 3.21 pages before leaving the test. Subjects can abandon tests for many reasons, but usually high difficulty of tests and frustration can cause them to leave. Therefore it is important to ensure the test is not only randomised but balanced in such a way that the least performed pages are presented to the next subject first, to ensure that there is a strong level of consistency between the participants. The average time to complete each page was 3 minutes and 38 seconds, meaning the full test should have taken, on average, 21 minutes 38 seconds.

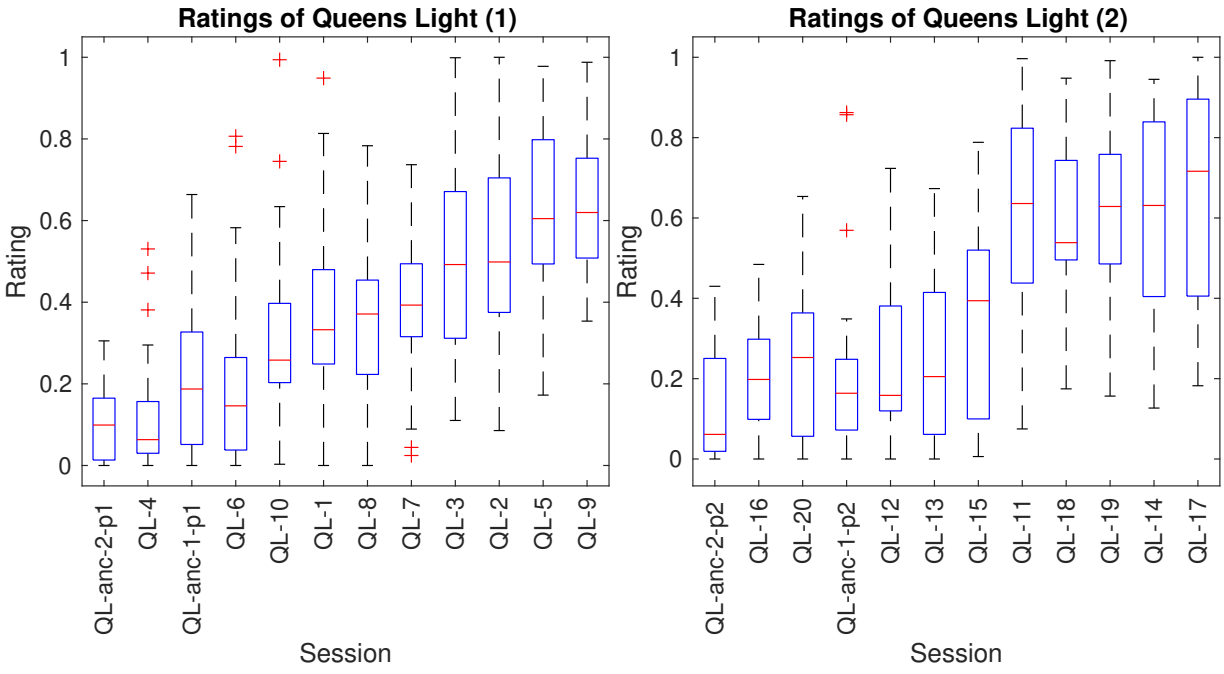
Each page used the APE testing framework and asked each participant to rank each mix. The question proposed was "Which mix do you like the most?". This question is open to subjective bias, for example if a participant does not like the song at all they may degrade the entire mix. To prevent this, the participants are only allowed to complete a page and move to the next song if the following criteria is met.

- At least one mix must be above 0.75 (Like very much) and below 0.25 (Dislike very much)
- All fragments must have been moved at least once
- All fragments must have been played at least once

These constraints require the subject to perform the test properly by ensuring every fragment has been fairly evaluated. Table 4.11 shows the number of subject pages which experienced at least one of these page errors.

Number of Fragments	Number of Errors
1	24
2	8
3	4
4	1
5+	8

Table 4.12: Summary of total number of not moved fragments on the page when the subject tried to submit the page.



(a) Group 1 (b) Group 2

Figure 4.6: Ratings from the listening test for the song Queen's Light

The average number of submission errors was 25.99% of all submitted pages, indicating there was a large amount of frustration. Table 4.12 shows the number of not moved fragments in the page when the subject tried to submit the page. This shows that for the majority of subjects, it was only one or two fragments which were not moved.

The results of the listening tests are presented in Figures 4.6 through 4.10. The Queen's Light test was split over two pages because there were 20 songs produced, which would create a very large page for the subjects to compare. Each page had 10 fragments, with 5 from group A and 5 from group B. Whilst each fragment individually was not compared with every other fragment, the system should still produce a suitable metric for comparison. On each page, the two anchor fragments represent the original starting positions. Tables 4.13 through 4.18 give the Wilcoxon Rank Sum test (Wilcoxon, 1945). This tests for the null hypothesis that, for randomly selected values from two distributions, the probability of X being greater than Y is equal to the

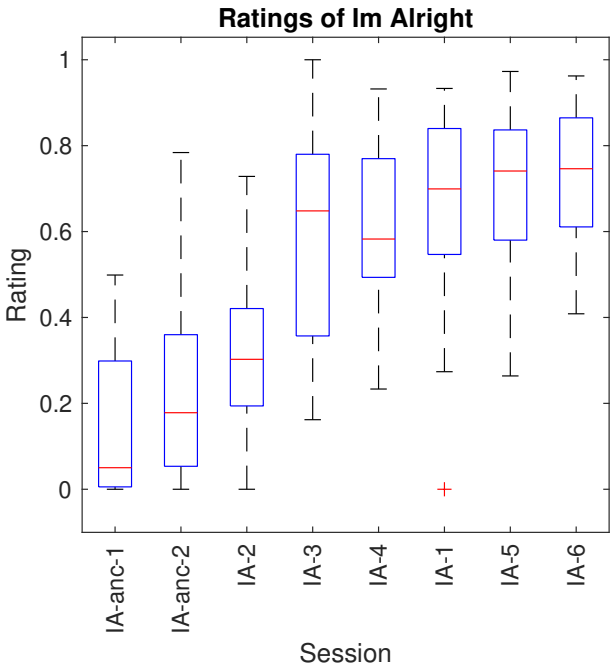


Figure 4.7: Ratings from the listening test for the song I'm Alright

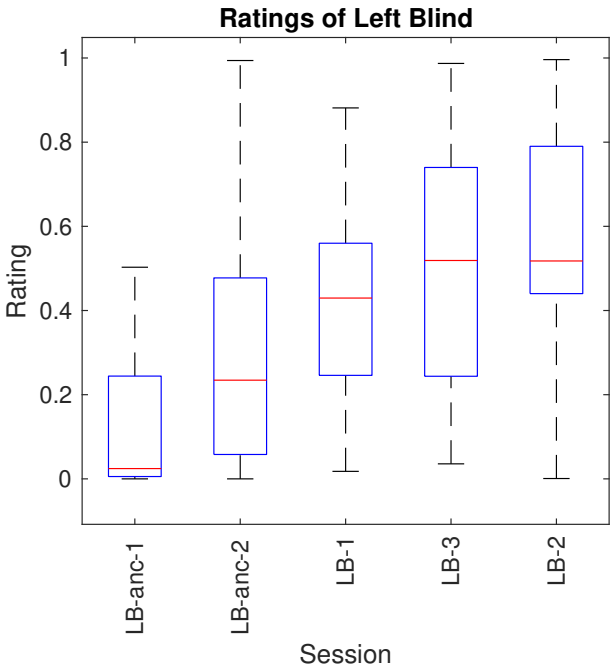


Figure 4.8: Ratings from the listening test for the song Left Blind

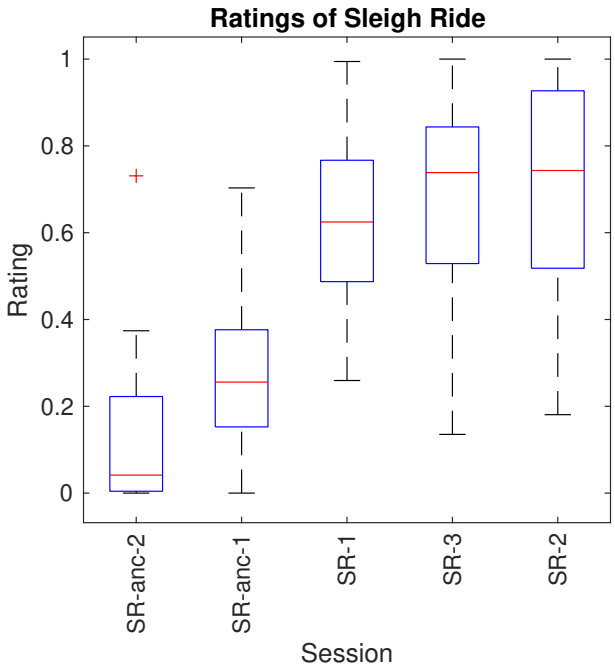


Figure 4.9: Ratings from the listening test for the song Sleigh Ride

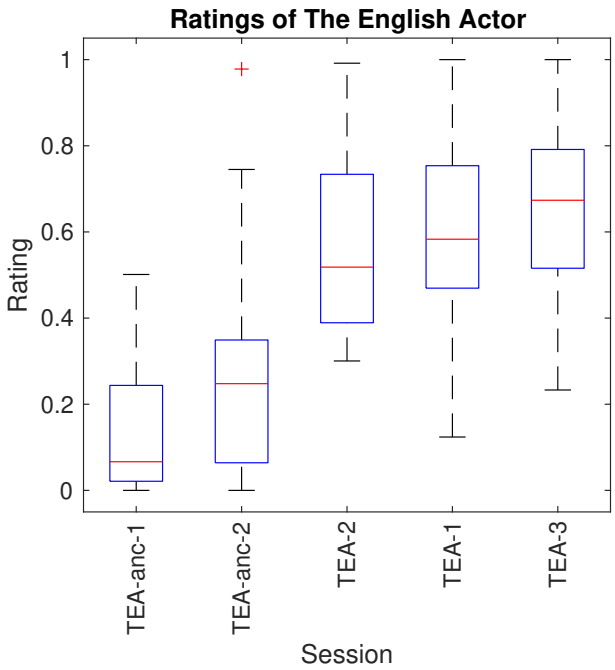


Figure 4.10: Ratings from the listening test for the song The English Actor

Mix	Anchor 1	Anchor 2	Null test pass
QL-4	0.015	0.700	No
QL-6	0.586	0.070	No
QL-10	0.071	<0.001	No
QL-1	0.003	<0.001	Yes
QL-8	<0.001	<0.001	Yes
QL-7	<0.001	<0.001	Yes
QL-3	<0.001	<0.001	Yes
QL-2	<0.001	<0.001	Yes
QL-5	<0.001	<0.001	Yes
QL-9	<0.001	<0.001	Yes

Table 4.13: Wilcoxon Rank Sum test for each mix performed on the Queens Light group in figure 4.6a.

Mix	Anchor 1	Anchor 2	Null test pass
QL-16	0.542	0.027	No
QL-20	0.707	0.069	No
QL-12	0.690	0.016	No
QL-13	0.778	0.038	No
QL-15	0.080	<0.001	No
QL-11	<0.001	<0.001	Yes
QL-18	<0.001	<0.001	Yes
QL-19	<0.001	<0.001	Yes
QL-14	<0.001	<0.001	Yes
QL-17	<0.001	<0.001	Yes

Table 4.14: Wilcoxon Rank Sum test for each mix performed on the Queens Light group in figure 4.6b.

Mix	Anchor 1	Anchor 2	Null test pass
IA-2	0.004	0.059	No
IA-3	<0.001	<0.001	Yes
IA-4	<0.001	<0.001	Yes
IA-1	<0.001	<0.001	Yes
IA-5	<0.001	<0.001	Yes
IA-6	<0.001	<0.001	Yes

Table 4.15: Wilcoxon Rank Sum test for each mix on the I'm Alright song in figure 4.7.

Mix	Anchor 1	Anchor 2	Null test pass
LB-1	<0.001	0.05	Yes
LB-3	<0.001	0.001	Yes
LB-2	<0.001	<0.001	Yes

Table 4.16: Wilcoxon Rank Sum test for each mix on the Left Blind song in figure 4.8.

Mix	Anchor 1	Anchor 2	Null test pass
SR-1	<0.001	<0.001	Yes
SR-3	<0.001	<0.001	Yes
SR-2	<0.001	<0.001	Yes

Table 4.17: Wilcoxon Rank Sum test for each mix on the Sleigh Ride song in figure 4.9.

Mix	Anchor 1	Anchor 2	Null test pass
TEA-2	<0.001	<0.001	Yes
TEA-1	<0.001	<0.001	Yes
TEA-3	<0.001	<0.001	Yes

Table 4.18: Wilcoxon Rank Sum test for each mix on The English Actor song in figure 4.10.

probability of Y being greater than X. If this is the case the two distributions are not related and therefore significantly different. Table 4.13 and 4.14 give the scores for the song Queens Light. As can be seen, more than half the mixes produced were rated significantly higher than the anchor mixes. Performing the Wilcoxon rank test, only mixes QL-4, QL-6 and QL-10 were not significantly different than the anchor in the first listening test group. And in the second QL-12, QL-13, QL-15, QL-16, and QL-20 were not significantly different than both the anchors, with high probabilities that the result distributions are related. This shows that the samples, whilst different, are similar in the metric being evaluated, that being which is the subjectively better mix. In I'm Alright, table 4.15, all of the mixes except IA-2 performed significantly better than both anchor mixes. For Left blind in table 4.16, Sleigh Ride in table 4.17 and The English Actor in table 4.18, all three of the completed mixes performed significantly better than the two anchor mixes.

4.3 Discussion

This section aims to answer the four research question proposed at the start of this Chapter. In each subsection, different parts of the results are analysed and compared against past studies and known references. The questions aimed to cover the aspect of mixing focused on creating and producing the balance mix.

Action Type	1st Action	2nd Action	3rd Action
Transport	31 (88.57%)	22 (62.86%)	11 (31.43%)
Volume	2 (5.71%)	8 (22.86%)	7 (20.00%)
Solo	2 (5.71%)	3 (8.57%)	9 (25.71%)
Mute	0 (0.00%)	2 (5.71%)	3 (8.57%)
Pan	0 (0.00%)	0 (0.00%)	3 (8.57%)
Create Track	0 (0.00%)	0 (0.00%)	2 (5.71%)

Table 4.19: The first three actions for the filtered sessions

4.3.1 How do engineers approach a mixing session?

The first question the study aimed to uncover regards how an engineer begins the mixing process. Because the system stored the actions performed and the time the action was performed, it was a trivial process to extract the initial data points. Table 4.19 shows the first three actions in each session. Texts suggest that engineers that are not familiar with the mix, either because they were not part of the recording process or composition, should listen to the content first before performing any other actions (Izhaki, 2012, p. 35). Since the sessions are randomised and unlabelled, even if the engineer was familiar with the piece initially, they would not know what the mix is meant to be, without playing the song first.

Once the session had been loaded, 88.57% of the first actions by the subjects were to begin playback. The study was performed blind and no engineer had any prior knowledge of the mixing stages, suggesting this practice was adopted. This clearly shows engineers will listen to the audio content before making any mixing decisions, even if they are structural. Once the engineer has listened to the track, they begin to make mixing decisions based on what was exposed to them. By the third action several mixing decisions are being made as the search process begins. Structural changes are starting to be performed, where two sessions already had a bus being made. The volume changes, whilst high, are not as important as the Solo and Mute actions which account for the majority of the third actions. This shows the importance at this phase for the engineers to isolate the mix and focus in on certain areas immediately.

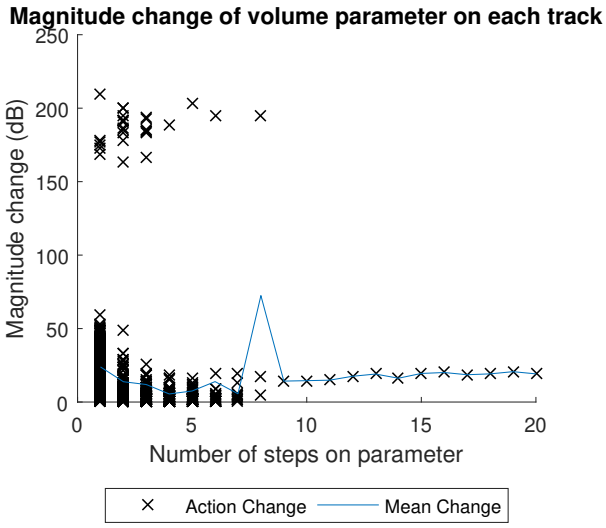
Table 4.20 shows the total action count for parameters used in the experiments. An action is counted if it has occurred between the loading of the page and the closing of the page, to determine when the session started and stopped. Of the 35 filtered sessions, 3391 individual actions were logged by the system. An action is only recorded when the *mouseup* event in the browser fires. This means when the user drags a control around in the space, only the final landing position is logged and recorded as an action. The highest recorded action is volume control, indicating a significant amount of importance is placed on this individual action type. This action indicates there are at least 3 movements of the fader for each track. The volume is therefore the most critical aspect of this mixing stage, above setting the pan position which only accounts for 12.33% of all the actions in the session. Starting and stopping playback is clearly very high, which makes intuitive sense since the task requires listening to the audio critically. Additional mixing tools like solo and mute also account for

Action Type	Number
Volume	1094 (32.26%)
Transport (Play/Stop)	980 (28.90%)
Pan	418 (12.33%)
Solo	387 (11.41%)
Adding to Group	210 (6.19%)
Mute	102 (3.01%)
Add Send	82 (2.42%)
Create Group	67 (1.98%)
Send Volume	30 (0.88%)
Send Phase	11 (0.32%)
Rename Track	8 (0.24%)
Send Mute	2 (0.06%)

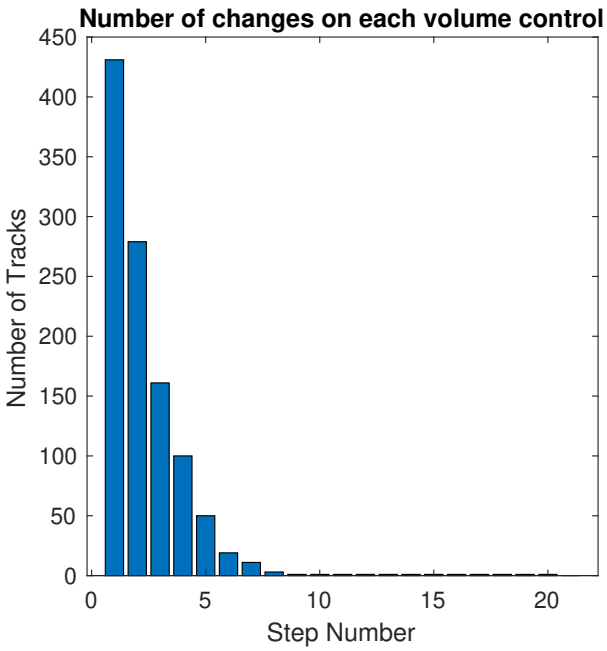
Table 4.20: The total action counts for the 35 filtered sessions

Instrument	1st Action	Mean to 1st Action
Drums	25	38.11s
Bass	5	21.91s
Keys	3	81.38s
Vocals	2	48.43s
Percussion	0	NA
Guitar	0	NA

Table 4.21: The instrument which had the first track based action to occur in the session



(a) Magnitude of volume interactions in dB



(b) Count of tracks having a number of volume actions

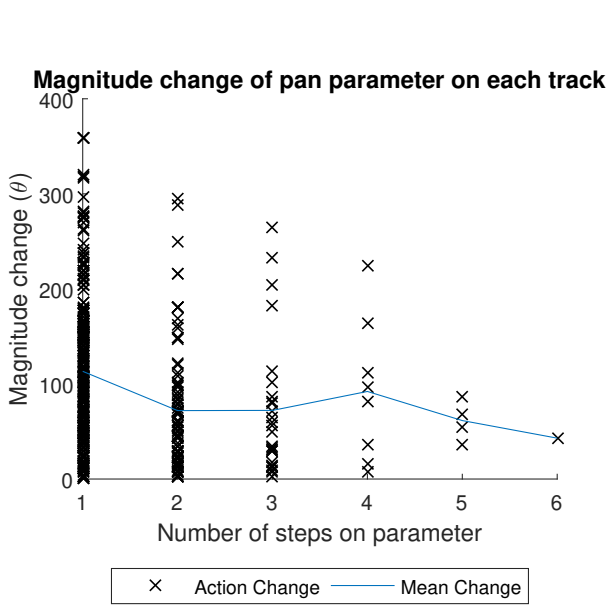
Figure 4.11: Magnitude of volume interactions over steps

14.080%, suggesting they are of high importance, as the engineers can dissect the audio to listen to specific components of the mix.

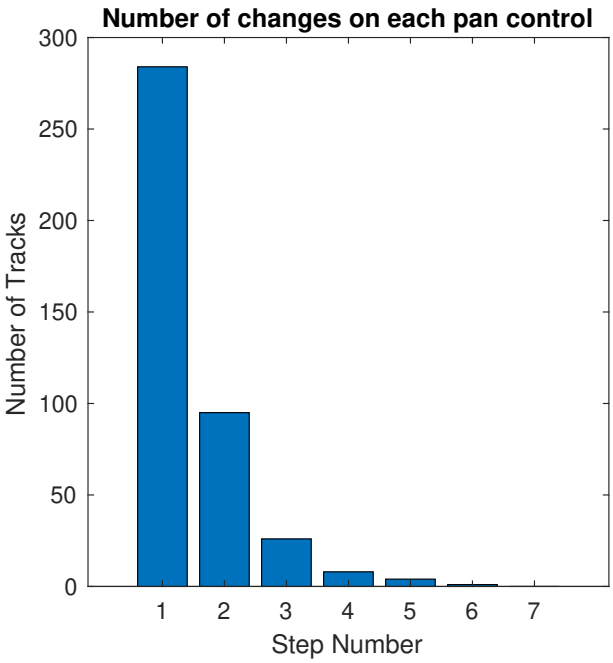
The first track the engineer chooses to interact with is presented in Table 4.21. This shows that Drums are extremely likely to have the first track based interaction occur on them, followed by Bass. In none of the 35 filtered sessions were the Percussion of Guitar tracks the first track to be interacted with. Interestingly, across all the tracks, the time to the first action averaged at 47.46 seconds. Given each song is 30 seconds long, this time gap would indicate that the engineer does listen to the whole track before starting to mix.

Since the act of mixing is compared to an iterative process, where each decision should take the mix closer to the desired end state, each action should result in smaller changes over time. Each track has one pan and one volume control associated. The pan has a numerical value of -180 to +180 (numerical range of 360) and the volume a range of -200dB to +12dB. By taking each pan and volume control individually, and extracting the sequence of changes made, it is possible to show how the magnitude the change in actions gets smaller as more changes are made. Figures 4.11 and 4.12 show this magnitude change over time for the volume and pan controls respectively. The step count indicates how many individual tracks had that many changes occur on its pan and volume control. As shown in Table 4.20 there are over twice as many volume changes as there are pan changes recorded. Equally, no single pan dial was changed more than six times, as opposed to the volume where one control was changed more than twenty times.

Comparing the magnitude of changes against the time of change relative to the session start time shows a different story. Figure 4.13 gives this change information for both volume and panning. As can be seen, the volume magnitudes decrease over time on average with a relatively strong correlation. The large magnitude

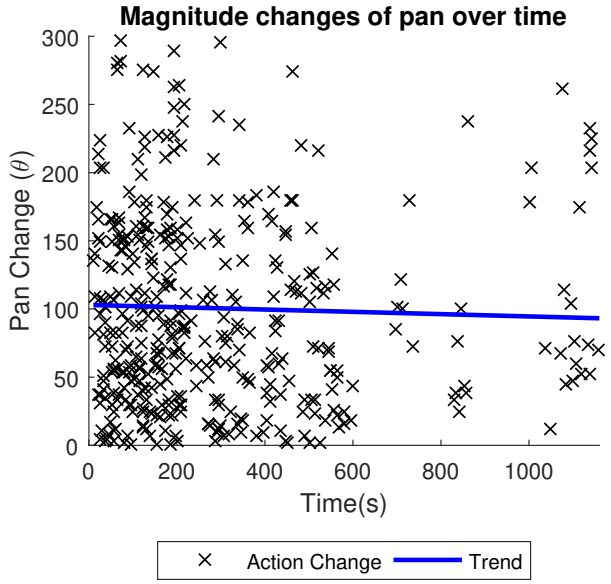


(a) Magnitude of pan interactions in degrees

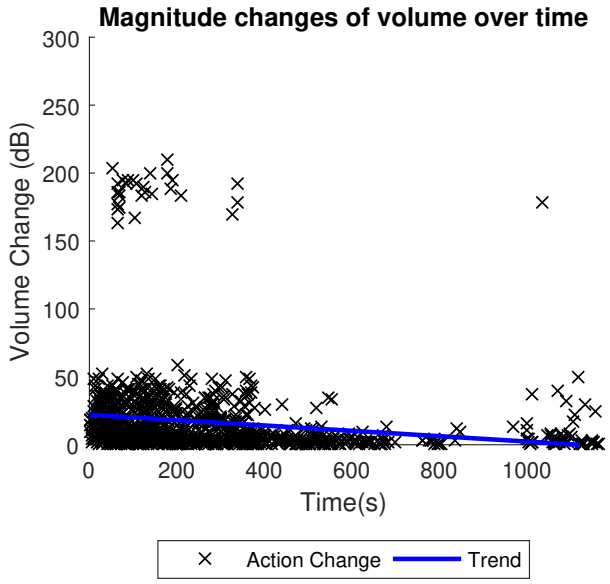


(b) Count of tracks having a number of pan actions

Figure 4.12: Magnitude of panning interactions over steps

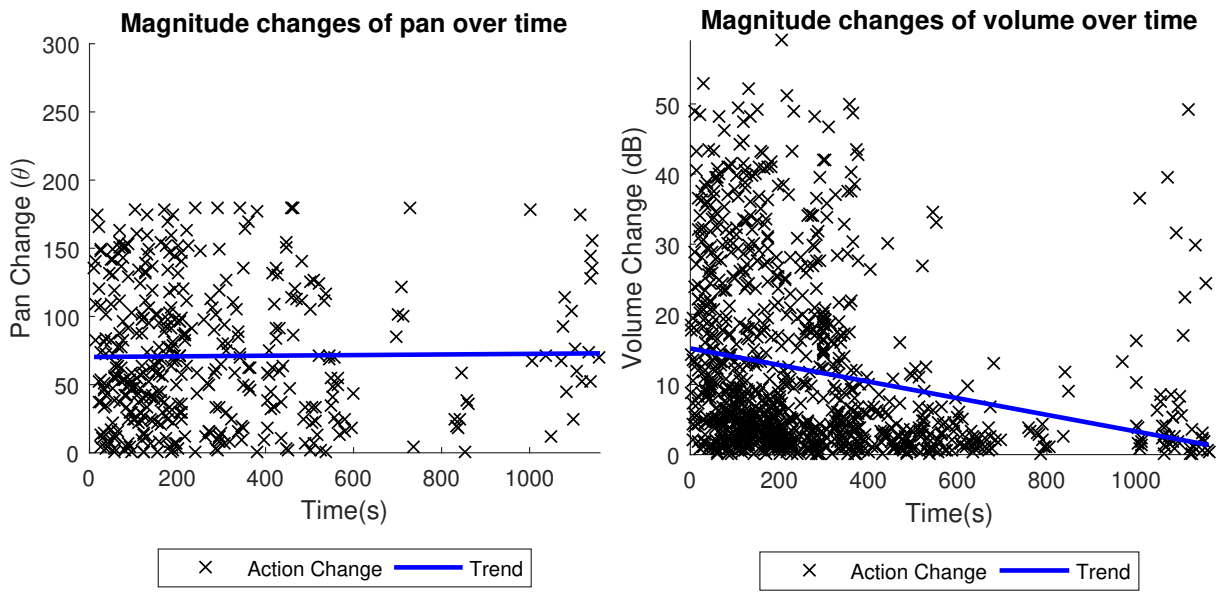


(a) Magnitude of pan interactions over time



(b) Magnitude of volume interactions over time

Figure 4.13: Magnitude of panning and volume against session time since first user action. Blue line shows the trendline over time.



(a) Magnitude of pan interactions over time, filtering pannings over 180deg

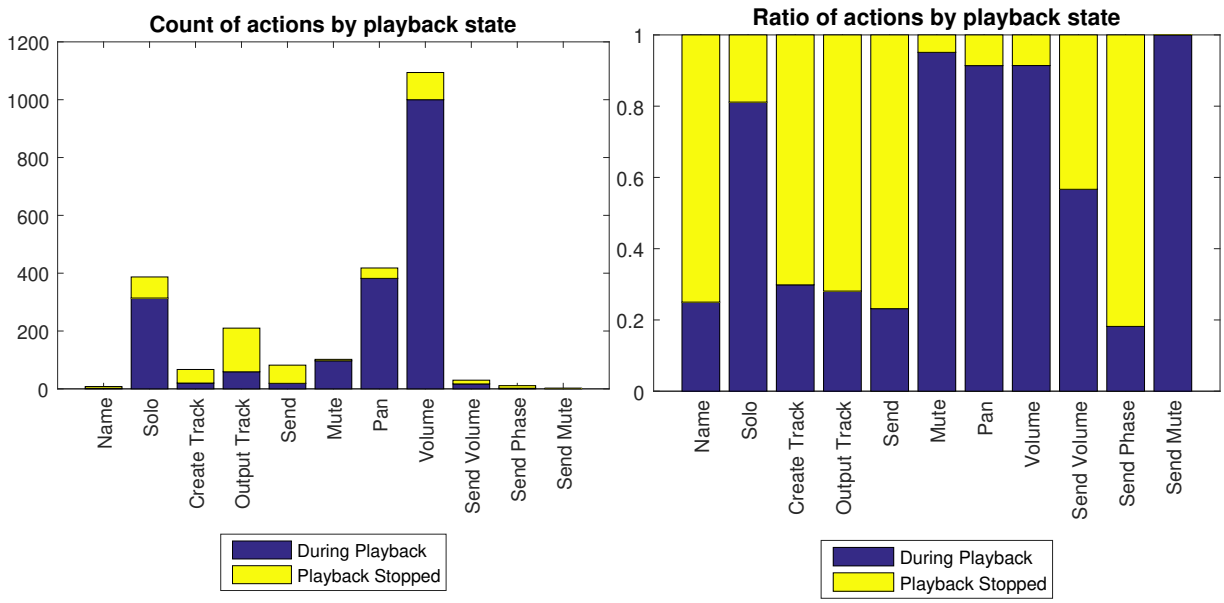
(b) Magnitude of volume interactions over time, filtering for dB changes over 100dB

Figure 4.14: Magnitude of panning and volume against session time since first user action, filtered for actions less than 100dB, where volume could be used as mute, and panning of absolute changes which is stereo flipped but spectrally similar.

changes of near 200dB would indicate situations where the track is being muted and un-muted using the volume control instead of the 'Mute' control.

Figure 4.14 shows the result of filtering the mute actions out by setting a threshold at 100dB. This demonstrates that the volume changes are negatively correlated with time. Panning does not follow the same pattern, where consistent parameter changes happen over time. This would indicate that panning is less stable than loudness, and that engineers would flip the left and right channel bias as the mix goes on. In this case a change from 60L to 60R or vice-versa would both mean a magnitude change of 120, even though perceptually they are equally off-centred. By adjusting for this magnitude distance from centre score it is clear that panning magnitudes do not decrease over time like volume controls do.

The actions performed also depend on the current state of the system. The DAW can be in two states at any given time, during playback or not playing anything. Most modern DAWs can have multiple views too, but for this purpose only the playback state was captured. Figure 4.15 shows the actions captured, grouped by whether the DAW was currently playing or not. What is immediately apparent, is certain actions are far more likely to occur depending on the state the DAW is in. Of the 2411 actions which are not transport related, 1914 took place whilst the DAW was rendering audio, meaning 79.39% of the total actions occurred during playback. The most prominent actions to occur when listening to audio is 'Solo' (81.14%), 'Volume' (91.41%), 'Pan' (91.39%) and 'Mute' (95.10%). These all have direct impacts on the current playback of the system. 'Solo' and 'Mute' are used to isolate and remove certain elements respectively, allowing the engineer to focus on a given section at any given time. Whilst 'Volume' and 'Pan' could only be evaluated during the



(a) (b)

Figure 4.15: Count of actions grouped by whether the DAW was playing

playback of audio. Conversely, the non-playback actions 'Create Track' (70.15%), 'Change Output Track' (71.90%), 'Change Name' (75.00%) and 'Create Send' (76.83%) are more structural. These will create the mixing structures the engineer needs, and require multiple user interactions to complete. Therefore these are more commonly done when the DAW is not actively playing.

The listening test provided a set of subjective rankings for the 35 mixes created. Section 4.2.1 gives an overview of the processing performed and the results obtained. Since the aim of the balance mix is to ultimately convey a rough approximation of the final mix it is appropriate that the mixes should be comparable to each other at this stage (Izhaki, 2012). Each mix is given two data points, the number of actions performed and the mean ranking score, which is detailed in section 4.2.1 and gives an overview of the processing performed. The action counts are dependent on the number of tracks in the session, as shown in Figure 4.16, with a correlation of $R = 0.4413$ and a P-value score of 0.0080.

The action counts performed by each session should be related to the ranking score of the listening test for that mix. This hypothesis is because a mix with no edits would effectively be the same as the anchor, whilst a song with many edits would be different and therefore score a different result. By its very nature each engineer aims to improve the mix so it should be a positive correlation. To remove the bias that more a session with more tracks requires more actions to complete the process, the actions counts are divided by the number of tracks to give the number of actions per track in session. This is to ensure that larger sessions are not penalised because of their increased control space. The listening tests are normalised by taking their ranking position and dividing it by the number of elements to give a rating between 0 and 1. With the action counts and listening test normalised, the comparison between the two can be evaluated and is shown in Figure 4.17. When comparing the action counts performed by each session against the mean ranking score of the listening tests, a slight

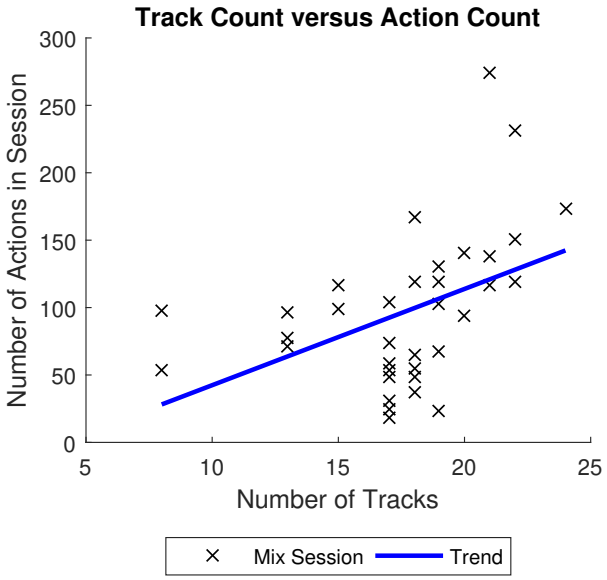


Figure 4.16: Comparison between the number of tracks in the session versus the total number of actions performed in the session.

correlation is found between the two of $R = 0.2598$ using the Pearson correlation. This shows a slight positive correlation between the number of actions an engineer performed and the rating of that session.

Further to this, the type of actions being performed also has an impact on the quality of the mix. The audition count shows how many times the engineer played the mix during the study. As presented earlier certain actions are generally only done when the DAW is currently playing, such as volume and panning. Therefore, the amount of times the engineer played the session should have an effect on the quality. Figure 4.18 gives the comparison between the number of auditions and the ranking given to each mix. There is a positive Pearson correlation of $R = 0.1972$ and p-valoe of 0.2561, which shows a very weak correlation, although not significant This shows that the number of auditions is not related to the performance of the mixing engineer. When compared to the minutes spent auditioning the relationship changes. Since the start and stop events of the DAW are logged, it is possible to collect the amount of time the session was in the played state and stop state. Figure 4.19 shows there is a positive correlation between the two metrics, showing that engineers who spend more time listening to the mix would tend to have better mixes. The Pearson correlation of this data was $R = 0.3878$ with a p-value of 0.0213.

This strong positive correlation though could be down to a few other influences. The first is that engineers spending more time auditioning their sessions are also probably spending more time in those sessions themselves. In fact, the amount of time an engineer spends in the session against the amount of time spent auditioning had a correlation of $R = 0.8460$, showing an incredibly strong relationship. Therefore the duration itself might be the correlated variable.

Figure 4.20 presents a comparison between the number of minutes it took to complete the mix and the ranking given to that mix. This was given a Pearson correlation of $R = 0.2629$, which shows there is a slight but not conclusive correlation between them, with a p-value of 0.1605. This is mostly due to the fact that very short

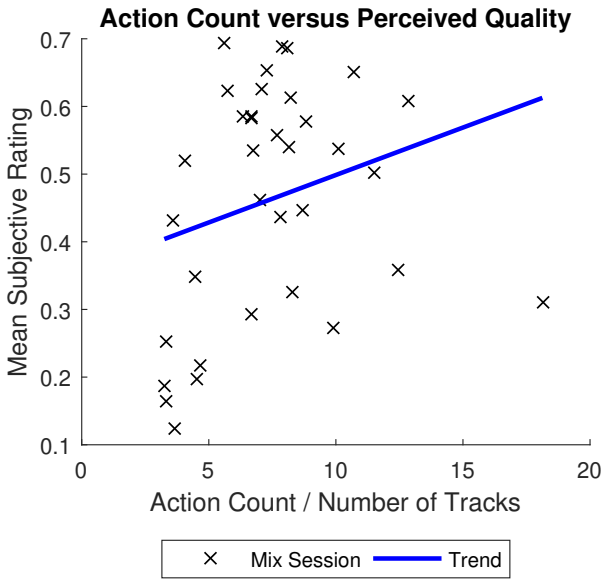


Figure 4.17: Comparison between the number of actions performed and the ranking of each mix.

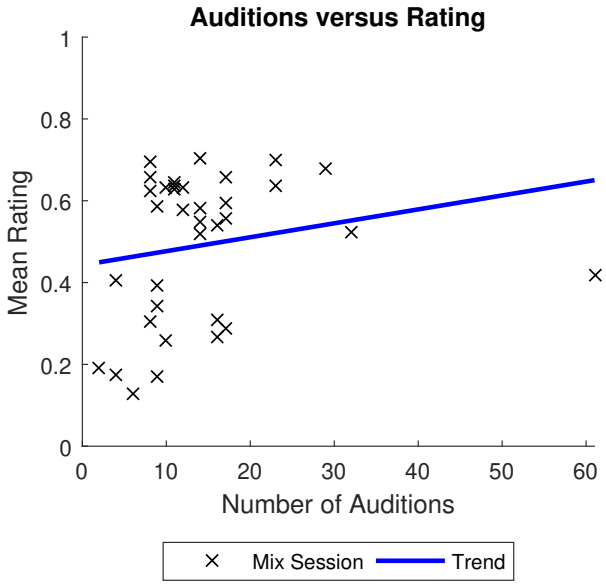


Figure 4.18: Comparison between the number of auditions (playbacks) performed and the ranking of each mix.

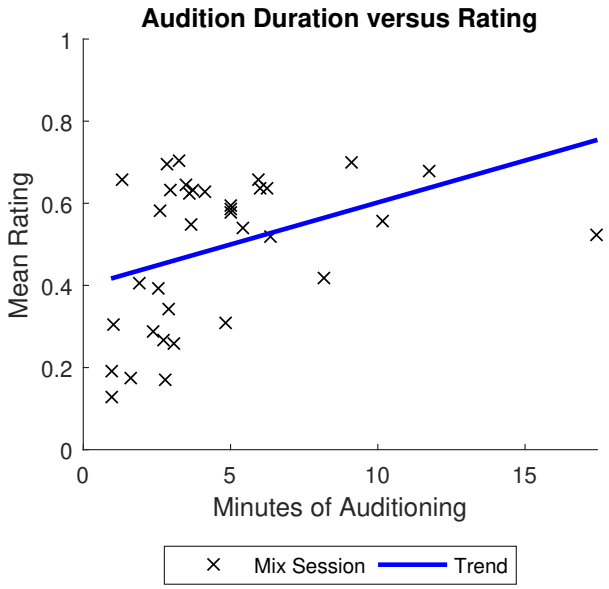


Figure 4.19: Comparison between the number of minutes spent auditioning each mix and the ranking of each mix.

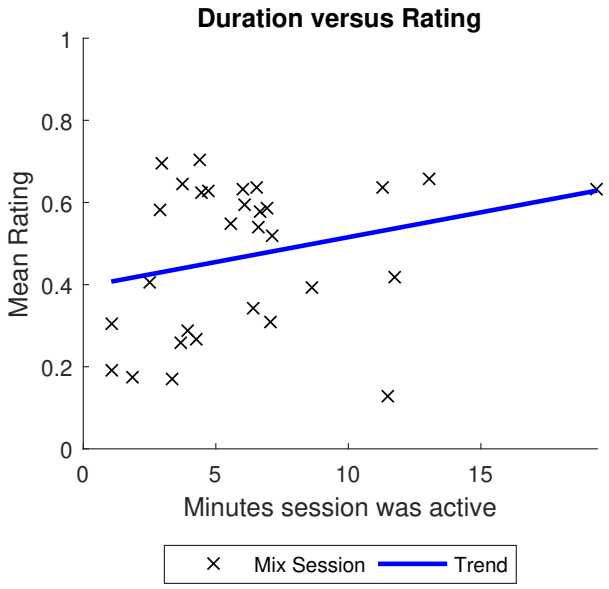


Figure 4.20: Comparison between the number of minutes spent creating each mix and the ranking of each mix.

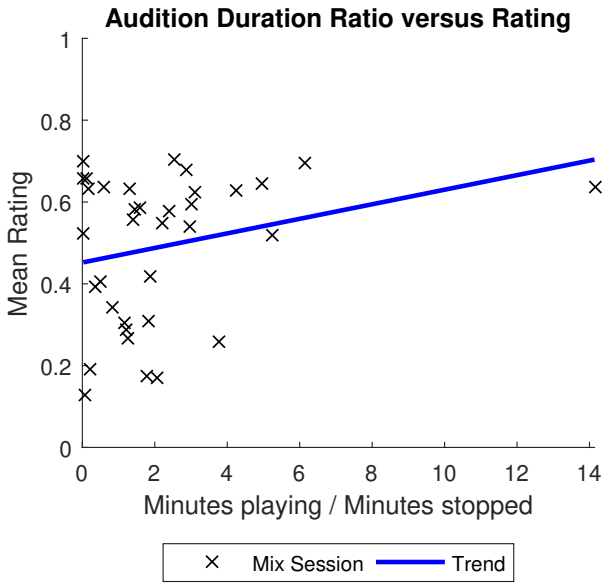


Figure 4.21: Comparison between the ratio of the minutes spent auditioning to minutes spent silent, and the ranking of each mix.

durations tended to perform worse, whilst those with a 4 to 8 minute range seemed to perform better. There was no evidence that longer durations led to improved mixes. Comparing the ratio between the minutes playing and minutes not playing could show a better metric for engineers, where a higher ratio of time spent playing could indicate a better mix. Figure 4.21 gives this figure and again the Pearson correlation shows there is a very slight correlation between the two with a Pearson correlation of $R = 0.0.2586$ and a p-value of 0.1337. This is possible because the relationship itself may not be linear, that there is a sharp increase of the performance of the mix but over time gains are limited.

4.3.2 How are groups and sends used in the session?

The second question investigates how engineers configure their session routing, if at all, during the balance mixing process. The two types of structural tools available to mix engineers are sends and groups. *Groups* are a subset of tracks from the mix which are routed to a bus before the mixer, effectively inserting another channel strip between itself and the master output. An engineer does this to help organise the session and to increase the capacity of their control surface (Izhaki, 2012, p. 129). *Sends* also route audio information to the bus, but this is done in parallel, so the output of the track is unaffected (Huber and Runstein, 2005, p. 409). Unlike groups, which route the audio through another track for summation and processing, a send creates a copy of the audio to pass to a different track. This means sends are used primarily for parallel effects processing, such as reverberation or other time based effects (Huber and Runstein, 2005, p. 409). A send can be taken from before the effects (Pre-Effects), before the fader (Pre-Fader or Post-Effects) or after the fader and panning (Post-Fader) (Huber and Runstein, 2005, p. 409). Previous studies showed that there is a strong correlation between session organisation (sends, busses and grouping usage) and perceived musical quality (De Man and Reiss, 2017; Ronan et al., 2015b), although the mixing task was for a completed mix. Engineers

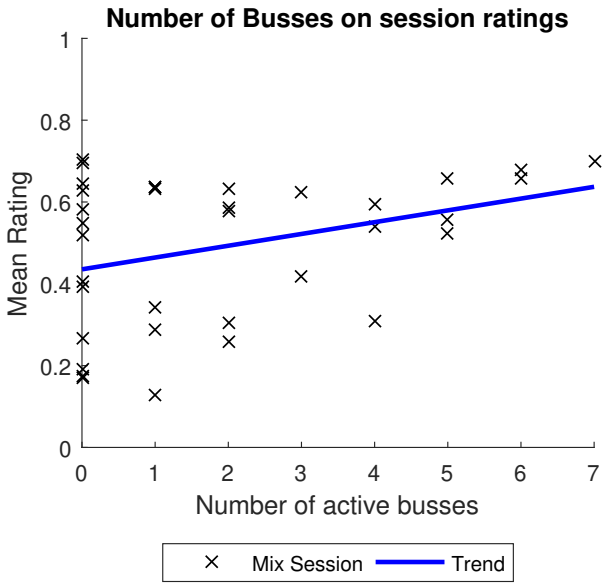


Figure 4.22: Comparison between the number of busses created and the ranking of each mix.

who actively grouped their mixes into separate sub-mixes performed better as they gain more control over the mix. Neither study showed where in the mixing process the engineers would start to use these features.

There is a correlation between the number of active busses in the session and the overall ranking of the mix given. Figure 4.22 shows this linear correlation, which has a Pearson correlation of $R = 0.3433$. This confirms that sessions which use busses tend to have higher scores. As the figure shows, sessions that have no busses can achieve very high subjective rankings. However the spread of these was very large, with the mean ranking being equal to 0.4552 and a standard deviation of 0.2018. By just having two busses in the session, this score is increased to 0.4709, with a narrow standard deviation of 0.1755. Generally, each time a bus is added to a session the mean ranking increases and the probability that it will be a higher scoring mix increases.

Configuring session structures (sends and groups) comprised 10.050% of the actions. This was surprisingly low given the results in prior research suggesting that the more groups a session has, the better the perceived final mix (Ronan et al., 2015b). Out of the 35 filtered sessions, 22 of them made at least one bus track. This track behaves like a normal audio track, except its input can only come from another track or a send. This means it is only used for structural purposes in the sessions. In these 22 sessions, a total of 68 busses were made. This shows that a significant amount of routing is constructed at this early phase. In total across the 35 filtered sessions there were 514 starting tracks. Of these 514 starting tracks, 221 were sent to a bus instead of the master.

Since a bus could be used for both grouping and as a send destination, it is important to filter these two out. A bus is considered to be a group if it has at least one other track being routed to it. This definition allows for groups of size one and groups which hold other bus inputs as well. A send bus is a bus which has at least one send being received on that bus. These definitions allow for overlap, where a bus can be both a send bus and a summing input bus. By using these definitions, of the 68 bus tracks that were created, 57 are group busses

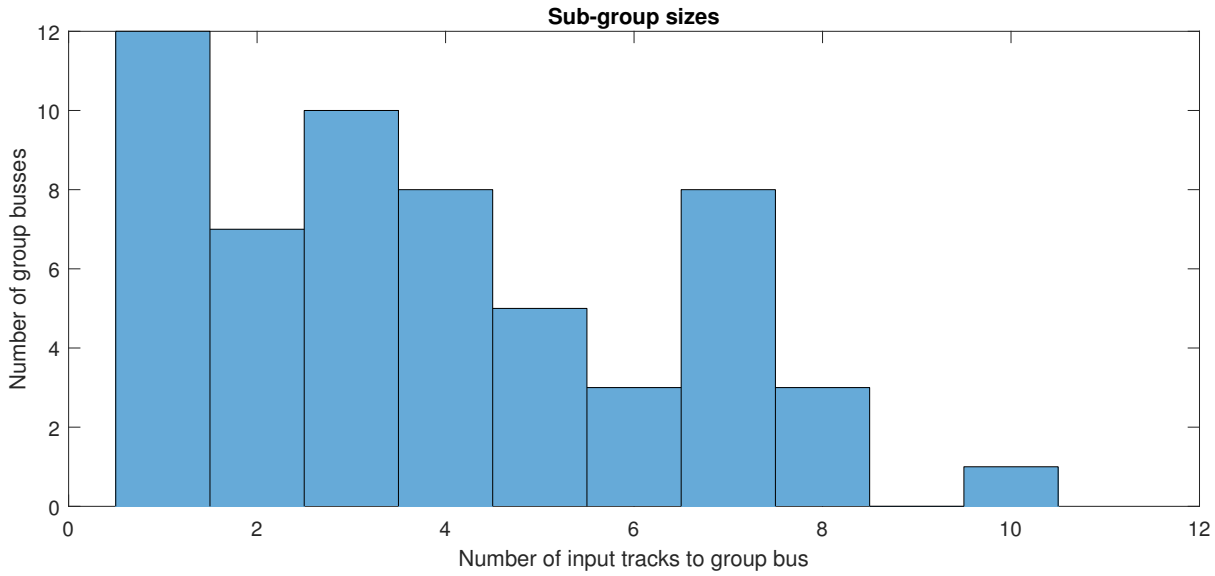


Figure 4.23: The number of group busses across all sessions against the number of input tracks to those group busses.

and 17 are send targets. This means 6 bus tracks are a combination of the two. Of the 22 sessions which used busses, 20 had a group structure and 13 had a send structure showing sessions often have both included.

Figure 4.23 shows the size of each of the 57 groups that were made over the session. Most interestingly, the most common size was a single bus, which doesn't match with the theory that busses are used to segment and control the mixing space. Larger groups are not common because of the size of the sessions involved. Figure 4.24 confirms this statement as well. The mean group size is 26.13% of the total incoming audio tracks in the session. The largest group only took 62.5% (10 out of 16) of the total incoming audio tracks. These confirm the act of segmenting the session into more manageable chunks.

Another important measurement is the number of tracks that are fed into a bus relative to the session size. This can be viewed as how many tracks are sent to a bus instead of to the master as is the default behaviour. This coverage ratio would give a percentage of the tracks in a group bus structure. Figure 4.25 shows this as a histogram. Of those 20 sessions which had a group structure included, 11 of them had a coverage ratio above 88%, with a mean coverage ratio of 74.47%. This shows that when busses are used, they should cover the majority of the session. Only 6 sessions had a coverage ratio under 50%, and even then the majority of these were above 40%. This shows that when busses are used as groups, they should aim to simplify the session appropriately.

Further examination of the groups shows that the overwhelming majority of the groups created were for drums. Table 4.22 gives the details of the 57 groups created in the session. The drum names are given, along with any other alternative names which are for the same group meaning. For example the 'Drums' includes groups named 'Drum Bus' and 'Kit' as a reference to 'Drum Kit'. 19 of the 57 groups were for a drum kit, indicating this is the most important instrument to be grouped. This is intuitive since the drum kits were all multi-mic sources in the mixes, meaning the drum kit had multiple microphones recording it. Therefore placing this in a

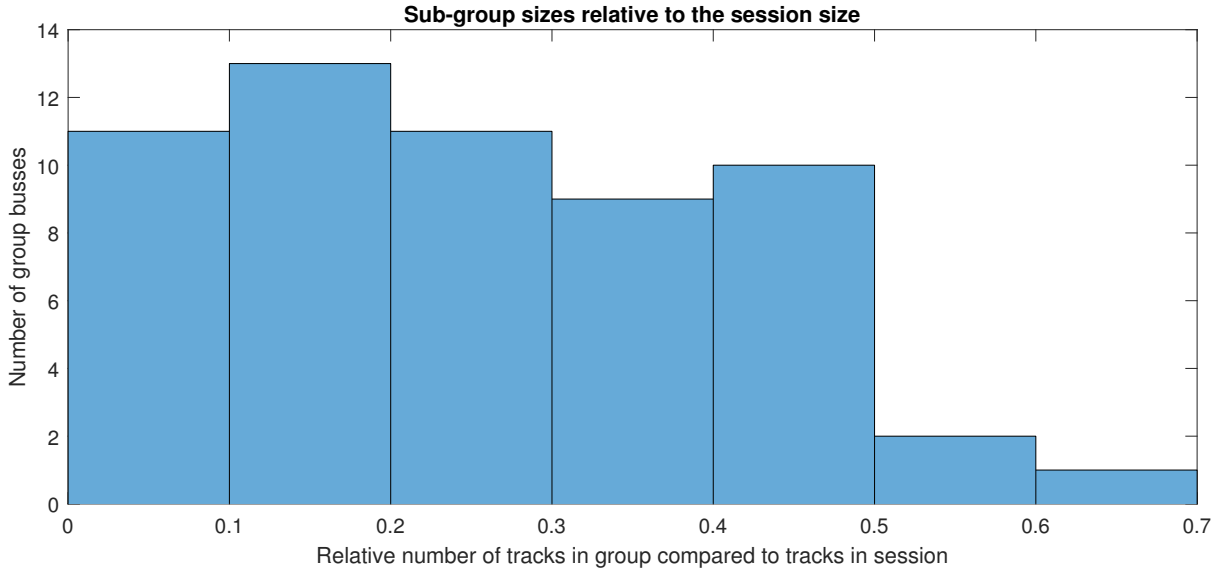


Figure 4.24: The number of group busses across all sessions against the number of input tracks to those group busses relative to the number of tracks in the session.

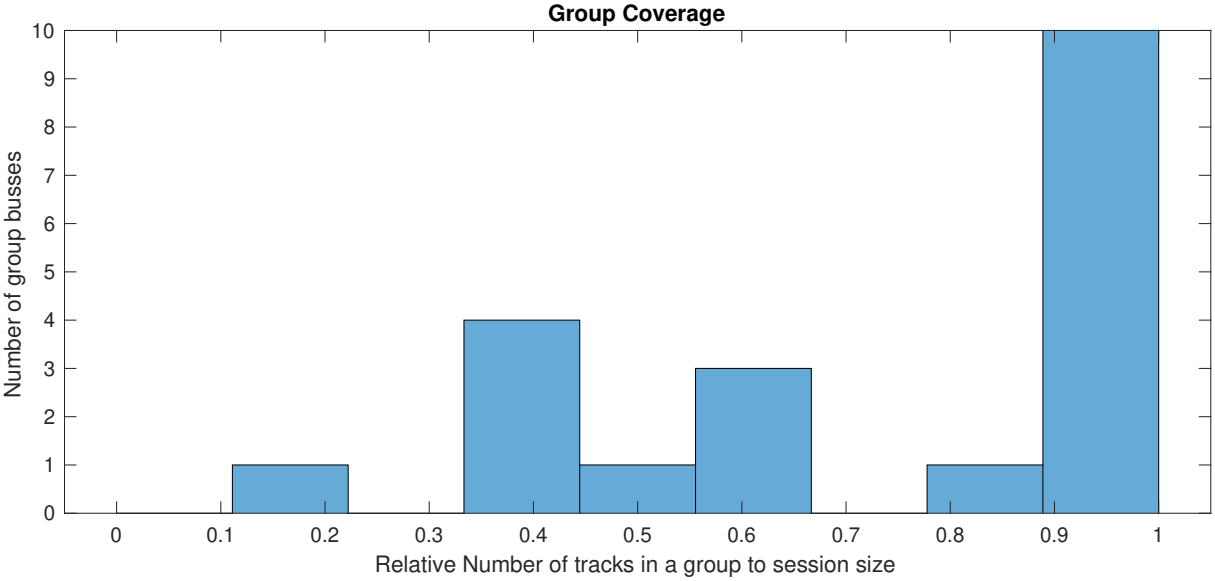


Figure 4.25: The coverage of all tracks in a session which are being sent to a group. A higher number indicates more of the audio tracks are in a bus.

Group Name	Number of Groups	Number Of Sources
Drums (Drum Bus, Kit)	19	121
Guitars (Guitar, Gtr)	8	24
Percussion (Perc)	7	19
Vocals (Vox)	7	21
Bass	6	7
Piano (Keys)	5	13
BkVox	5	13
LdVox	5	12
Synths	4	20
Master Right	1	1
Master Left	1	1

Table 4.22: The names of the groups created across the 35 sessions, along with the number and the number of tracks sent.

Send Name	Number of Sends	Number Of Sources
Verb	5	19
Big2	2	9
PhaseR	2	5
Drum Bus	1	7
Master Right	1	1
Master Left	1	1

Table 4.23: The names of the sends created across the 35 sessions, along with the number and the number of tracks sent.

group would allow the engineer to approach this as a singular instrument to be processed and controlled, just like the other instruments.

Guitars were also very commonly configured into groups. All the names of guitar groups tend to reference an instrument, not an effect. This shows clearly that grouping is done by instrument types. The master right and master left entries were found in one mix, and received sub-mixes from other groups. These were both hard panned left and right as per their name, indicating these were for master based effects.

Sends are more often used in live or recording environments, where send mix can be created to give an artist specific parts of the mix for them to play along to. For example a guitarist may just want to hear the drums and their own guitar, not the other instruments going at once. Using sends allows the engineer to do this by copying those aspects of the mix and sending it to the artist. In mixing terms, it is almost exclusively used as a method for effects which should run in parallel. Common methods include the parallel compression on drum kits and reverberation where having the direct sound and reverberated sound on individual controls gives the engineer more creative freedom.

In this experiment the sends accounted for a very small amount of the total number of the actions, just 3.69%. This was not unexpected as no plugins were available for the engineers. Therefore there is no benefit for setting these up as it will copy the mix, effectively adding gain which can be done through the mixing console. It is interesting that several mixing pipelines were created and used at this point, indicating that some engineers do want to have these structures created ahead of them using it.

The mixing structures also include the methods that the engineers use to isolate certain aspects of the session when performing the mix. Each track in the DAW is fitted with two isolation methods: ‘Solo’ and ‘Mute’. The ‘Mute’ action will mute the output of the track, removing it from the mix. The engineer may decide the track is no longer needed and believes it should be removed. Or the engineer can use it to temporarily silence a part of the song that they do not wish to have interfere as they focus on another section. The ‘Solo’ is a feature, which when active mutes all other tracks except those with the solo active. It therefore completely isolates the track quickly from the rest of the mix without needing multiple mute functions. The use of both of these was extensive in the balance mix task. Nearly 14% of the total actions comprised interactions with these two functions.

Table 4.23 gives the names of each send bus created with an active send in place. This shows a big difference in what sends are used for, compared to the group busses in Table 4.22. Here the send names are all to do with size and space, with the most common name ‘Verb’ being a common alias for ‘Reverberation’. Likewise ‘PhaseR’ was actually used for some spatial effects in the DAW itself. Whilst no effects would normally be allowed, the sends busses in the DAW did have phase control, allowing the signal to be inverted. By copying the signal using the sends, phase flipping it, and panning it hard to the left or right channel, a phasing effect is created which can be disconcerting to the listener but is an effect. But as shown before, these were used far less than the groups at this stage due to the fact they are for effects based processing primarily. And with no effects to employ there was no mixing benefit to using them.

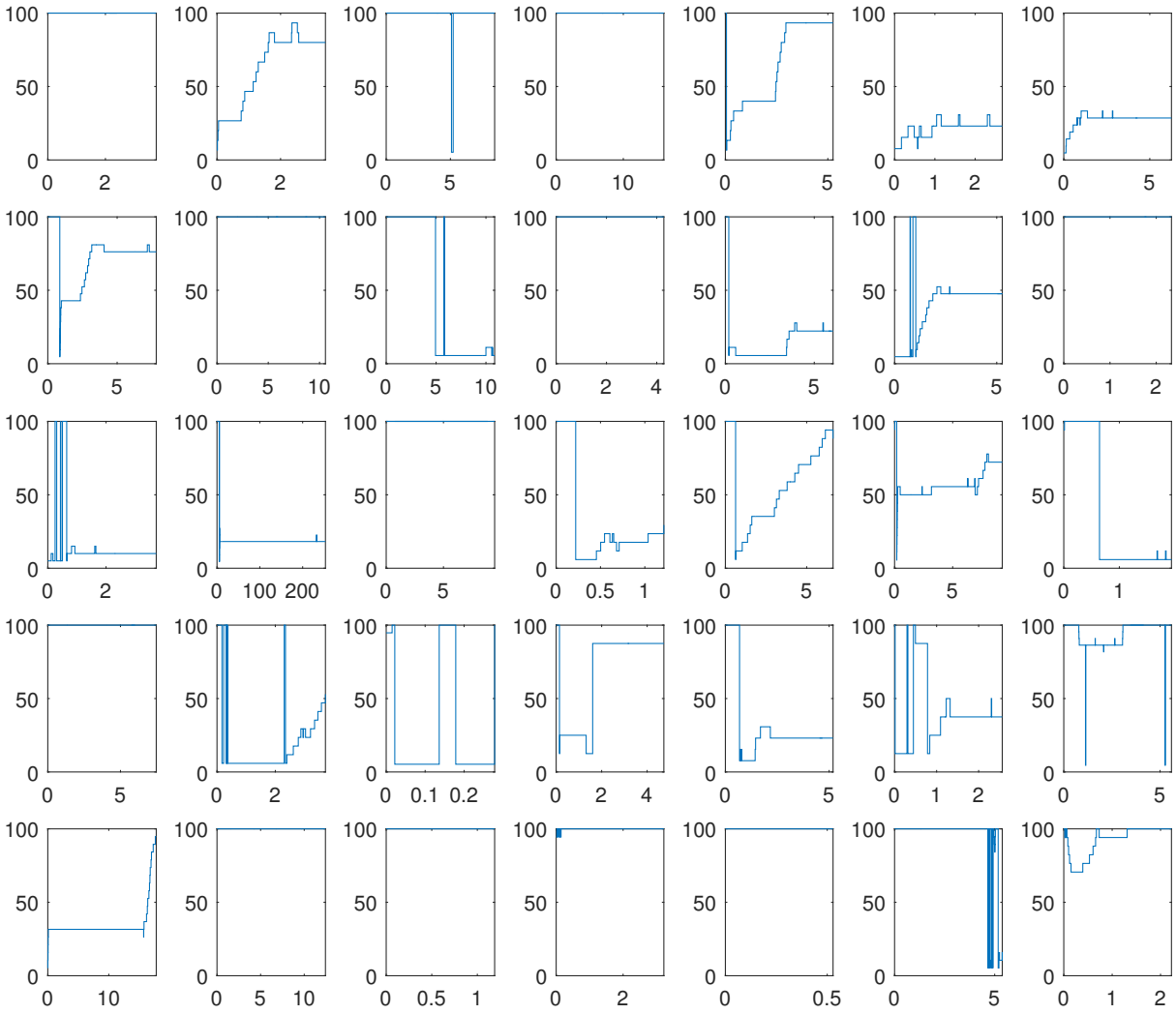


Figure 4.26: Timeline of the percentage of active tracks for all 35 filtered sessions. The bottom axis is time in minutes, with the left axis being tracks active as a percentage.

Figure 4.26 shows the active track counts, as a percentage of total tracks in the session, for all of the filtered captured sessions. It is immediately clear there are two distinct methodologies engineers take. One group of engineers will mute everything other than a small number of tracks, then re-introduce tracks over time until they are all present in the mix. This is called the serial approach, where each track or section is mixed individually before adding in another group (Izhaki, 2012, pp. 37-38). This approach does speed up the workflow, but tends to leave the engineer without much headroom in the mix. Another approach is to start with the rhythm section, then the harmony and finally melody. This approach is also used by some of the engineers to, but usually those with groups. The final approach is to operate by order of importance, such as mixing the vocals first and then working their way down the mix.

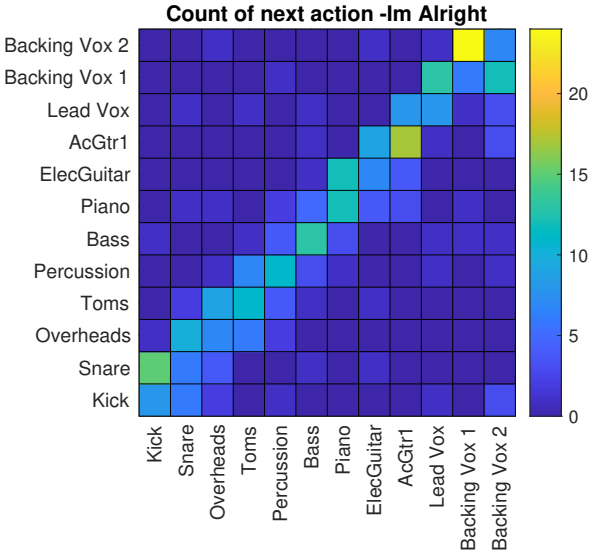
Some tracks are still muted at the end, possibly because the engineer feels the track should not be in the final mix, or they have not completed the mix process when the exited the test. Another group will use the solo button to selectively isolate tracks before returning to the global mix. A third group never isolates any tracks at any stage, called the parallel approach (Izhaki, 2012, pp. 37-38). This has the advantage that no track is ever in isolation, meaning every mixing decision is built upon the previous iteration.

4.3.3 How many user interactions occur during a balance mix?

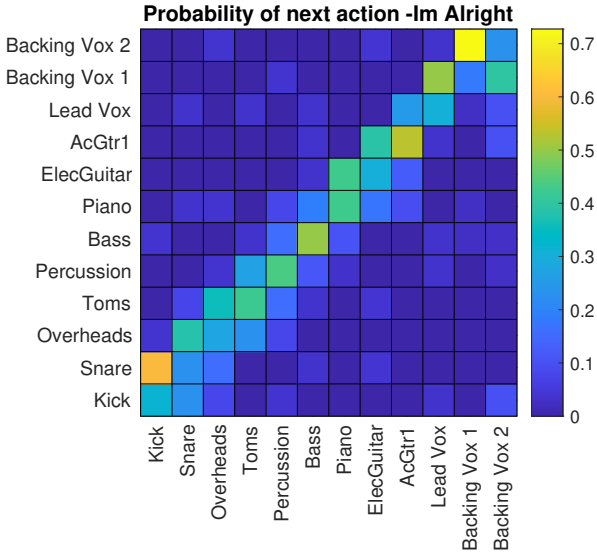
During the mixing process there were 174.72 user actions on average per session. The majority of these were for transport control and volume changes (63.78%). The full breakdown is given in Table 4.20 for all 35 sessions together. The number of actions experienced in a session is strongly correlated with the total number of tracks in the session, including user created tracks such as groups and busses. Using the Pearson correlation score, the relationship between the number of tracks and the number of actions is $R = 0.433$, showing that as the number of tracks grows the number of actions grows too. This is depicted in Figure 4.16. Figures 4.27 to 4.31 show the probability of the next action to take, based on which track the preceding action occurred. The action lists were taken from the filtered sessions and show all track-based actions. Therefore it ignores any session-based actions, such as playing or stopping, group and bus creations, and send interactions. It also ignores the first action, since this does not have a previous action.

The results show that when an action is applied to a selected track, there is a higher probability the next action will take place on one of its neighbouring tracks in the mixer view. This is shown through the magnitude of the diagonal cells in the matrix often having two neighbouring regions of activity. This indicates engineers operate on a left-to-right basis, therefore the ordering of tracks in the session should have a significant impact on the outcome of the mix.

On the smallest sized session ‘Sleigh Ride’, which has 7 tracks, there is still a clear preference to work on the neighbouring tracks. It it also fairly clear that similar, or overlapping instruments, are closely controlled. This session has two microphones on the drum kit (‘Kick’ and ‘Overheads’), two for the Double Bass (‘Bass DI’ and ‘Bass Amp’), and two Piano microphones. Therefore, these form natural groups as they must be mixed together. With the ‘Kick’ track, the next action occurs on the same track 50% of the time, or on ‘Overheads’ 40% of the time. Conversely with ‘Overheads’, 50% of actions occur on ‘Overheads’ and 25% on ‘Kick’. Similar levels of control are visible on the Piano tracks.

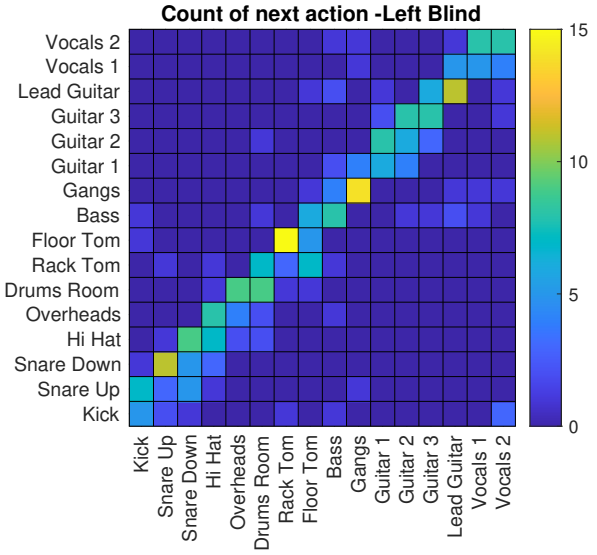


(a) Count of actions occuring on the next track for 'I'm Alright'

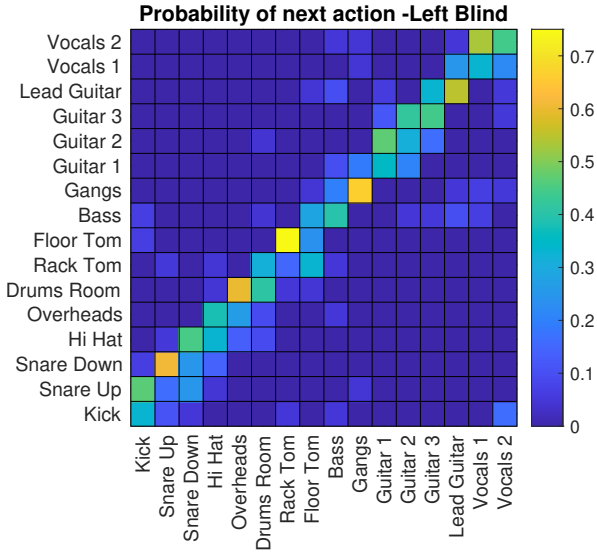


(b) Probability of action occuring on next track for 'I'm Alright'

Figure 4.27: Matrix showing where the next action occured as a count and probability for 'I'm Alright'.

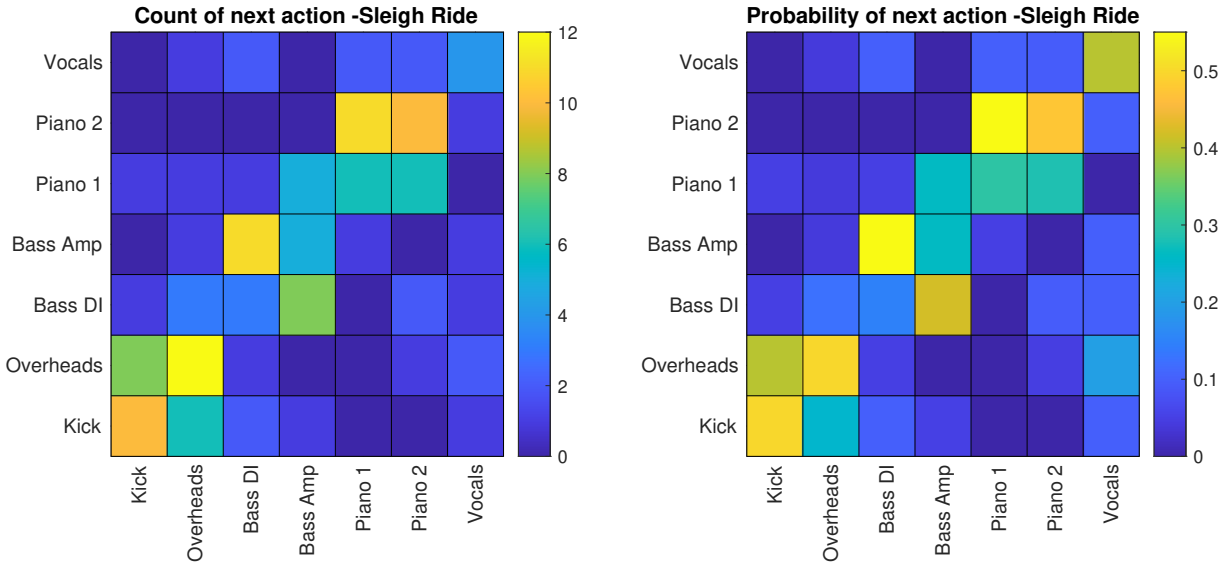


(a) Count of actions occuring on the next track for 'Left Blind'



(b) Probability of action occuring on next track for 'Left Blind'

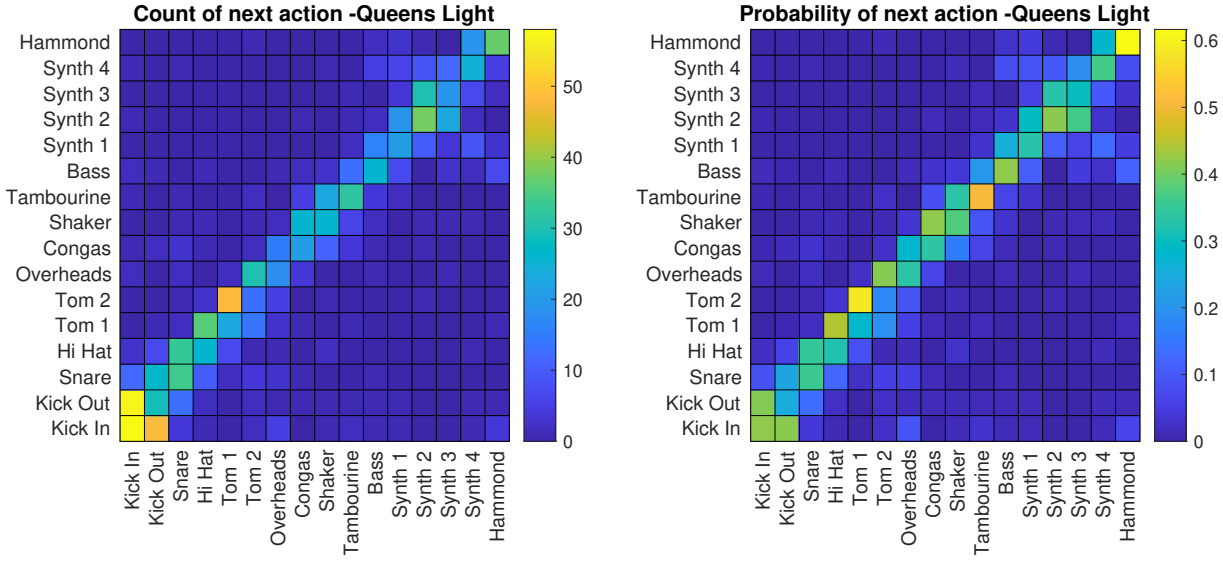
Figure 4.28: Matrix showing where the next action occured as a count and probability for 'Left Blind'.



(a) Count of actions occuring on the next track for 'Sleigh Ride‘

(b) Probability of action occuring on next track for 'Sleigh Ride‘

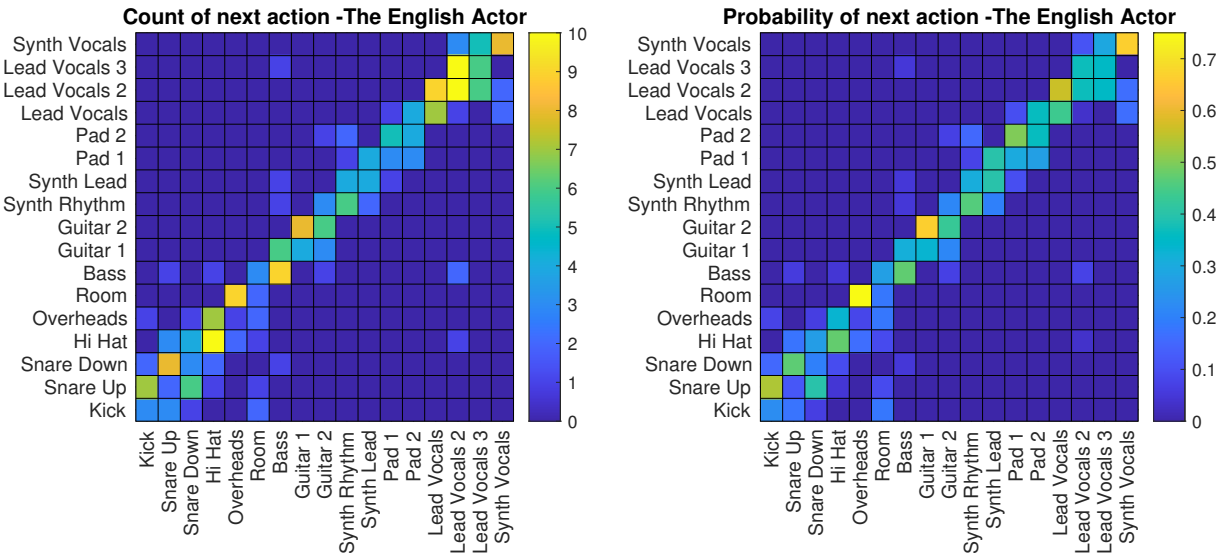
Figure 4.29: Matrix showing where the next action occured as a count and probability for ‘Sleigh Ride’.



(a) Count of actions occuring on the next track for 'Queens Light‘

(b) Probability of action occuring on next track for 'Queens Light‘

Figure 4.30: Matrix showing where the next action occured as a count and probability for ‘Queens Light’.



(a) Count of actions occuring on the next track for 'The English Actor'

(b) Probability of action occuring on next track for 'The English Actor'

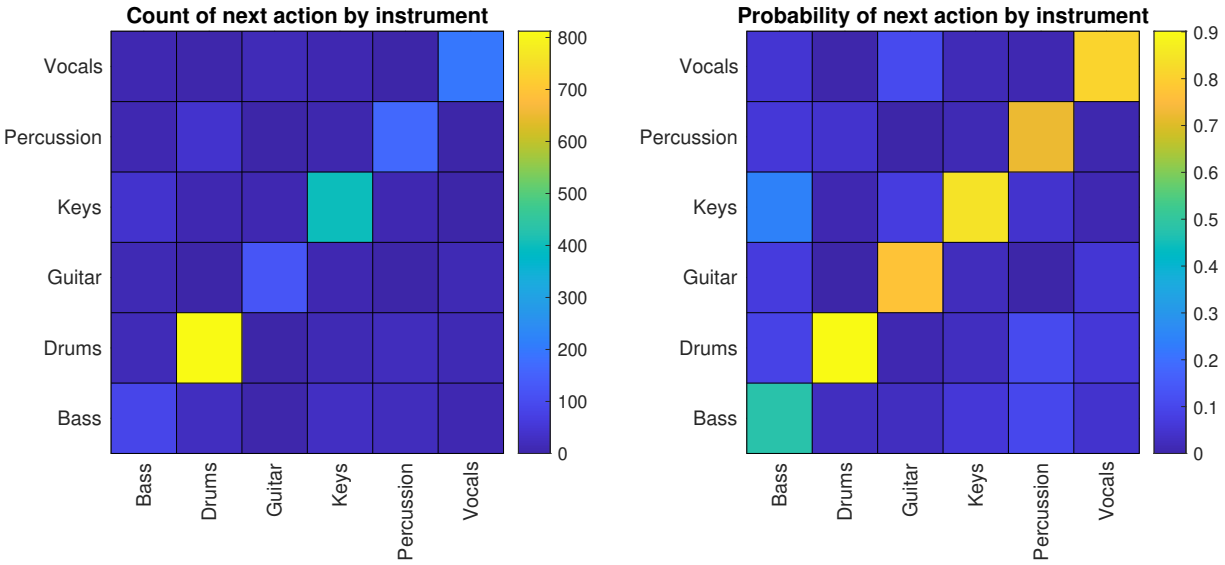
Figure 4.31: Matrix showing where the next action occured as a count and probability for 'The English Actor'.

This level of grouping is present on the larger sessions as well. In Figure 4.30 there is a cluster forming around the 'Synth 1, 2, 3, 4' and 'Hammond' tracks. With larger sessions, it is more likely that not all of the user interface was in view at the same time. This might make it more difficult for the engineer to control, or go back to previous sections once they are deemed to be complete.

Figure 4.32 shows the same count and probability information, but for all of the sessions combined. Each track is grouped into an instrument tag, which shows the movement between instrument types rather than just track. This strongly suggests engineers will work on a group of tracks initially to determine the mix of that particular set of tracks. There is a self-bias in here, given certain actions are naturally grouped together, such as panning and volume controls. These would be listed as two actions on the same track, whilst it may be just one mixing decision. This would increase the likelihood that the same action occurs on the same track, but it shows how multiple actions are required to perform a single mix decision. Figure 4.33 further supports this point. This graph shows the same count and probability organised by the action types.

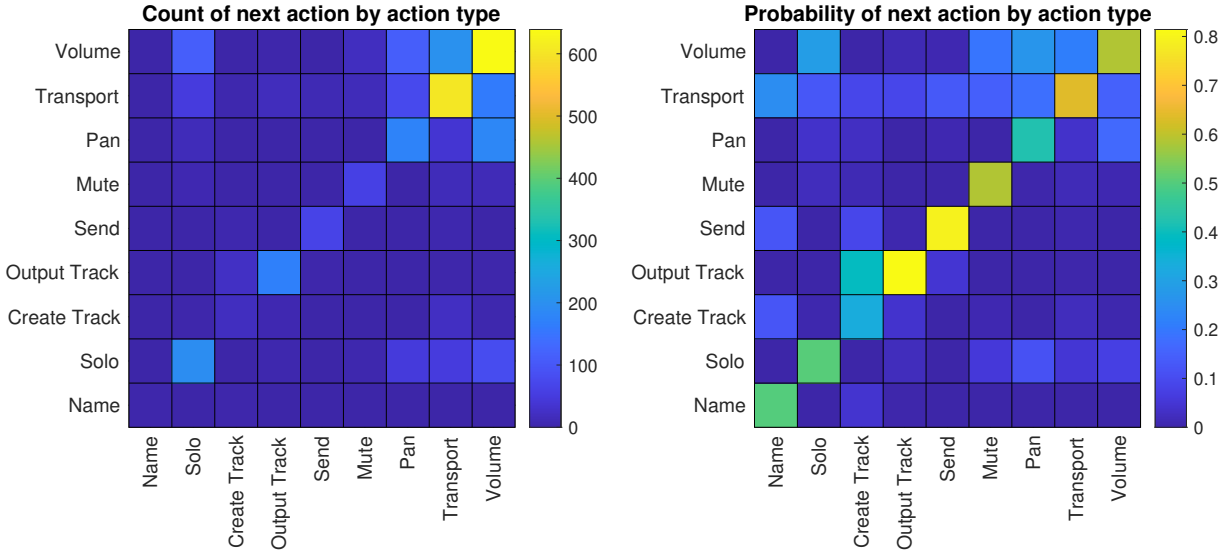
Volume and pan both tend to have quite a strong connection to each other. On executing a volume change, 16.87% of the next actions were pan and 58.57% of the next actions were volume. Conversely, on executing a pan change, 27.03% of the next actions were volume and 42.34% of actions were pans. The strong connection between the two shows a significant amount of energy is spent interacting between these two controls. Volume generally has a large probability of being the next action. This is not unexpected since the interaction of two tracks together in volume is very critical for mixing.

Transport also has a large chance of being the next action. Again, this makes sense since transport includes start, stop. These three actions are combined into one group, so when the engineer stops they often start it



(a) Count of next action occuring by instrument type (b) Probability of next action occuring by instrument type

Figure 4.32: Matrix showing where the next action occured as a count and probability adjusted by instrument type.



(a) Count of next action occuring by action type (b) Probability of next action occuring by action type

Figure 4.33: Matrix showing where the next action occured as a count and probability adjusted by action type.

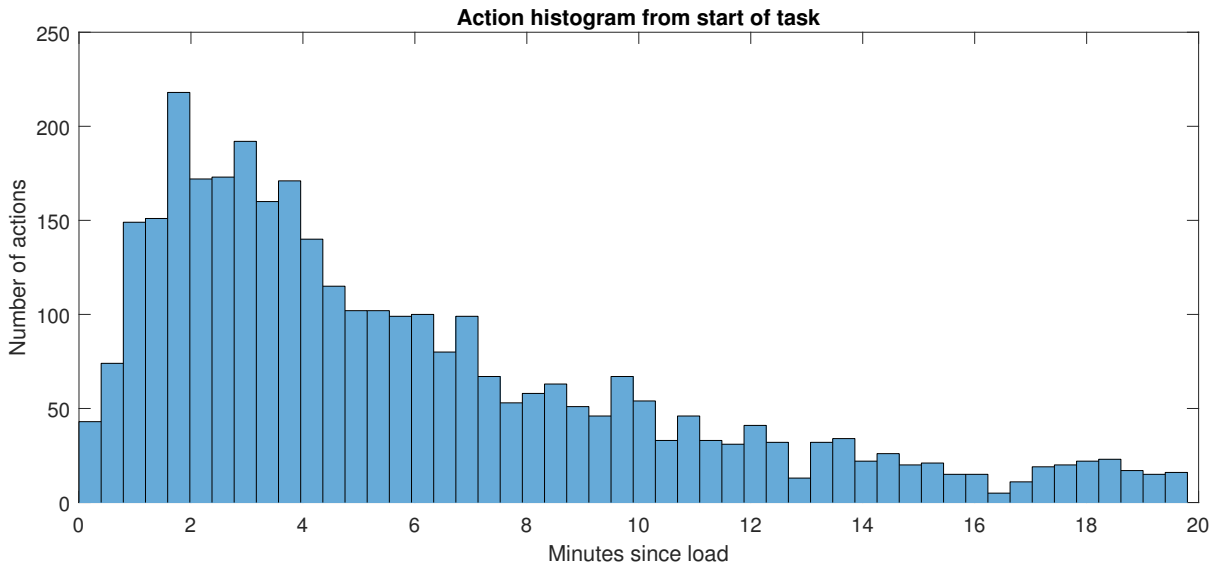


Figure 4.34: Histogram of the time an action occurred since the start of the session. There is a larger amount of activity being undertaken at the start of the sessions.

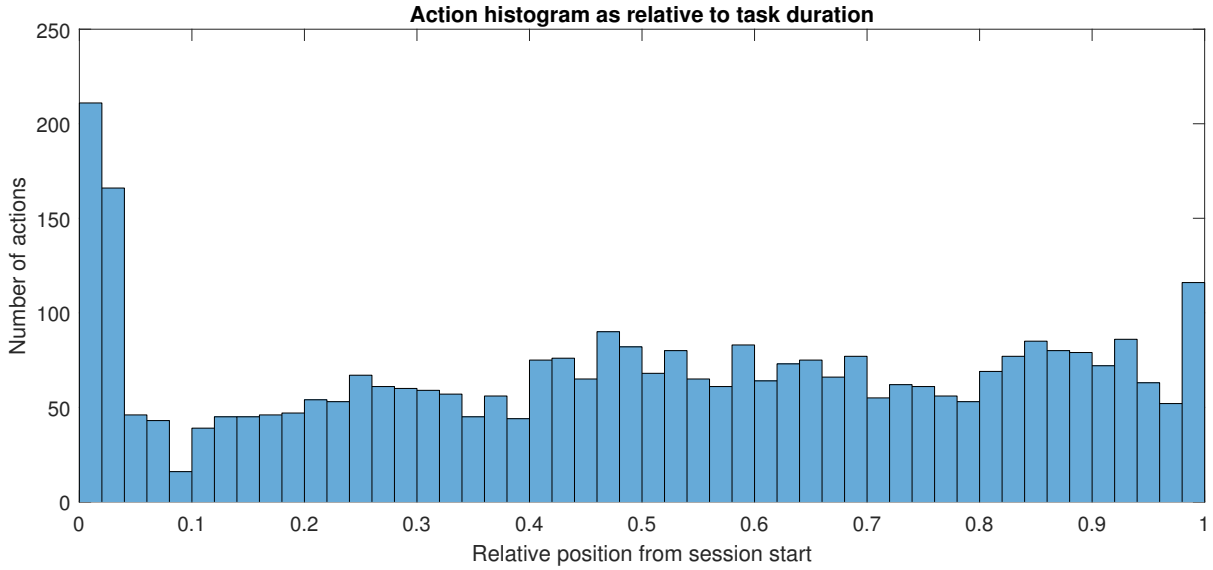


Figure 4.35: Histogram of the time an action occurred since the start of the session relative to the duration. There is a larger amount of activity being undertaken at the start of the sessions.

again. The data collection platform also did not have a looping function, so the engineer would have to stop the track when it reached the end to start it again.

When actions occur is also of interest, since the process is ever evolving as the engineer makes adjustments to reach their solution. As already shown in Figure 4.13 the magnitude of actions decreases as the time of the session increases. And figure 4.15 shows that certain actions are far more likely to occur during session playback than when the session is stopped. Figure 4.34 shows the distribution of session actions occurring during time bins since the start of the session. There is a large amount of activity at the start, and the number of actions slowly decreases over time. This can be due to the fact that different sessions are completed or finished at different times. By normalising this data for the session time, such that 0 is equal to the first user action or load event, and 1 to the close or last user action, the density of actions can be observed. Figure 4.35 gives this density of actions over time and it clearly shows that the density of user actions is relatively stable, and actually slightly increases after the initial peak at the start.

4.3.4 Are there commonalities in the final mix?

The commonality between mix engineers is a question that helps identify artistically driven or practical mixing styles. Previous experiments have shown engineers are not always aligned with fairly stable scientific terms (Bitzer et al., 2008). This research question aims to answer the question of commonality by exploring four key concepts: loudness, spectral features, starting position and masking.

Loudness Differences

Previous studies have showed that experienced engineers tend to mix vocal tracks to sit significantly higher in the final mix in terms of loudness (Wilson and Fazenda, 2015b; De Man et al., 2014a). These studies measured a set of produced multi-tracks with the ITU-R 1770 loudness measurement to obtain the relative loudness of each track and concluded that vocals are approximately 5LU louder than the other tracks (International Telecommunication Union, 2011). This is a common trend in pop and rock genres, as the vocals are typically the focal point of the piece (Izhaki, 2012). When mixing for a balance mix, using the same measurement methodology rather than a full production, this bias towards vocal prominence is less significant. The Figures 4.1 to 4.5 give the relative loudness of each track compared to the mix.

Figure 4.36 presents the loudness of each track grouped by its instrument type. This shows that the bass instruments are often louder in the mix than the vocal tracks, although the vocals are still very prominent. This can be down to the fact that the bass instruments often are comprised of one track per session, whilst 'Vocal' can cover both lead (which would be at the fore of the mix), and backing or harmonic (which would be lower in the mix) (Izhaki, 2012). The results can also vary as the balance mix does not include any effects processing of the audio, which can inherently add or remove gain. Commonly a vocal track is processed using a dynamic range compressor to ensure it stays at a more consistent loudness level throughout the piece (Izhaki, 2012). This form of processing will boost the overall loudness of the track as it is artificially boosted when it would normally be quieter, as the human voice has a large dynamic range. It is also possible in the earlier mixing stages, that engineers may not yet place as high an importance on overall audibility than the musicality

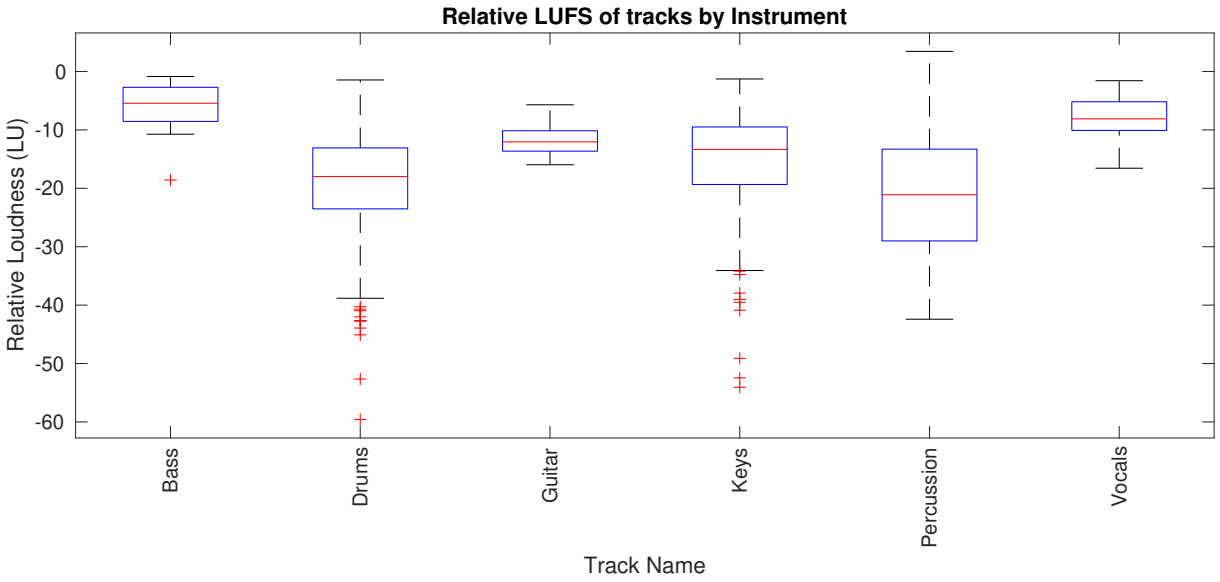


Figure 4.36: Relative Loudness (LU) of each track compared to the mix grouped by instrument type

of the tracks. For a song with no vocal tracks, there was no evidence of a clear lead track to replace it as shown in Figure 4.2 for Left Blind.

One major outlier from the other tracks is the prominence of the Bass heavy tracks, such as the 'Bass Guitar' in I'm Alright (Figure 4.1). In the previous studies the Bass instruments are usually part of the mix and do not significantly stand out, whilst here it is more prominent in the mix (De Man and Reiss, 2017). To test this significance, the instruments can be compared using the Wilcoxon ranked sum test. Table 4.24 shows the resulting p-values for the Wilcoxon rank test, where only the 'Toms' was significantly different to the 'Piano', 'Electric Guitar' and 'Lead Vocals'.

Table 4.25 shows the same Wilcoxon rank sum test but for the relative loudness of each track as grouped by their instrument types. The 6 main instrument types, Bass, Drums, Guitars, Keys, Percussion and Vocals encompas all 512 sample tracks from the test study. What is clear from these results is the distribution of the Bass and Vocal track groups are significantly different from the other four instrument groups. This is most likely caused by the restriction on processors and other production tools engineers would normally use. In a mix, there are usually fewer bass sources than mid and high frequency sources. This is supported by there being 38 bass tracks, versus 212 Drum tracks, 30 guitar tracks, 124 Keyboard tracks, 66 percussion tracks and 42 vocal tracks across the dataset. Therefore more processing is often required to provide the bass out of fewer sources, either through effects or by artificially adding more sources. By removing this aspect, engineers would have had to boost the only suitable bass source significantly to make up for the lack of lower frequency energy. This could also explain why the kick is louder than the other drums.

Comparing each mix together showed that, whilst individually the mixes could have larger dynamic ranges,there was not a large amount of variation between the mixes. This is explained by the lack of dynamics processing available, meaning each engineer was restricted to dynamics already exposed by track. Combined with the restrictions of editing and automation not being available, then the only dynamics that could occur is when

Track	Kick	Snare	Overheads	Toms	Percussion	Bass	Piano	Elec. Guitar	Ac. Guitar	Ld. Vocals	Back. Vocals 1	Back. Vocals 2
Kick	1	0.67	0.70	0.13	0.82	0.37	1	0.66	0.67	0.31	0.82	0.94
Snare	0.67	1	0.78	0.37	0.91	0.17	0.37	0.38	0.79	0.08	0.37	0.56
Overheads	0.70	0.78	1	0.18	0.82	0.17	0.48	0.37	0.90	0.24	0.70	0.82
Toms	0.13	0.37	0.18	1	0.59	0.09	0.02	0.03	0.12	0.03	0.09	0.09
Percussion	0.82	0.91	0.82	0.59	1	0.09	0.59	0.46	0.79	0.09	0.48	0.70
Bass	0.37	0.17	0.17	0.09	0.09	1	0.46	0.37	0.23	0.79	0.29	0.22
Piano	1	0.37	0.48	0.02	0.59	0.46	1	0.67	0.91	0.24	0.94	1
Elec. Guitar	0.66	0.38	0.37	0.03	0.46	0.37	0.67	1	1	0.22	0.90	0.78
Ac. Guitar	0.67	0.79	0.90	0.12	0.79	0.23	0.91	1	1	0.22	0.78	0.78
Ld. Vocals	0.31	0.08	0.24	0.03	0.09	0.79	0.24	0.22	0.22	1	0.24	0.24
Back. Vocals 1	0.82	0.37	0.70	0.09	0.48	0.29	0.94	0.90	0.78	0.24	1	0.48
Back. Vocals 2	0.94	0.56	0.82	0.09	0.70	0.22	1	0.78	0.78	0.24	0.48	1

Table 4.24: P-value results of the Wilcoxon rank sum test performed on the loudness data shown in Figure 4.1a.

Instrument Group	Bass	Drums	Guitars	Keys	Percussion	Vocals
Bass	1.000	<0.001	<0.001	<0.001	<0.001	0.102
Drums	<0.001	1.00	<0.001	<0.001	0.395	<0.001
Guitars	<0.001	<0.001	1.000	0.097	<0.001	<0.001
Keys	<0.001	<0.001	0.097	1.000	<0.001	<0.001
Percussion	<0.001	0.395	<0.001	<0.001	1.000	<0.001
Vocals	0.102	<0.001	<0.001	<0.001	<0.001	1.000

Table 4.25: P-value results of the Wilcoxon rank sum test performed on the relative loudness of tracks by their instrument type, shown in Figure 4.36.

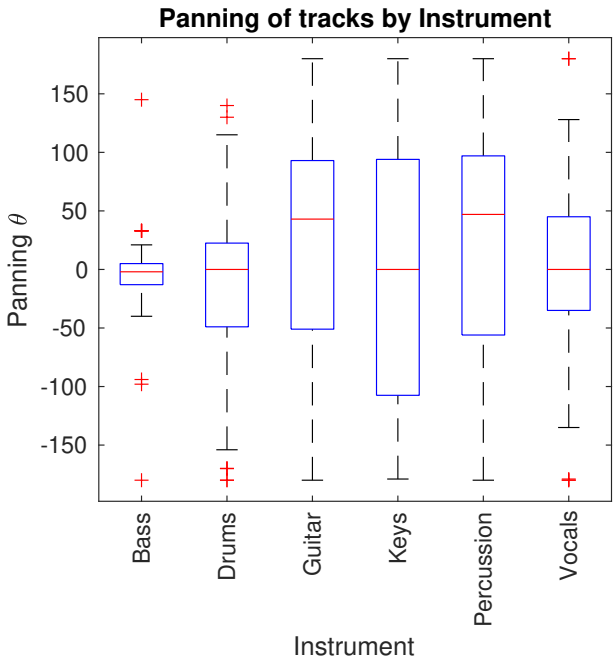


Figure 4.37: The pan controls used by instrument type across all 35 mixes

there is any interference between tracks when they are mixed-down. With further mix analysis from the engineers progressing further in the mix, some conclusions could be made. However these will be very specific to the structure and intent of the mix overall.

Spatial Differences

Data on the spatial positioning of the session was also gathered from each engineer through the pan position. The pan control takes a numerical representation of how much the signal should be panned to the left or the right. This is passed through a panning law calculation to determine a gain control for the left and right channtracksels. The equation 4.1 gives the sine-cosine panning law.

$$\begin{aligned} y_L[n] &= \sum_c (\cos\theta_c x_c[n]) \\ y_R[n] &= \sum_c (\sin\theta_c x_c[n]) \end{aligned} \tag{4.1}$$

Here, each channel's gain control is converted to an angle θ and passed into either the cosine or sine function depending if it is the left or right channel respectively. This panning law is often used as it has a natural dip when $\theta = 0$ such that both channels are equal to 0.707, or -3dB. This means when the two channels are summed together they give a small boost, but perceptually it is flat to the listener.

Some instrument types are almost universally centred, or to put in another way are mixed equally into both the left and right channels. Figure 4.37 shows the box plot distribution of the panning laws used, accounting for any panning that occurs through group routing as well. As can be seen, the bass instruments are centrally panned by majority with a tight distribution around the centre point 0. Drums and vocals also have centrally weighted distributions with low standard deviation and a number of outliers, whilst the other instrument types all have

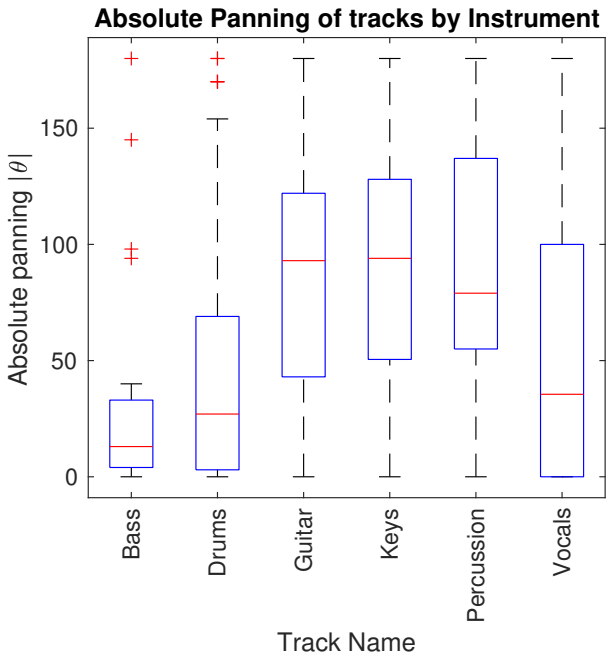


Figure 4.38: The absolute pan controls, showing distance from centre panning, used by instrument type across all 35 mixes

very wide distributions. This is because the pan can go either to the right (positive θ) or left (negative θ), so if a track is often panned off-centre but is not predominantly biased left or right, it will have a distribution which averages to the centre. This data can be refactored to use the absolute value of theta, such that the measurement represents the absolute distance from a central position. This is shown in Figure 4.38 for the tracks and shows the Drums and Vocals are more centrally panned.

The Drums are traditionally mixed to the centre with historical techniques and music education most likely having a significant impact on this (Izhaki, 2012). The Drum Kit is traditionally placed centrally on a stage in production due to its prominence in the mix. Therefore a lot of bands are recorded with the drums centrally panned to reflect this real-world positioning. The overheads are centrally panned as they were presented to the engineer as a stereo track and panning them would lose the spatial information encoded in the recording. Because the kick and overheads are centred, most other drums are centrally aligned with minor variations depending on the layout of the drums. For example, some engineers will exaggerate the spatial characteristics of the drum kit by mimicking the drum layouts. Figure 4.39 gives the information from Figure 4.37 except grouped by instrument sub-group. The percussive instruments such as the 'Kick Drum' are almost always panned to the middle of the mix, along with 'Snare Drum' and the 'Drum Kit' tracks. The lesser Drum-Kit instruments, such as 'Floor Tom', 'Hi-Hat' and 'Tom Drums' are more likely to be panned off-centre from the drum kit itself. 'Vocals' are very likely to be panned to the centre as well if it is the lead vocal, whilst the backing and supporting tracks are often spatially panned off the centre mark.

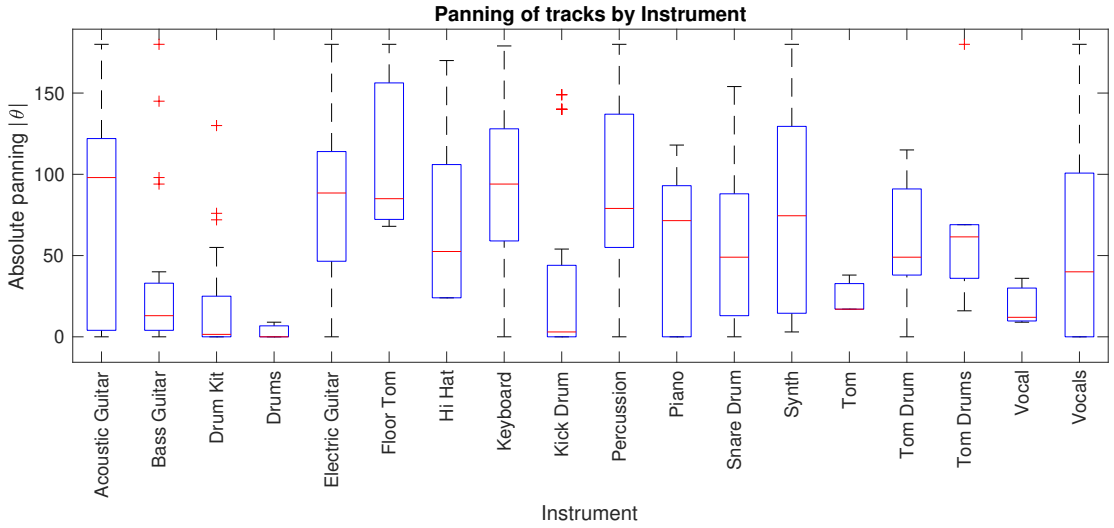


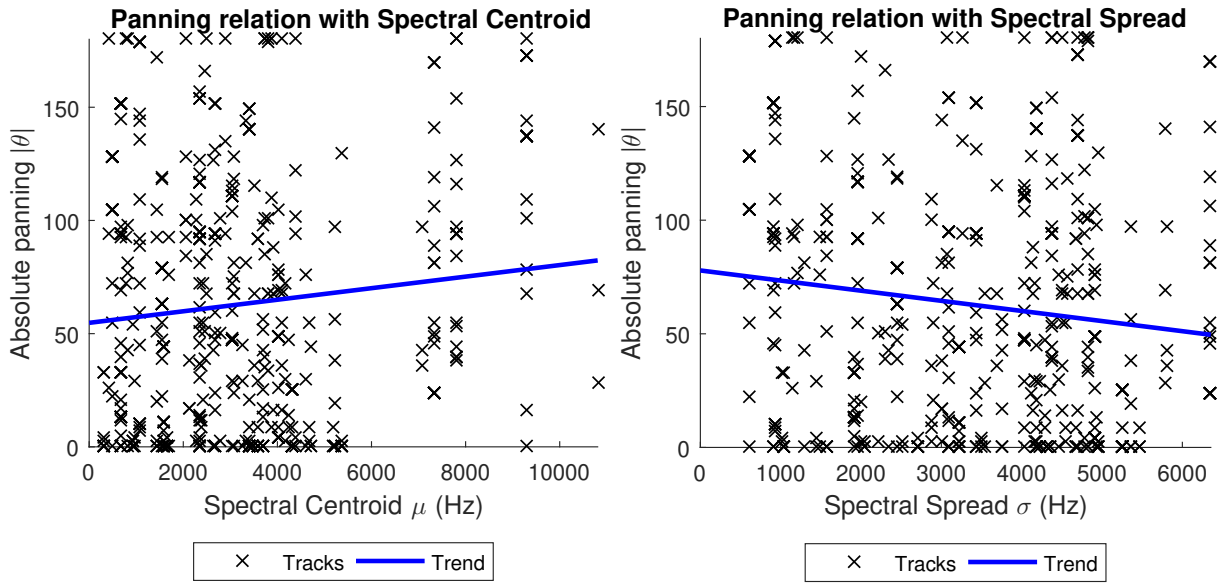
Figure 4.39: The absolute pan controls, showing distance from centre panning, used by instrument sub-type across all 35 mixes

Spectral Basis for Spatialisation

To understand why the engineers might have applied panning to a track, further analysis was undertaken. Previous studies have intimated that panning is related to the spectral content of the mix and should therefore be panned away from centre as the track exhibits higher frequency content (Perez-Gonzalez and Reiss, 2009). Two such measurements that can be applied to quantify the spectral content is the Spectral Centroid and Spectral Spread. Spectral Centroid (μ), defined in equation 4.2, where F_k is the central frequency of bin k and X is the incomming FFT, measures the gravitational mean of the spectral content, such that the frequency values of the FFT bin are weighted by the amount of energy in that bin. An equal power noise signal through the FFT, where the energies are perfectly flat, would give a spectral centroid equal to half the Nyquist frequency, whilst a signal which was only comprised of a single sinusoidal component would have a spectral centroid of the sinusoidal frequency. To calculate it in a discrete system, equation 4.3 is used instead. Bass heavy tracks, such as the Kick Drum and Bass Guitar, should have lower Spectral Centroids than high frequency tracks such as 'Electric Guitar' or 'Vocals'. Perez-Gonzalez and Reiss (2009) states that tracks with higher centroids tend to have more off-centre panning but this does not seem to hold true during the balance stage as shown in Figure 4.40. Here the panning laws show that with a spectral centroid there is a very weak negative correlation, where increased spectral centroid leads to narrower mixing styles.

$$\mu = \int F_k \frac{X_k}{\sum X} \, dk \tag{4.2}$$

$$\mu = \frac{\sum_{k=0}^{K} (|X_k| \, F_k)}{\sum_{k=0}^{K} |X_k|} \tag{4.3}$$



(a) Absolute panning related to spectral centroid

(b) Absolute panning related to spectral spread

Figure 4.40: Comparison of panning applied to tracks against the tracks spectral centroid and spectral spread.

The spectral spread shows the width of the distribution of the frequencies about the spectral centroid. A high spectral spread indicates a broadband signal, a lower spread a narrowband signal. The equation 4.4 gives the standard deviation of the spectral centroid μ, where F_k is the central frequency of bin k and X is the incomming FFT. Once applied to the discretised form the spectral spread can be compared against the apnning of the signal. This is shown in Figure 4.40 where there is little correlation with the panning angle given and the spectral spread of the signal, indicating that whether a signal is broad or narrow band has no impact on the panning placement.

$$\sigma^2 = \int (F_k - \mu)^2 \frac{X_k}{\sum X} \, dk \tag{4.4}$$

With panning, it has been shown that generally a mix should have equal amounts of spectral energy between the left and right channel (Pestana et al., 2014). Previous studies showed that for 928 mixes from the US charts, the average deviation in energy between the left and right channel was 0.8dB. The RMS (Root-Mean Squared) delta can be calculated by taking the RMS of the two channels as shown in equation 4.5. The incomming stereo signal x is split into its individual channels x_L and x_R. Each sample n is then squared and summed, before being divided by the number of samples windowed N. This is square rooted to give the RMS value and is then passed through the linear to log conversion in equation 4.6 to give the energy in dB.

$$\begin{aligned} RMS_L &= \sqrt{\frac{\sum_{n=0}^{N} (x_L[n])^2}{N}} \\ RMS_R &= \sqrt{\frac{\sum_{n=0}^{N} (x_R[n])^2}{N}} \end{aligned} \tag{4.5}$$

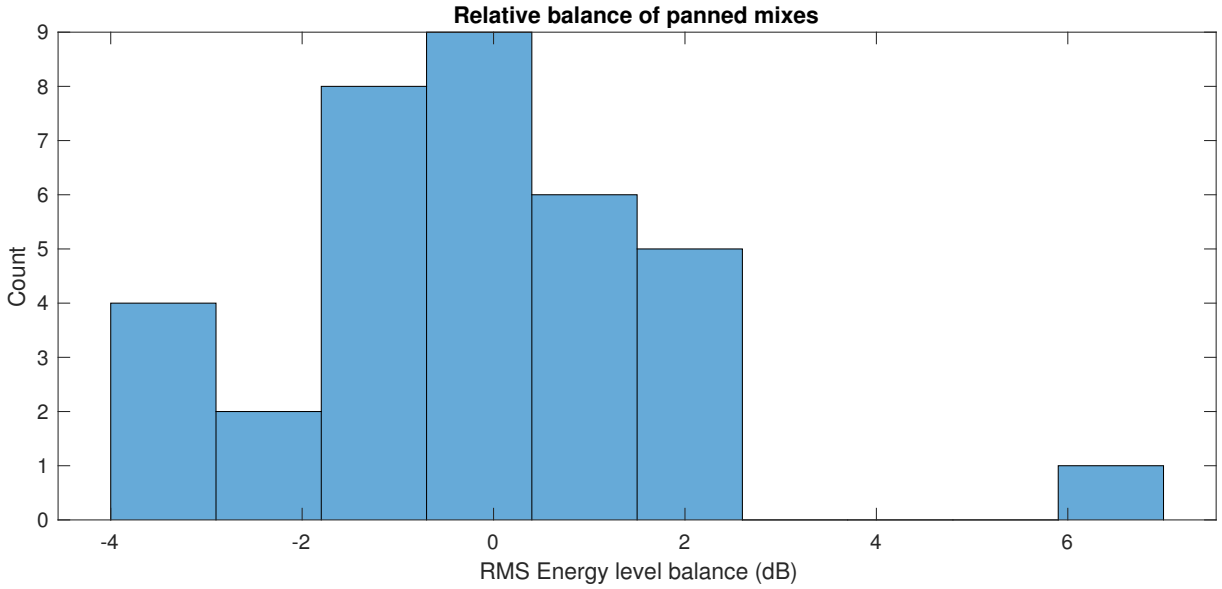


Figure 4.41: Histogram of the level differences between RMS levels of the left and right channels

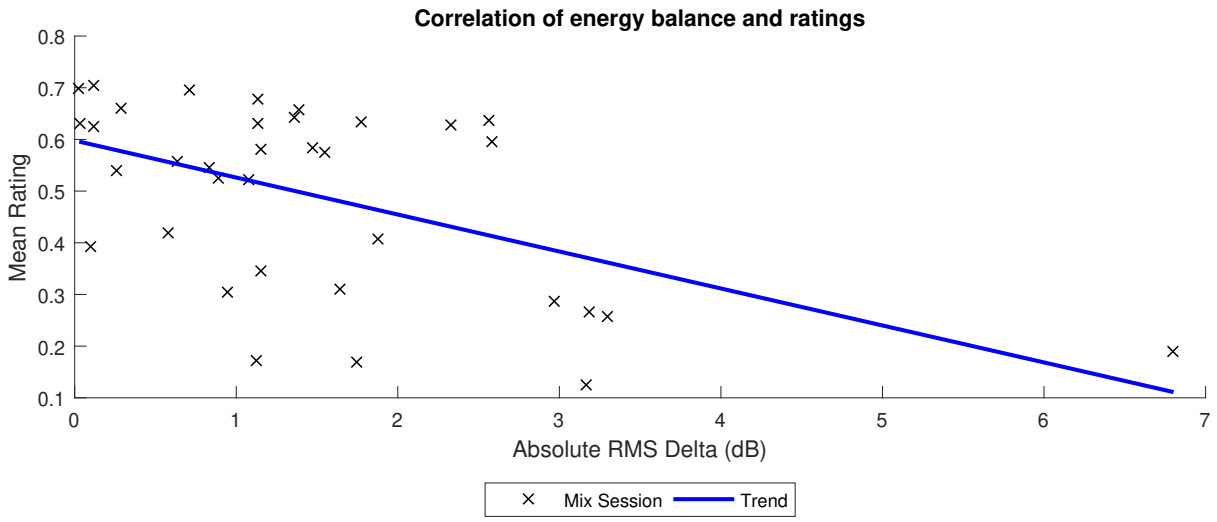


Figure 4.42: Correlation between the absolute level difference and the listening test ratings.

$$RMS^{dB} = 20\log_{10}\left(RMS_{Linear}\right) \tag{4.6}$$

Given the balance task is a stereo mix, every track is mixed as a stereo effect when it passes through the panning phase. This gives a figure for the amount of energy present in both channels. The RMS was calculated for the each of the two channels using the entire length of the mix. Then the delta RMS of the two channels is given as $\delta_{RMS} = RMS_L - RMS_R$. The delta figures are shown in the Histogram in Figure 4.41. 10 mixes had an RMS delta of less than 0.8dB between the two channels, or 28.57%. Most had an RMS delta of less than 1.2dB, slightly wider than the perceived level from the previous study. Overall, 19 mixes met this mark (54.28%). This could indicate that spatial balancing is less important at this phase and will be corrected at a later date.

When the listening test data is included it does show this is a very strong factor that engineers should consider. Taking the absolute RMS delta $|\delta_{RMS}|$ it shows that as the mix deviates from having a balanced amount of energy in the two channels the lower the score. The relationship is given in Figure 4.42 with the trend line in blue showing the negative correlation. As the mix becomes less balanced in energy between the two channels, the delta RMS gets bigger and therefore the score drops. The Pearson correlation here shows the negative correlation with $R = -0.577$.

Spectral Differences

The spectra for all 35 songs were extracted using the following process. All the mixes were loudness normalised to -23 LUFS using the ITU-R 1770 loudness measurement (International Telecommunication Union, 2011). Then each mix was split into 4096 sample windowed chunks, with a 50% overlap. The window effect applied to each sample was the Hann window function (Harris, 1978). This window is suitable for its rejection of side-leakage in most scenarios, and is recommended for broadband based random or noisy signals (Braun, 2001). The windowed frames were sent through the Fast-Fourier Transform to calculate the spectrum for each frame. Each frame was added on to each other and then the average of all the frames energy was taken. Figure 4.43 shows all 35 mixes plotted along with the average energy in blue. This result is comparable to previous studies, where modern productions exhibit a peak in energy around the 100Hz mark (Pestana et al., 2013).

Figure 4.43 shows the average spectrum for all 35 mixes. This analysis is harder to extract useful information from as the noisiness of the data is very high. To improve the quality of the data a perceptual model similar to the human ear to show the abstracted energy across a spectrum is used. The Mel-Frequency Cepstral Coefficients (MFCCs), which are a perceptual representation of the spectral energy, are calculated by passing the audio through a set of filter banks to calculate the energy in a set of auditory sub bands. The process takes the Fourier Transform of the signal to analyse, if working on a time-based decimation of frames then these should be windowed. The filter bank centre frequencies are calculated by first defining the upper and lower frequency bands, f_{max} and f_{min}, and the total number of bands N. These upper and lower bands are converted into Mels using equation 4.7.

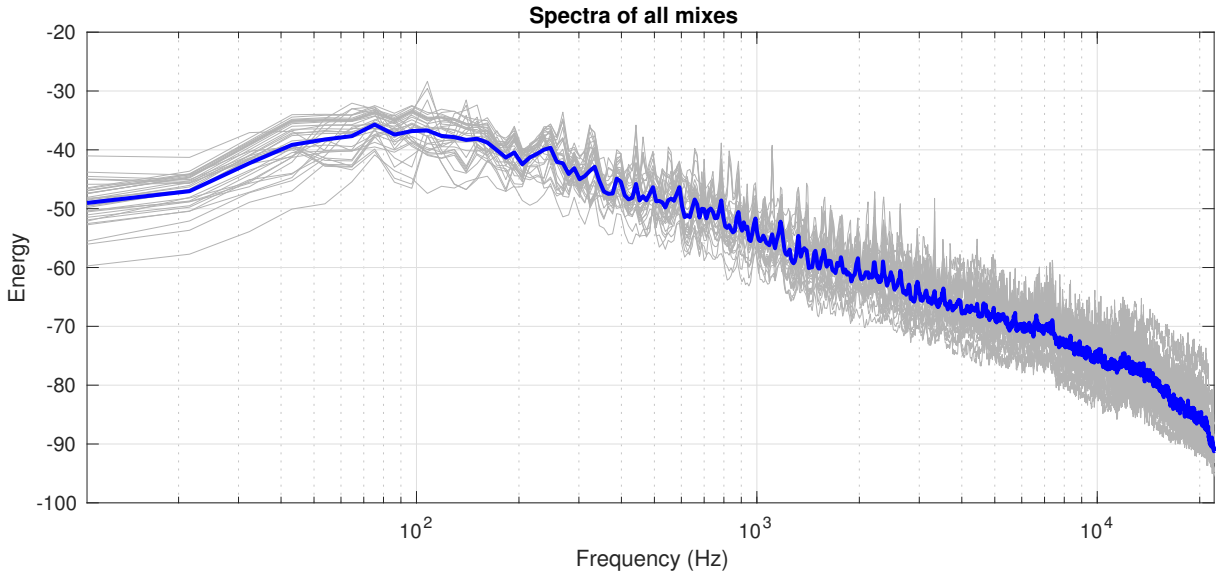


Figure 4.43: The average spectrum for all the 35 mixes.

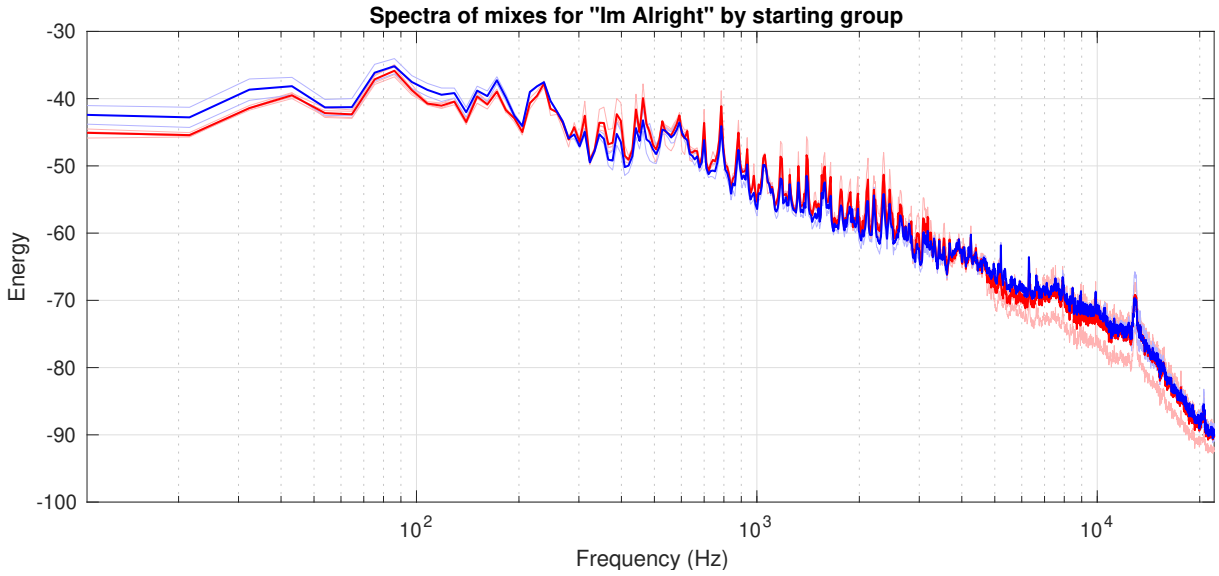


Figure 4.44: The spectrum of the mix for song I'm Alright with the average for the two starting groups

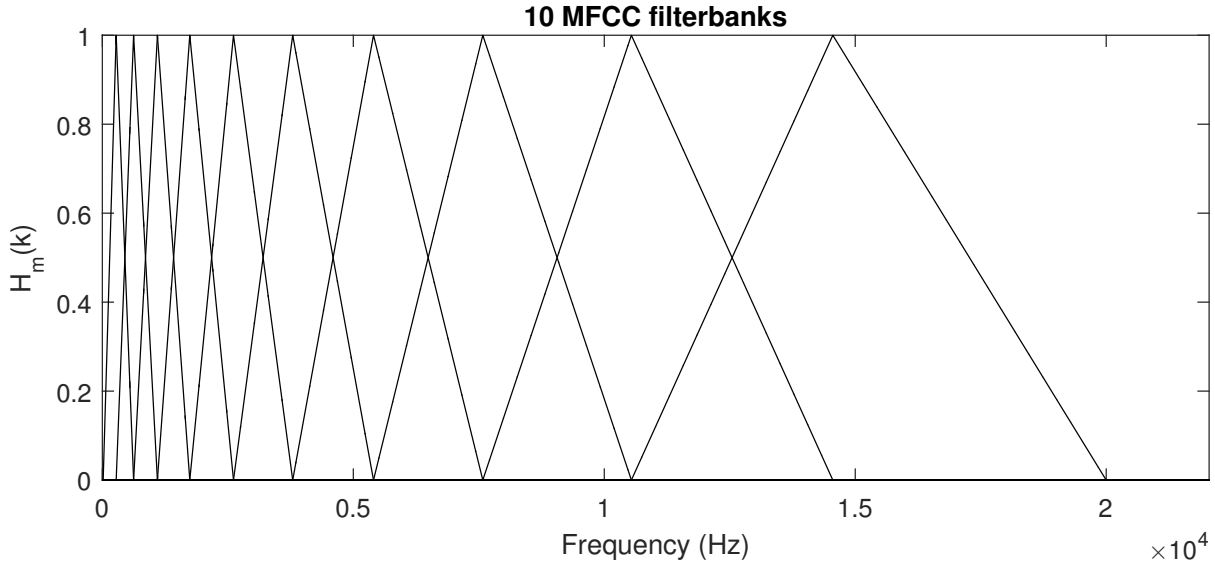


Figure 4.45: Plot of ten MFCC filter banks for a sample rate of 44100Hz.

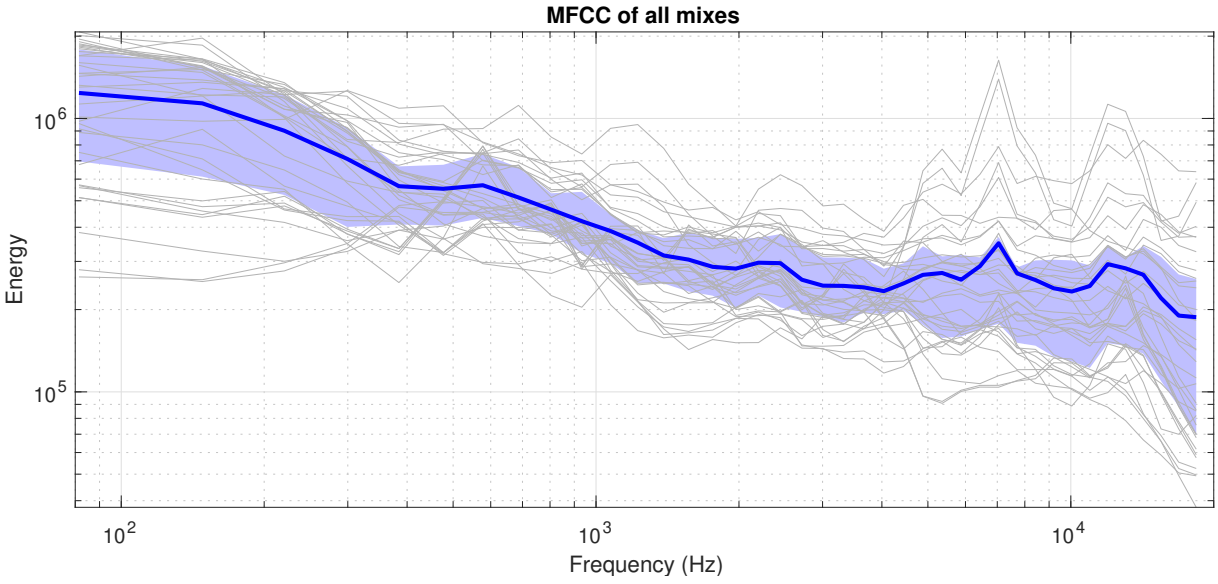


Figure 4.46: MFCCs for all 35 mixes (grey) with the mean (blue), 75th and 25th percentile (dark blue) and 95th and 5th percentile of the variation for each of the 40 MFCC bands.

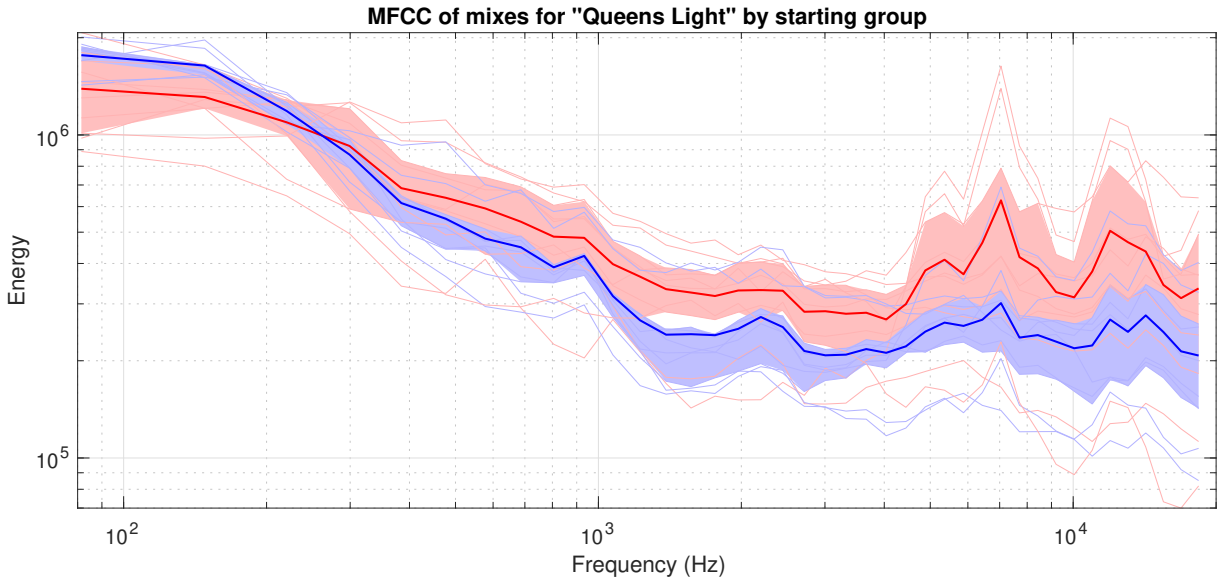


Figure 4.47: MFCCs of the song Queen's Light for the two groups (red and blue), with their averages and 75th and 25th percentile of the variation for each of the 40 MFCC bands.

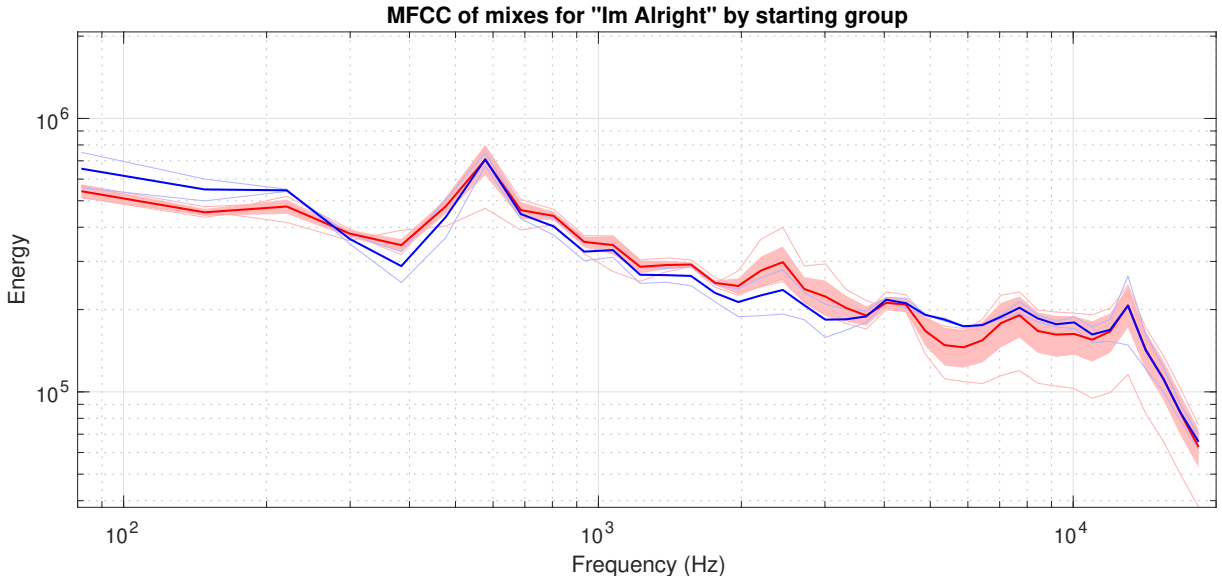


Figure 4.48: MFCCs of the song I'm Alright for the two groups (red and blue), with their averages and 75th and 25th percentile of the variation for each of the 40 MFCC bands.

$$M(f) = 1125 \ln (1 + f/700) \tag{4.7}$$

A set of frequency points are calculated, linearly spaced from $M(f_{min})$ to $M(f_{max})$ for $N + 2$ points, so the array starts and ends at the minimum and maximum points with N points in between. Then we pass the array of points through equation 4.8 to convert the Mel frequencies back to Hertz, h.

$$M^{-1}(m) = 700 \exp^{(m/1125)-1} \tag{4.8}$$

Since the frequencies will not map directly onto a set of FFT bins, the nearest bin is used instead, by passing the frequencies h through equation 4.9 to get the FFT bin number. Now the FFT bins are known, the triangle filters can be constructed using equation 4.10.

$$FFT(i) = \left\lfloor \frac{(nfft + 1) * h(i)}{F_s} \right\rfloor \tag{4.9}$$

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \leqslant k \leqslant f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \leqslant k \leqslant f(m+1) \\ 0, & k > f(m+1) \end{cases} \tag{4.10}$$

An example of these is given in Figure 4.45. The signal to analyse x is passed through the Fast-Fourier Transform to obtain X. The MFCCs are calculated using the pre-computed filter bank using equation 4.11.

$$MFCC(m) = \sum_{k=0} K \left(X[k] H_m[k] \right) \tag{4.11}$$

Figure 4.45 shows the MFCCs plotted for all 35 of the mixes for the five songs in grey. The mean of these is then calculated and plotted in blue, with the 25th and 75th percentile of each band shaded in dark blue, and the 5th and 95th in light blue. Most of the mixes had stronger low frequency energy than they do higher frequencies and confirms what the spectrum was showing but with more relevance to the auditory system.

Starting Positions

As can clearly be seen in Figure 4.47 for Queen's Light, varying the starting position of the mixing parameters results in significantly different mixes. The energy ratio is skewed to favour the lower frequency content versus the higher frequencies for the second group (blue) than the first group (red). This confirms previous works that the starting conditions have an impact on the outcome of the mix, and this effect happens at the very start of the mixing process with the balance mix (Wilson and Fazenda, 2015b). When combined with the panning positions for each track, a clear distinction between the two starting groups could be seen. Figure 4.49 shows

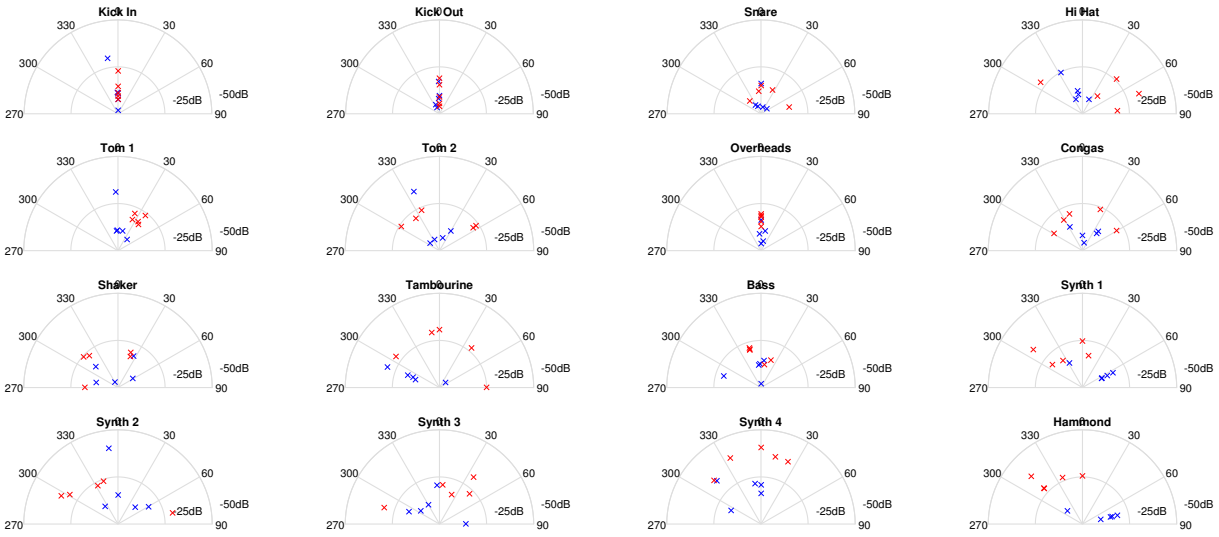


Figure 4.49: The 2D space representation of several tracks from group 1 (blue) and group 2 (red). The further from the origin (0,0) the quieter the source (higher perceived distance). The angle represents the panning position.

the mix-space position of tracks varied across the two starting positions for the song *Queen's Light*, which had 10 participants for both groups.

There was little variation for the loudness or action history depending on the starting position. Figure 4.49 shows the mix-space position of tracks varied across the two starting positions. There are some immediate comparisons between all the mixes, such as the Kick Drum being centrally panned, with a greater degree of variation in the volume position between the mixes. The same is also true for the stereo track 'Overheads', which is a single track for both the left and right spaced overheads on this mix. All other instruments had a degree of variation in the panning, however certain commonalities of the mixes can be seen especially between the groups. 'Tom 1', for example, shows a cluster of panning around 30R and a volume control around -20dB. The Synth and Hammond instruments also show similar cluster performances for the groups. This shows that whilst there are defined rules that mixing engineers are following, the initial perception of the mix does have a measurable impact on the mixing decisions that the engineer may make.

Clustering can be done by measuring the distance between each control surface point and comparing against the distance to another point. K-Means clustering would be one suitable measurement. This method requires the specification of clusters to be known before hand. Each cluster is given a randomised centroid position to start with and on each iteration the entry points are assigned to the cluster with the shortest euclidean distance. Then the cluster centroid is updated by taking the mean position of all the data points assigned to that cluster with the whole process repeating until no changes in the cluster centroids occur. Over-estimating the number causes the algorithm to over-fit, where a single cluster is split into multiple sub-clusters.

The method used here is the Agglomerative Hierarchical Clustering algorithm (Maimon and Rokach, 2005, pp. 321-352). This starts with each entry point being assigned to its own cluster and then progressively joins

the data points together as pairs of clusters, forming a new cluster as it moves up the hierarchy. Each of these cluster pairs forms a leaf which can be converted into a distance score, allowing groups which are closely related to be positioned near each other, whilst unrelated clusters are positioned further away. To start with, each initial cluster centroid is calculated to give the position of the cluster in the space. This is similar to running k-means except on each entry cluster at a time. In this example, there are 16 tracks so there are 16 clusters to calculate, each with 20 co-ordinates representing the pan and volume positions.

The distance from each cluster centroid is calculated using a distance metric, commonly Euclidean distance, given in equation 4.12 (Tabak, 2004, p. 150). This compares two vectors a and b by finding the difference between each point. These are then squared and summed together before being sent through the square-root.

$$d = \sqrt{\sum_i (a_i - b_i)^2} \tag{4.12}$$

The Euclidean distance has the advantage that it provides a Cartesian distance measurement, which means the angle between the two vectors does not affect the distance since it is based upon Pythagoras theory. For this reason it can be found referred to as Pythagorean distance. An alternative to the Euclidean distance is the Manhattan distance. Instead of calculating the hypotenuse length of the triangle between two points, the Manhattan distance calculates the distance along the axis. This is the sum of the absolute values of the differences between the dimensions given. This is shown in equation 4.13.

$$d = \sum_i |a_i - b_i| \tag{4.13}$$

A final commonly used distance metric is the Chebyshev distance or maximum distance, equation 4.14. Instead of returning the sum of distances using the Manhattan distance, it returns the maximum distance along a given axis.

$$d = \max_i |a_i - b_i| \tag{4.14}$$

Of these common methods the Euclidean distance was used to calculate the distance between the clusters.

Once the cluster distances are calculated the hierarchical tree can be constructed using a linkage criterion (Murtagh and Conreras, 2012). This determines the distance between the sets of observed data as a function of the distance between the observations themselves. As with the distance, different measurements can again be used to calculate the distance between the cluster sets. Single Linkage and Complete Linkage calculates the distance between two clusters as either the minimum or maximum distance between any two members of each cluster respectively. Average Linkage algorithms calculate the distance between two clusters as the average distance between every single member. And centroid linkage calculates the distance between two clusters as the distance between the centroid of the two clusters. Once the distances have been identified, the process will then select each pair when closest to the next pair based on the distances.

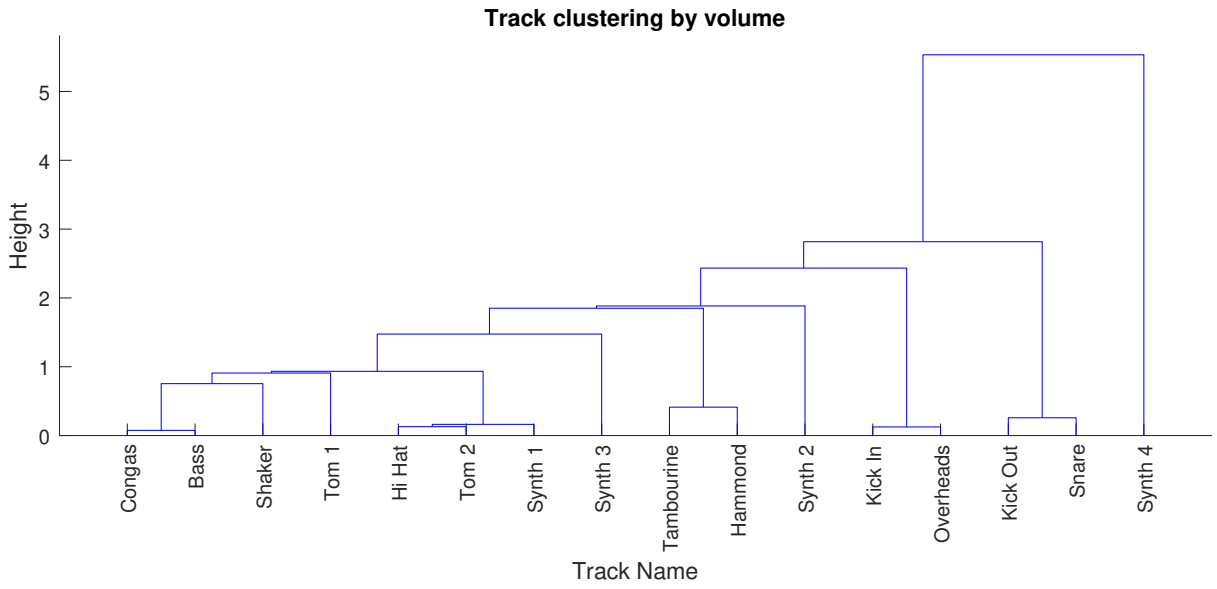


Figure 4.50: Distance measurements between all the tracks compared across all 10 mixes of the Queen's Light song for the volume control.

With the linkage complete, it can be visually displayed using a dendrogram. This shows the pairings between each cluster set with their distances. Nodes which are closely related are placed near to each other due to the leaf structure. This shows not only how the clusters are related but how close that relationship is.

Figure 4.50 gives the distances for the song Queen's Light based on the volume and figure **??** for the pan position of each track. This shows for all twenty mixes of Queens Light the distance between each of the tracks based only on the volume or pan position. The volume and pan positions used are calculated as the mean value based on the 10 incoming volume or pan controls from the 10 mixes. The clustering shows there is a strong relationship between the Kick Drum and Drum Overheads in terms of the pan control. Most educational texts stipulate that the Kick Drum should be centrally panned, based upon the stage metaphor that a drum kit is centrally placed (Izhaki, 2012; Senior, 2019).

The 'Overheads' track is also a stereo track, meaning it already has the spatial information provided, therefore this would normally be centrally panned. Bass Guitar is also closely related, but not as consistently and hence is grouped with the other tracks. By running the hierarchical clustering again with the absolute pan metric, the distances show more useful linkage information. The Kick Drum and Overheads are still linked, but the Hi-Hat and Toms show a clustering relationship as well. With the Synth and Hammond also showing a weaker but relational link with their pan settings. This shows that groups of instruments would tend to be positioned together, or at least their positions are related to each other.

For grouping by volume the reasons for groupings are not clear. This is because the volume controls themselves are heavily dependent on the content. A track could be boosted because it is important, or because it is quiet. Since the dendrogram was built using the raw volume control data, the loudness of each track should be used in its place instead. This is highlighted by the cophenetic score of the volume linkage being $C = 0.7093$, which

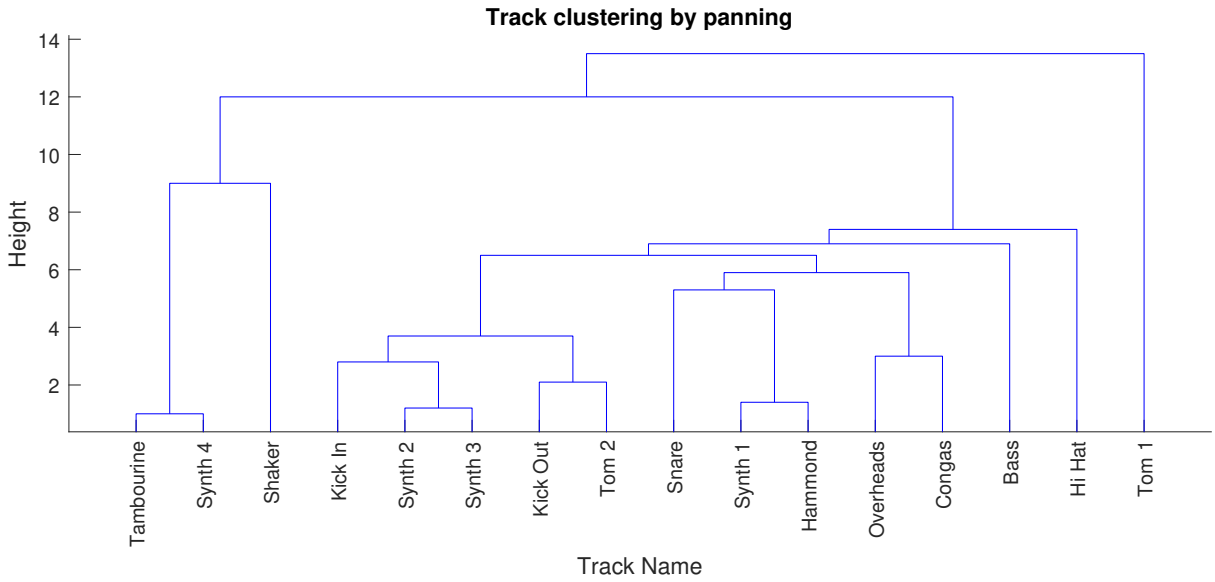


Figure 4.51: Distance measurements between all the tracks compared across all 10 mixes of the Queen's Light song for the panning control.

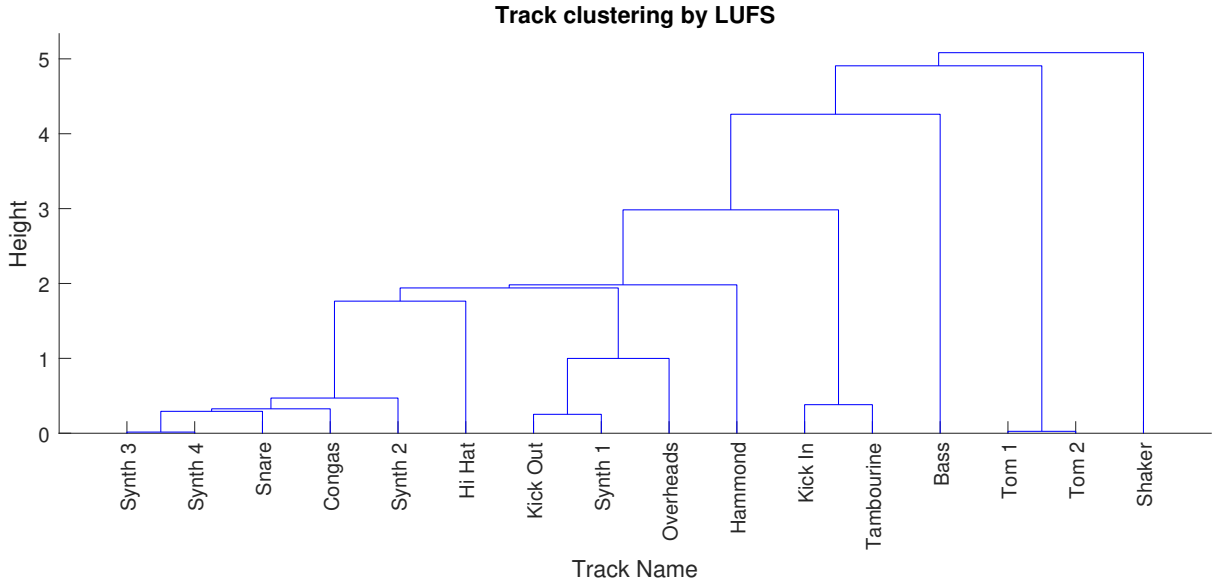


Figure 4.52: Hierarchical clustering of the tracks for Queen's Light based upon the Loudness Unit (LUFS) of the processed track.

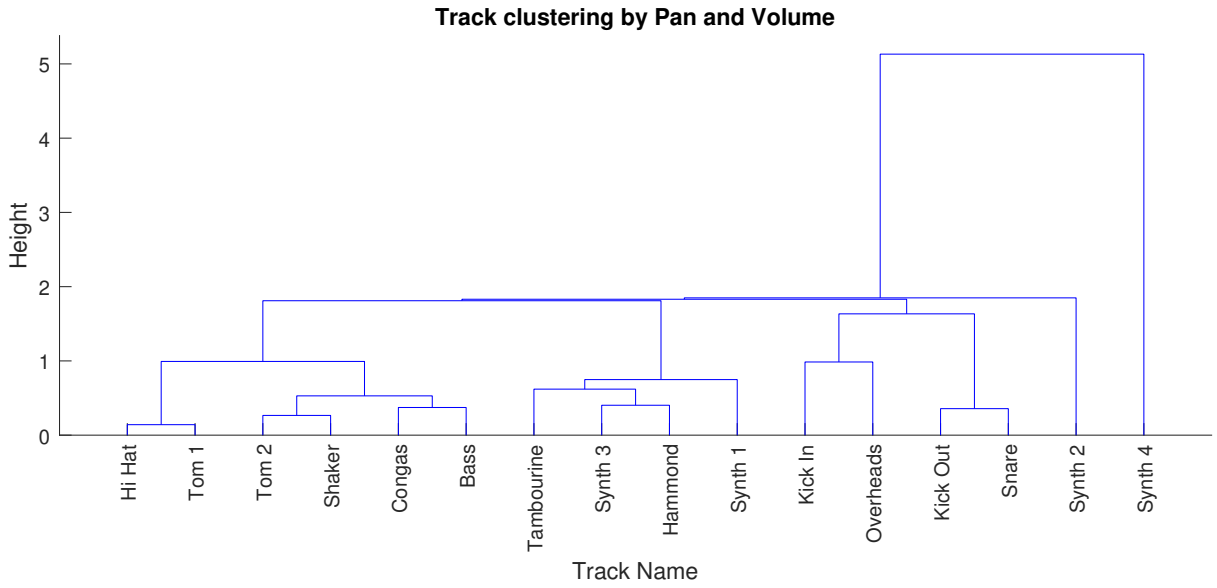


Figure 4.53: Hierarchical clustering of the tracks for Queen's Light using both the volume and pan controls

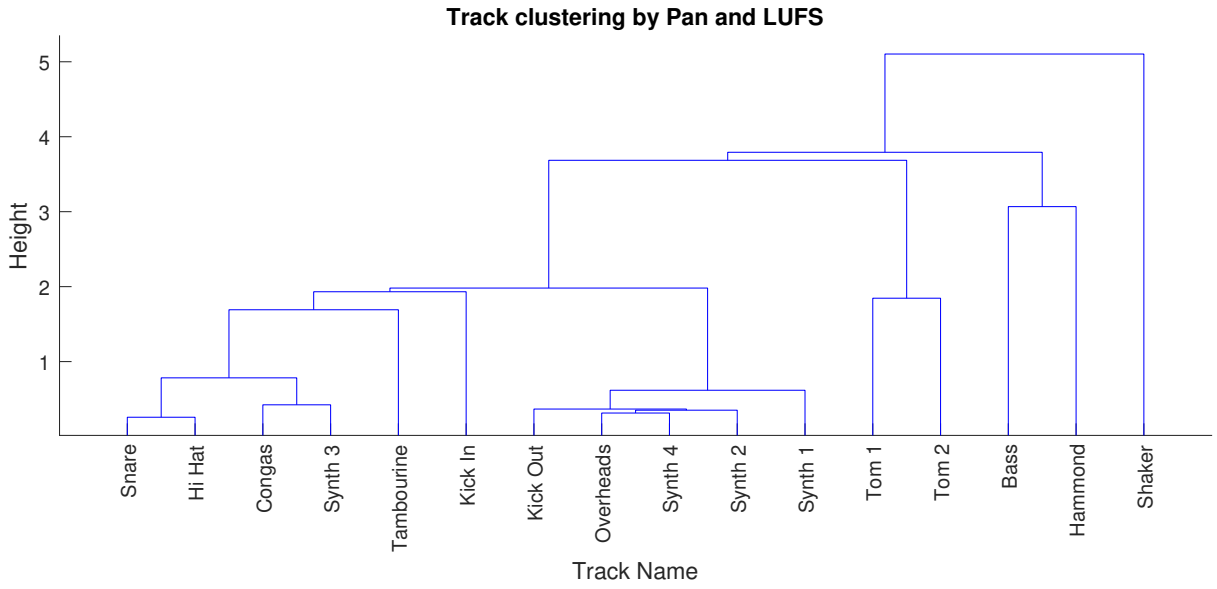


Figure 4.54: Hierarchical clustering of the tracks for Queen's Light using both the LUFS and pan controls

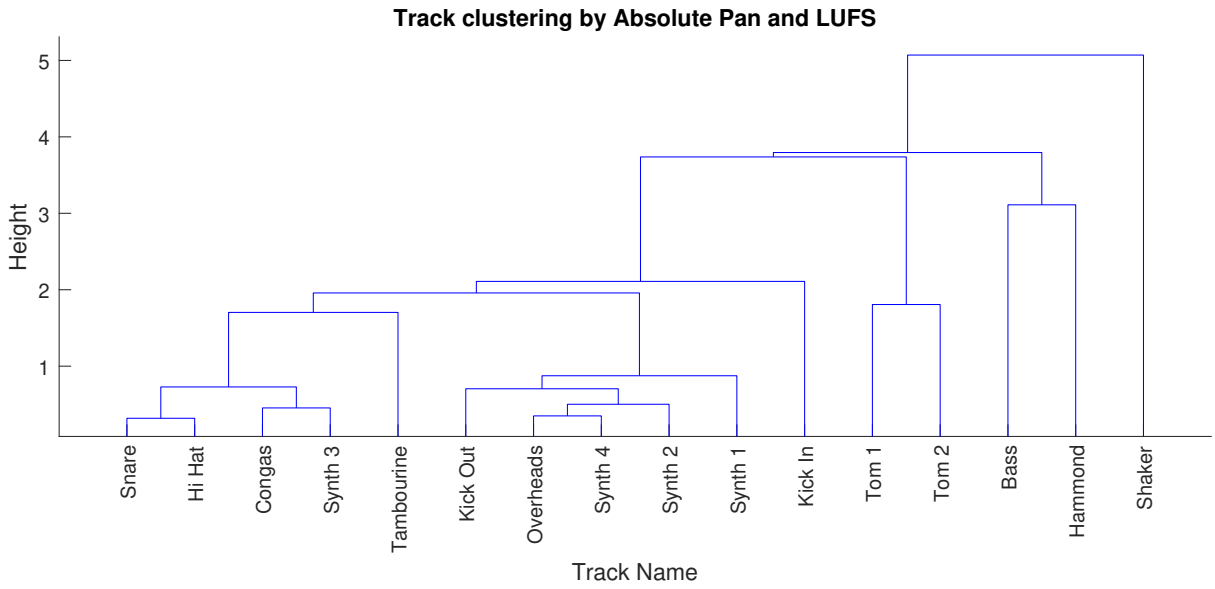


Figure 4.55: Hierarchical clustering of the tracks for Queen's Light using both the LUFS and Absolute Panning.

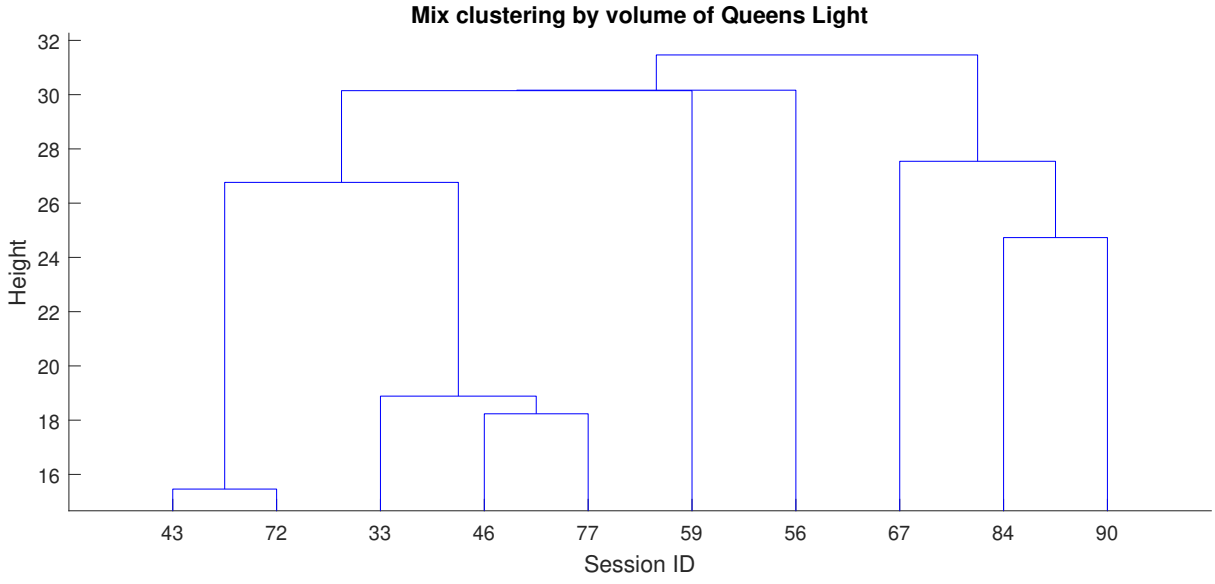


Figure 4.56: Hierarchical clustering of the mixes for the Queen's Light using the volume positions.

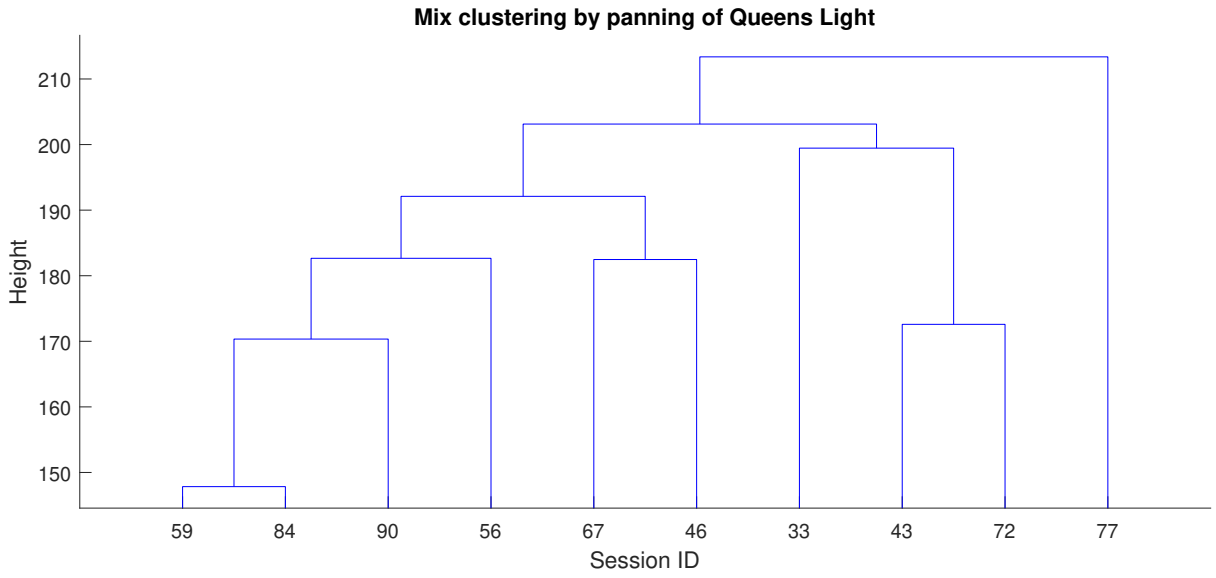


Figure 4.57: Hierarchical clustering of the mixes for the Queen's Light using the pan positions.

is not a particularly high score for the clustering, showing the data itself has a high degree of spread. For example, the 'Kick In' has a range of -10dB to +10dB on the volume control across all 10 mixes.

By applying the gain to each track and using the ITU-R 1770 Loudness Calculation, the loudness of each track can be obtained (International Telecommunication Union, 2011). Performing the clustering using this method shows a more detailed structure shown in Figure 4.52. In this case, the two 'Tom' drums are placed very near to each other, which makes sense as these are going to have a similar loudness property and balance in the mix. The 'Kick In' and 'Kick Out' are also positioned near to each other, showing they are related quite strongly together, although they do connect to different leafs of the linkage tree. There is a strong correlation between the Synthesizer tracks, Tambourine and Hammond Organ. This scored a high cophenetic score of $C = 0.8042$.

$$v = 10^{G_{dB}/20} \tag{4.15}$$

$$G_{dB} = 20\log_{10}(v) \tag{4.16}$$

Of course the volume and pan are related to each other, to position the instrument in the mix correctly. Combining both of the sources together required calculating the clusters before processing. The volume controls were converted from their decibel value to a linear value by using equation 4.15 to get a magnitude. The pan control was converted from the control range of +/- 180 in degrees to radians. This gives the polar coordinates of the two parameters, as visually shown in 4.49. Once all ten variations of the track are converted, the means of the volume and pan polar controls are calculated. The means are then passed to the Euclidean distance algorithm to calculate the distances between each point, and linked using the *Single linkage* method, linking together the clusters with the shortest distance. Figure 4.53 gives the dendrogram plot of the linkage. The

linkage scored a cophenetic score of $C = 0.7495$ which shows the clustering is stronger than just using the two individually, but the groupings also still have the same issue of using volume not loudness.

Repeating the clustering for loudness instead of the volume control the score increases to $C = 0.8117$. This is shown in Figure 4.54. This shows the strong relationship between the 'Kick Out', Overheads and the Synth instruments. This makes some sense since the Overheads and 'Kick Out' captured the majority of the drums together and the Synth make up most of the rest of the mix, being a very important and prominent part of the song. Other strong pairings were the Snare and Hi-Hat, which again could be due to education that these are panned together, and therefore may have the same perceived location and loudness.

As shown in Figure 4.49 the panning can be either left based or right based, with engineers often favouring either side. Therefore the mean of the pans themselves could be misleading, as a pan control with 4 strong left and 4 strong right entries would have a mean pan control near the centre, which is not accurately reflecting the engineer decision. The final linkage that we performed had the pan controls set to absolute, that means the pan is just a distance from the central position, removing the left and right bias. Figure 4.55 gives the measurement using the LUFS and absolute pan position. With a cophenetic score of $C = 0.8134$ this scores the lowest dissimilarity score of the linkages produced. The main change is the 'Kick In' is moved to be closer to the Tom Drums instead of the Synth 3 and Conga, but mostly the distances for the leafs are better representing the clusters themselves.

These results show that certain tracks are intentionally positioned near to or away from other tracks in the mix. Tracks with similar content, such as the 'Kick Out', 'Kick In' and 'Overheads' are often placed in similar locations due to the fact that the tracks represent the same source-types. Tracks with conflicting content are often spaced away from each other in the mix, such as the Shaker and the Snare, or the Hammond Organ and the Synthesizer. These would potentially have masking issues, which the engineer is trying to reduce.

Flipping the data however, such that the engineer mixes are the dependent variable, allows analysis of the similarity of the mix space created. Each engineer produces 16 pan and volume positions to summarise their mix, therefore the mix is a 16 element complex vector representing the completed space. Calculating the distances between each mix engineer is trivial and therefore hierarchical clustering can be performed to show if any two engineers are similar. Figure 4.56 shows the dendrogram for the 10 mixes created for Queens Light, grouping by the volume, with figure 4.57 giving the grouping by pan. Mixes 1 to 5 are the first group of engineers and mixes 6 to 10 are the second group. What is immediately clear is there is a strong separation of the groups based on the two controls.

The volume shows a strong clustering around the first group, with mixes 7 and 9 forming a very tight correlation. Mixes 1-5 are less strongly linked, but shows they are similarly related. The pan control was less indicative on its own, with some agreement but the distances are very high. Using the absolute pan measurements shows a much tighter set of agreements for the two groups, with a clear distinction between 1 to 5 and 6 to 10. This clearly shows the engineers do approach different mixes based upon the starting point of the mix, as the only difference between the two groups was the track starting positions.

Masking Minimisation

The original formula by Aichinger et al. (2011) for uncovering the masked to unmaskted ratio required the loudness of the track in question to be compared to the loudness of the mix overall. The models proposed by Moore et al. (1997), and implemented by Ward et al. (2012) return two metrics: Short-Term Loudness (STL) and Short-Term Partial Loudness (STPL). The Short-Term Loudness is the loudness of the mix in free-field, unaltered by the other tracks or sources being passed through. The Short-Term Partial Loudness is the loudness of the mix with masking applied, and represents the loudness of the track as perceived after masking has taken place. Therefore, these two numbers can be used to calculate the same ratio, of how much the original track is present in the final mix. Equation 4.17 shows the completed formula to achieve this, along with the definitions in Equation 4.18 and 4.19.

$$r[n] = \frac{\sum_{k=0}^{K} f\left(STPL_n[k], STL_n[k]\right)}{\sum_{k=0}^{K} g\left(STPL_n[k], STL_n[k]\right)} \tag{4.17}$$

$$f(x,y) = \begin{cases} (\frac{x}{y})^2, & \text{if } x > 0.003 \text{ and } y > 0.003 \\ 0, & \text{otherwise} \end{cases} \tag{4.18}$$

$$g(x,y) = \begin{cases} 1, & \text{if } x > 0.003 \text{ and } y > 0.003 \\ 0, & \text{otherwise} \end{cases} \tag{4.19}$$

To calculate the ratio of masked to unmasked energy in each given frame, the ratio of partial loudness against total loudness can be used, giving a similarly suitable metric to measure masking. This is provided in Equation 4.18, where x is the $STPL_n$ and y is the STL_n. As shown in the equations, both are required to be above 0.003 sones, as in the original source to match the threshold for human hearing. If they are, then the ratio of both is taken and then squared. These are iterated over each frame to get a total summation of these ratios and are then divided by the total number of matched frames to give a mean-squared representation of the loudness ratio into $r[n]$. This r contains the ratio for each track as a masked-to-unmasked ratio, where 0 means a track is completely masked and 1 a track completely present. To get the average of all the tracks, and therefore the masked to unmasked ratio, the mean of the vector r is taken.

Figure 4.58 shows the Masked-to-Unmasked (**MUR**) of the 10 mixes of Queen's Light by track. The 'Kick Out' and 'Tom 1', for example, have very poor MUR ratios, showing that nearly all of this track's content is masked in the final mix. This is most likely a failure of the way masking is measured. The system takes each source microphone individually, and compares against the energy present in all the other microphones. A drum kit often has several microphones attached, but the microphones will pick up the other drum elements as well as the drum they are focused on. In some cases, such as the 'Kick Drum', there will be more than one microphone attached. This means the masking calculations will interfere with each other, causing them to be in conflict when the underlying source itself is not masked at all.

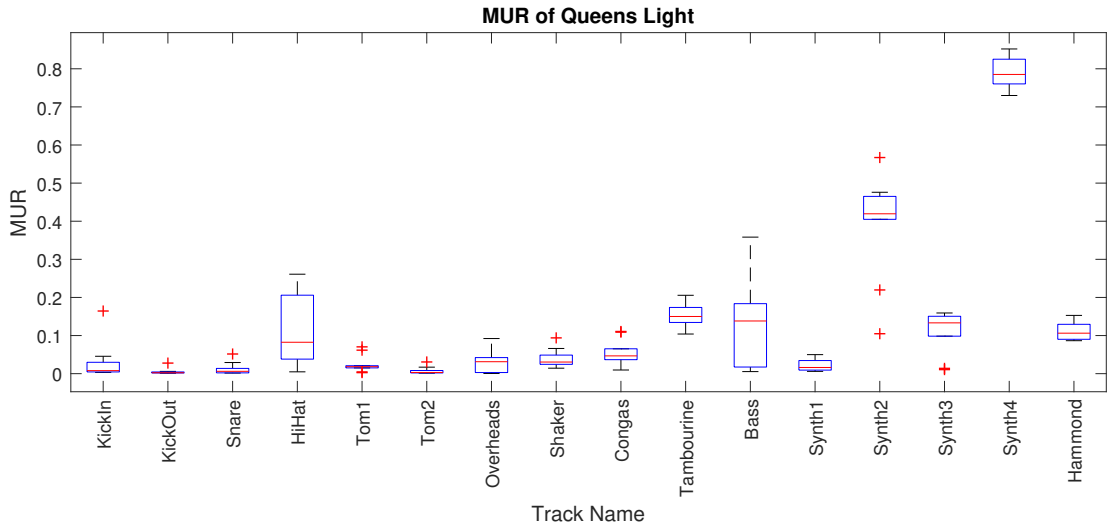


Figure 4.58: The masked-to-unmasked ratio of the song Queens Light of all 10 mixes by track.

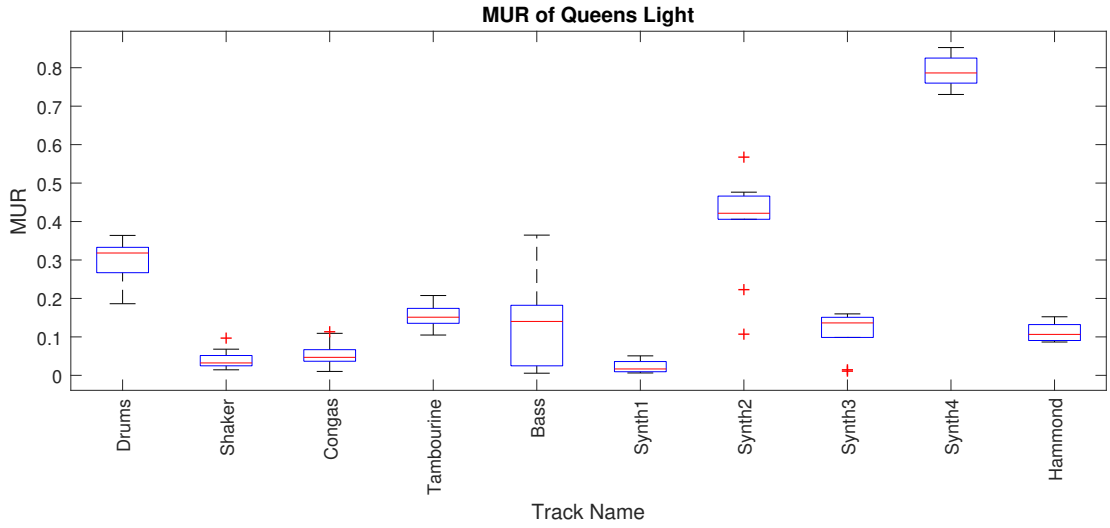
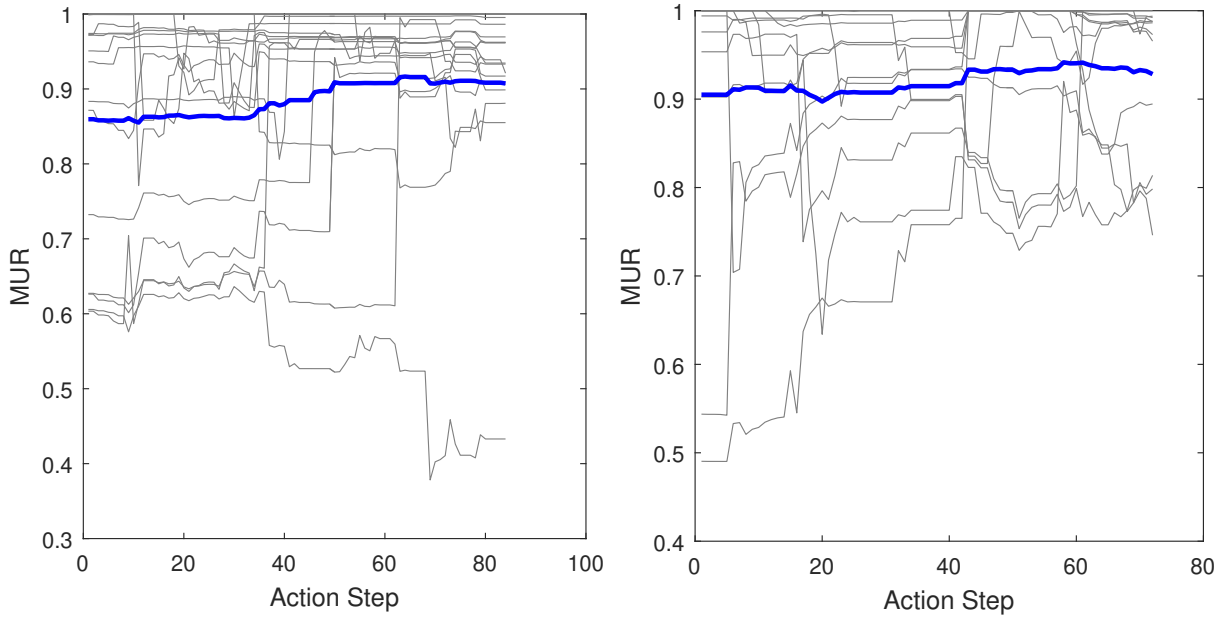


Figure 4.59: The masked-to-unmasked ratio of the song Queens Light with the Drums tracks combined

To process the data without the microphone spillage, the data was recalculated with the 7 drum tracks combined into one. Each mix was constructed as before and the drums combined into a single track. As can be seen in Figure 4.59 the drums perform better than their individual tracks. This shows that the drums were counteracting themselves when being combined for the masking model. With the drums no longer conflicting, the MUR increases, with the other tracks remaining unaffected as intended, since the masker signals are not changed for them.

From this, it is clear that masking has a very high form of agreement in the mix space, compared to panning actions or loudness decisions. Looking at the mix evolution, it is clear that engineers tend to reduce masking as they mix. Figure 4.60 shows two of the 35 mixes, one for Queen's Light and the other I'm Alright. In both situations, the grey lines trend upwards from 0 to 1, indicating that as the mix progresses the engineer will attempt to reduce as much masked content in the mix. Even without suitable processors to control microphone



(a) QL-46 showing the MUR of each track after each action occurs

(b) IA-47 showing the MUR of each track after each action occurs

Figure 4.60: The masked to unmasked ratio as the mix progresses over each action change of a volume or pan control for two sessions

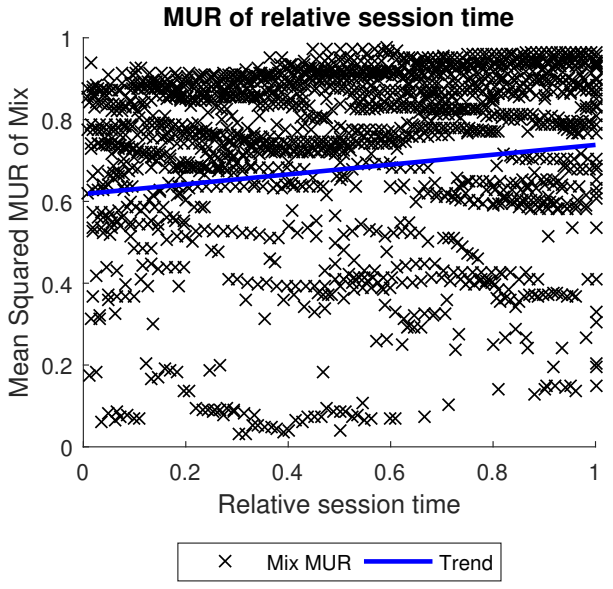


Figure 4.61: The masked to unmasked ratio of all mixes at each significant mix stage, relative to the session time normalised.

spill, spatial, spectral or temporal effects to help improve the situation, engineers are still reducing the amount of masking. Interestingly, this cannot be done to completion, with a few tracks always having a high degree of masking, but the mean (blue) is trending lower.

Over time, the trend for each session is to decrease. With the time normalised between 0 and 1, with 0 being the start and 1 being the end of the mixing session, it is possible to compare multiple mix profiles together. Figure 4.61 shows the various mixing decisions of each mix over this normalised relative session time. Each time a mixing decision is made which alters a tracks overall pan or volume control, a new mix was created and the MUR of each track calculated. Then the mean MUR was extracted and plotted onto this graph. The blue line shows the trend of the system to decrease over time, with a Pearson correlation of -0.1126. Whilst this is not a strong positive correlation it does show the mean masking energy will decrease as the mix continues.

The masking of each track is modified by the engineers decisions and shows engineers aim to reduce the average masking of tracks over time. Whilst there are several tracks who are deliberately left being masked, this appears to be down to tracks which either have significant overlap because of spill or multiple microphones recording the same instrument, for example drums, or because they are less important to the final mix.

4.4 Conclusion

From 35 mixing sessions, over 10 hours of mixing was captured in a re-playable environment which has allowed for detailed analysis. The collection of this data was not found in previous bodies of work, and has shown a detailed insight into audio engineering practices. From previous studies by De Man et al. (2015), Wilson and Fazenda (2015b) and Ronan et al. (2015a) it was clear that engineers have a significant impact on the mix, although all seem to follow certain guidelines, such as always boosting the vocal tracks (Wilson and Fazenda, 2015b). This was supported by a listening test to validate decisions by the engineers showing which properties of their mixing environment, decision making process or practices were helpful or detrimental to the overall perception of the mix.

4.4.1 General Findings

Generally, the engineers that were evaluated during this chapter first approach any mixing situation by listening to it. Auditioning was first action for 33 of the 35 sessions. Interacting with the audition system, to either start, stop or pause playback, accounted for 908 of all actions (28.90%). This importance of constantly auditioning the work played an important role in the overall success, with a positive correlation between minutes of audition and the rating given to that mix. Overall engineers spent 5 times as much time with the DAW in a playback state than they did when it was silent. For that state, the engineers would perform some specific actions mostly when auditioning, such as volume or pan control and soloing a track. These decisions are integral as these control the information the engineer perceives and affects the final balance of the mix. Over time, the magnitude of changes in the volume control gets smaller, indicating the engineer uses the loudness to balance the mix and approach a good level and mix. Panning does not have the same decrease in magnitude over time.

4.4.2 Session Structure

Busses form a way to divide the sessions up into smaller, more manageable chunks. Previous studies did not show when busses are created, just that sessions with busses had better scores (Ronan et al., 2015b). From this study it is clear that busses are created as easily as possible for the role of grouping. And in this study, even though there are no processing options for the busses to do, they still resulted in better mix ratings. When busses are used in a session, they tend to cover the entire session, as in all input tracks are sent to a bus.

These findings seem contradictory when busses are designed to lower the amount of work, but the most common bus size was 1. For balance mixes, the majority of busses were created only for the role of grouping, by creating a sub-mix. Of these groups, the most common was for 'Drums' with 19 busses created, followed by Guitars and Percussion. The group names all were for instrument labels, showing that groups are used primarily for grouping similar instruments. A few busses were created for sends, which is used to duplicate the track. This is primarily done for parallel processing (Izhaki, 2012) but since there were no processors available there would be no advantage at this stage. There were 5 send busses created with the label 'Verb', which is a shorthand for Reverberation.

4.4.3 Order of Operations

The order that the tracks were presented in also seemed to influence the style of mixing, with most actions most likely to occur on a neighbouring track. This indicates most engineers operate left to right across the mixer before returning to the start, and the fine-tuning as they see fit. This was true for large and small sized sessions. When operating by instruments, it shows that when an action occurs on a track, the next action is likely to occur on a track of the same instrument type. This is could be explained by confirmation bias as all the sessions were already laid out by instrument type, so it is unclear if this is a result of that action. When looking at just the action type, there are some strong positive correlations. An action to create a track often led to the next action being to change the output of another track. This is expected since creating a track would often be to create a group bus, and changing the output target of a track to this newly created bus would be the completion of creating a new group. Panning and volume were not as strongly interlinked, but panning, volume and transport all had stronger relationships with each other than most others.

4.4.4 Mixing Commonalities

When exploring if the engineers had any commonality in the final mix produced, the engineers were split into two different groups. Each group was given a different initial gain and pan structure for the mix. After the analysis, it was clear that in *Queens Light*, which had 5 sessions in each group, that there was a strong clustering between the groups mix space. This indicates the starting position did have a significant role to play in the outcome of the mix. The spectral content also showed engineers would often follow a similar trend based off the initial information presented to the engineer. Some universal concepts do come through, with vocal tracks and bass tracks being boosted significantly above the rest of the mix. The bass instrument boosting was not something previously uncovered, but due to the lack of processing available to the engineers, the bass tracks were gained stronger to ensure the bass content was present. Figure 4.43 shows the spectrograms of all

the mixes and follows the curve created by Pestana et al. (2013). When producing the mix one of the most important items that was minimised was masking, with the engineer using decisions which ultimately led to mixes having lower masking properties than the original mix. These also correlated well with the scores given by the listening study.

From this study two main aspects of mix engineering stand out as the most important aspects of a mix: grouping and the reduction of spectral masking. The grouping analysis showed that, whilst it did not guarantee a good mix it was less likely to lead to a poor mix. This shows that grouping is an important concept for engineers to use although many do not. The time taken to set up a group is not always trivial with many actions required to do it. For this reason an automated solution would be beneficial to engineers as a form of automated set up stage. Masking minimisation, and the initial balance mix, shows that engineers are influenced by their first impressions of the track. Therefore a system which would improve this first impression by creating an already masked-minimised mix would also improve the mixing quality of engineers. As seen the balance mix process is time and effort intensive, removing or reducing this load will allow the engineer to focus on their creative aspects.

Chapter 5

Automatic Track Grouping using Linked Meta-data

Audio production is a complex task where engineers need to undertake a large number of actions to achieve an end mix, as shown by the study conducted in Chapter 4 and the subsequent analysis. In its simplest form, mixing requires bringing together multiple sources, then combining them into a single audio stream for playback elsewhere. Busses and groups allow for the creation of sub-mixes; a mix which contains a subset of the overall collection of sources. Whilst groups and busses themselves do not perform any difference in processing they allow for improved control over the mix, since a whole section of the mix can be directly controlled in one action. It has been shown in previous works that engineers using groups to manage the session end up with perceptually 'better' mixes compared to ones that do not use groups (Ronan et al., 2015b). In Chapter 4 it has been shown that not only is this true during the balance mix stage, but it also shows that not all engineers set up groups in a mix, even though this is recommended by texts (Eargle, 2002, p. 328).

5.1 Background

In the most simplistic form, a mix could be described as a direct summation of sources. To add some control, each source x_n has a gain control g_n to allow a basic level control. If we assume this is our entire mixing surface, then any mixing surface can be mathematically described in equation 5.1 (Terrell et al., 2014). This equation defines the general mixing process in its most basic form, where N inputs are scaled by a vector, g, and summed together. For small mixes with a handful of tracks, this could be manageable for an engineer to work with. With mixes containing potentially dozens of tracks, this could be very difficult to understand and work with.

$$y(t) = \sum_{n=0}^{N} \left(g_n x_n \left(t \right) \right) \tag{5.1}$$

Instead of working with each track individually, an engineer can use sub-mixing to increase the capacity of their control surface (Izhaki, 2012, p. 129). A sub-mix is defined as a mixed group of tracks, whereby each track in the sub-mix is exclusive to that group. This means each source in a session can be routed to either a sub-mix or to the global mix and cannot be a member of multiple sub-mixes. On a traditional mixing console, these would be denoted as multiple output points, with the master bus also listed (Izhaki, 2012, p. 130). Sub-mixes are then combined hierarchically, and output to either a parent sub-mix, or the overall global mix. Using this definition, the equation 5.1 can be updated to include sub mixes.

$$y(t) = \sum_{m=0}^{M} \left[\hat{g}_m \sum_{n=0}^{N} \left(k_{n,m} g_n x_n \left(t \right) \right) \right] \tag{5.2}$$

Equation 5.2 is the general mixing equation for sub-mixing, where N incoming sources x_n are split into M separate sub-mixes. The sub-mixes are then combined, using their own set of gain coefficients in $\hat{g}_m$ to create the final mix. The sources also use another vector $k_{n,m}$ which indicates whether that source is part of the m-th sub-mix. If the track is part of that sub-mix, then $k_{n,m} = 1$, otherwise $k_{n,m} = 0$. Since this is only using linear mathematics, the total system gain on a source in sub-mix m is equal to $\hat{g}_m g_n$. This generalised equation only takes into account one-layer of grouping. It is possible to create sub-mixes of sub-mixes too, adding extra layers of control, but complicating the signal flow.

A source is exclusively mixed into a sub-mix, therefore the sub-mix is normally called a 'group' (Eargle, 2002). This is because the effect is some sources are grouped with other sources. An engineer will group tracks based on several decisions, but could group for any of the following reasons

- Instrument Type
- Spatial Placement
- Temporal Placement
- Spectral Information

The instrument type is one of the most common, because of how grouping can be used to partition the control surface. In music production, it is common to have multiple sources of the same instrument type, or even on the same instrument. A piano, for example, may have different microphones to record the high and low keys. This is mostly because of the size of the piano; it would be difficult to place a single microphone to evenly capture all the keys. Other instruments may have multiple sources for stylistic reasons, such as an electric guitar which could have a Direct Injection capture and a microphone on the cabinet. Finally, there may be multiple guitars in the same song which require similar processing, or just to provide the ability to turn all the guitars up or down in a mix with a single control. Without grouping, each guitar would have to be turned up and down by the same amount. A task made harder when each guitar could be made up of several sources.

Whilst the total number of tracks in our mix size has increased to $N + M$, each group is a smaller subset of the overall mix, making it simpler to manage. Each group also now has a common parameter space, allowing processing and control to affect the entire group at once. What would have previously been many tasks to

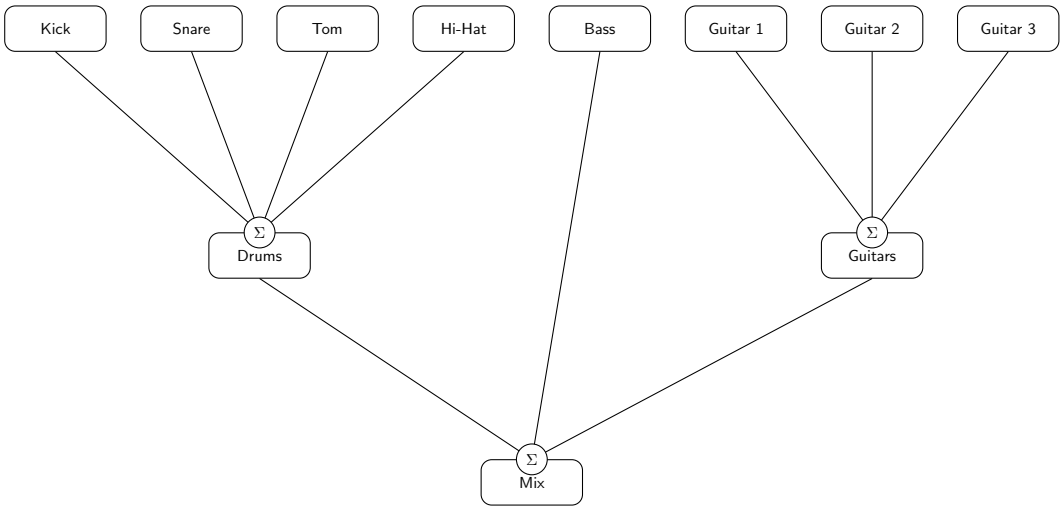


Figure 5.1: An example of grouping in a session.

reduce one multi-source instrument can now be done in one interaction. Likewise, processing all the tracks at once becomes simpler, as a processor could be placed onto the group rather than each individual track. Whilst this does change the response when using non-linear functions, oftentimes these are desired. A common example is compressing drums where each drum is normally recorded using a dedicated microphone, but the engineer would process them as a common instrument. Placing all the drums together into a group is therefore a very useful action for engineers.

The cost to an engineer for all of these benefits is setting up the group, which takes time. Adding a group requires creating the group tracks and evaluating which sources to place in which groups. On a session with a handful of tracks this seems trivial, but on a session with hundreds this becomes a significantly time consuming task.

5.1.1 Previous Works

Ronan et al. (2015b) performed a study to investigate how engineers group a given session, and then how this affected the perceptual quality of their final mixes. The study was conducted using junior engineers studying sound production as part of their university assessments. This allowed the researchers to have access to several versions of the same song, mixed to the same brief. Their study showed, given their source type being primarily rock songs, that drums were a prominent track type but not all students would create a group to control them. Likewise the study also showed that all of the grouping decisions were done using the instrument label alone. Several hierarchical groups were also made (groups of groups), which provided more expansive control surfaces to the mix engineer, such as the balance between instrumentation and vocals. Figure 5.2 shows the mixing structures created for one song by three different engineers.

Once the researchers had the mix, they performed a subjective listening test to assign a rank-based score. Using the results from the study, they evaluated the scores against the mixing practices used by each participant. It was demonstrated that there was a strong positive correlation between engineers using groups as a means of

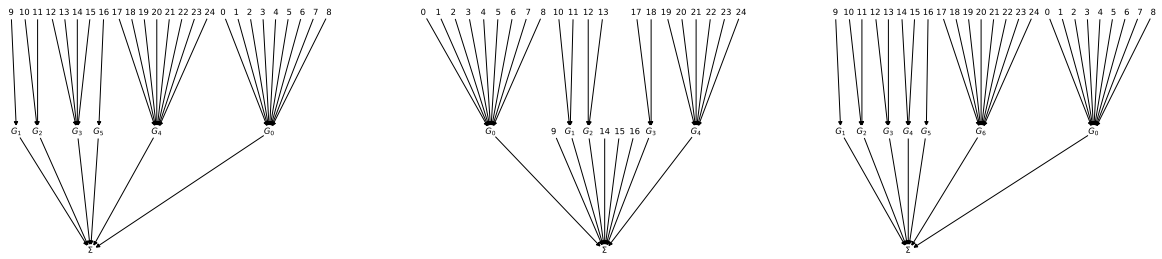


Figure 5.2: Three session grouping structures created for the song 'In The Mean Time' for the study performed by Ronan et al. (2015a). This shows the range of structures used by trained engineers is varied.

control, and the perceived quality of their mix. This was even more prominent when the respective groups were used to perform further processing such as equalisation or dynamic range compression.

This is aligned with the study conducted in Chapter 4, which shows that engineers using a bus configured for grouping, generally receive a positive overall mixing score. This study showed that, even with minimal control options, groups still lead to a better subjective rank, even in the balance mixing phase. This confirms the findings of the study conducted by Ronan et al. (2015b), and emphasises the importance of using groupings.

There has been little development in the area of automatic grouping of musical instruments. The only prominent work is by Ronan et al. (2015a). This study grouped tracks through their features rather than meta-data, despite their previous study (Ronan et al., 2015b) showing most engineers would group by the track instrument. They used data from a mixing competition, which gave them access to different mixing structures used by real-world engineers (De Man and Reiss, 2017). The similarity of the sources was calculated by extracting a set of features from each track, and then the data was fed to a Random Forest Classifier. Once this similarity had been discovered, the tracks could be grouped through a distance metric based on the classification tree. With distances between each track discovered, a hierarchical clustering could take place to group the tracks together based on these features. This method works well for musical instruments, as similar instrument types will share certain characteristics. For example, even without any musical knowledge, it would be apparent that a Floor Tom and Kick Drum share more timbrel similarities with each other than with a Violin.

This method is useful when used in unknown environments with no meta-data present or it is costly to add. Obtaining this meta-data is a less arduous process since several items could be inferred from real-world analysis. For example, most engineers would label a 'Kick Drum' instrument as 'Kick' or 'Bass Drum'. Likewise, asking for the instrument label could be exposed in a way for engineers to pass that information on.

5.2 Automatic Group Creation and Labelling using Web Ontology's

The aim of this system is to recommend a suitable routing structure from a given set of audio tracks. This system should be able to operate with minimal user input and correction. Previous works attempted to use audio features to dictate which groups are suitable (Ronan et al., 2015a). This assumption limits the potential

applications of the field by requiring a current audio stream to be playing through the system, or audio to be loaded.

In most digital audio workstations, each track can be tagged with meta-data to help the engineer quickly identify the track being used. These are usually for aesthetics, layout and project management roles rather than anything else. A DAW might, for example, place a suitable icon next to the track to help the engineer quickly navigate a complex session. This meta-data provides all the reasonable information required for instrument grouping, since similar instruments have some relationship between each other. For example, an 'Acoustic Guitar' could be said to be a part of a greater class of instruments called 'Guitars', which also encompasses 'Electric Guitar' and 'Bass Guitar'. Therefore grouping these instruments together would be expected. By looking at the meta-data it would be possible to suggest suitable connections between common classes or inferred properties of the class.

5.2.1 Instrument Relationship

The first challenge is to find a way to quantify the similarity between two instrument labels. For example, how similar are the instruments 'Guitar', 'Piano' and 'Violin' to each other? This question requires a method to extract the information so that grouping through clustering can be performed.

Instead, relying upon the instrument label itself requires a method of exploring the similarities. A data source representing the data must be found which encompasses a wide range of instrument information with relationship information to other instruments and instrument classes. Several data sets, in the form of ontology's, do exist to use Kolozali et al. (2011). But each contain bias, omissions or inaccuracies depending on the creator and their intention for the data set. For example, a taxonomy developed by Doktorski (n.d) attempted to group each instrument into a set of distinct classes. A violin is part of the 'Bowed', which in turn is part of the 'Chordophones'. A guitar is part of the 'Plucked', which is also part of the 'Chordophones'. The problem here is the relationship is incorrect, since the violin could also be part of the 'Plucked' group. Because the relationship tree is flattened, each instrument must be part of a single classifying parent, making this very inflexible. Furthermore, this method of instrument classification, does not typically represent the decisions made by audio engineers whilst mixing.

Instead of using expert created data sets which have errors or omissions in their structures, a crowd-sourced data set is instead used. The open project Wikipedia provides a rich Encyclopedia of information, maintained by public volunteers. Each page not only contains information in text or media formats, but also a network of connections to other pages through subjects and categories. These provide categories which can be used to navigate to other instrument pages allowing a rich structure to be built up by traversing through the subjects and categories. This gives a knowledge system based on subjects (entries or pages) and their relationships (hyperlinks between entries). Another advantage is the system is available through a SPARQL end-point through DBPedia. This is a part of Wikipedia and allows for the pages to be queried using SPARQL, a language for exploring Web Ontology's.

Ontology's provide a useful structure to hold how different pieces of data are related. They hold the information in the form of a store which contains itself plus the relationship to other entries. This can be represented using graphs, which allow pieces of information to be placed with a structure for their relationships. Graph theory is a method of modelling related data sources, where the information points (or vertices) are connected together by edges. An easy to grasp example are Encyclopedia entries. A single entry is a vertex in the graph, but that entry is connected to other entries through references (the edges). For example, an entry on Pompeii would be connected to Mount Vesuvius, Naples and the Roman Empire. By using a graph to represent the instrument data it would be possible to calculate the relationship between two data points by examining the graph. The links between the pages, subjects and categories form the edges of the system, allowing a complex graph to be built up to represent the structure. Graphs allow complex relational structures to be manipulated with ease, finding similarities between otherwise disparate information points. In notation a graph is often defined as $G = (V, E, \mu, v)$(Bunke and Shearer, 1998). V is a set of finite vertices, E is a set of finite edges which is often a matrix of size $E \subseteqq V \times V$, μ is the labels assigned to the vertices V and v the labels of the edges assigned to E. Throughout this section these definitions shall be used.

In a DAW, each track can be assigned a name or label, but often extra meta-data such as the instrument. These can be a subset field or free-text depending on the software and will provide a suitable hint as to the instrument expected on the track. The first stage of the graph building adds all the unique instrument vertices on to the graph G as a set of v_{inst}. These can be mapped directly onto a Wikipedia article, through their search system which will select the most appropriate pages. For example, 'Acoustic Guitar' is Acoustic_guitar. One each instrument page has been identified, the instrument is added onto the graph. The page can then be queried using DBPedia, using the Simple Knowledge Organisation System (skos)[1] *broader* tags, to give the Categories on the given page. For example, the page 'Flute' has the categories 'Flutes', 'Woodwind instruments', 'Jazz instruments', 'Classical music instruments' and 'Orchestral instruments'. Each of these are then added onto the graph as well as the edge connecting them. Then a graph can be made going backwards through the subjects and categories, collecting pages in each of those categories. Quickly this traversal of the graph can give a very dense graph with hundreds of connections to broader subjects. The page for *Piano* is linked with 310 subjects through 432 connections after just 4 levels. By adding multiple instrument tags to start with, it is quickly apparent that an interconnected graph can be uncovered. Figure 5.3 gives an example of the complexity of the graphs at only four searches deep.

Not all instruments are highly connected. Tables 5.1 and 5.2 give the density of the graphs for the instrument 'Violin' and 'Conga' respectively. The 'Violin' has 390 subjects related to it with 652 edges between those subjects just scanning 4 levels deep. Compared to the Conga, which has 214 subjects with 317 edges, it is clear the density of the 'Violin' graph is far higher, where there are more relationship connections between the vertices. Therefore it is important to go deep enough in the graph to capture enough information. At a depth of 8, most instruments converge on a similar graph shape.

[1] `https://www.w3.org/2004/02/skos/`

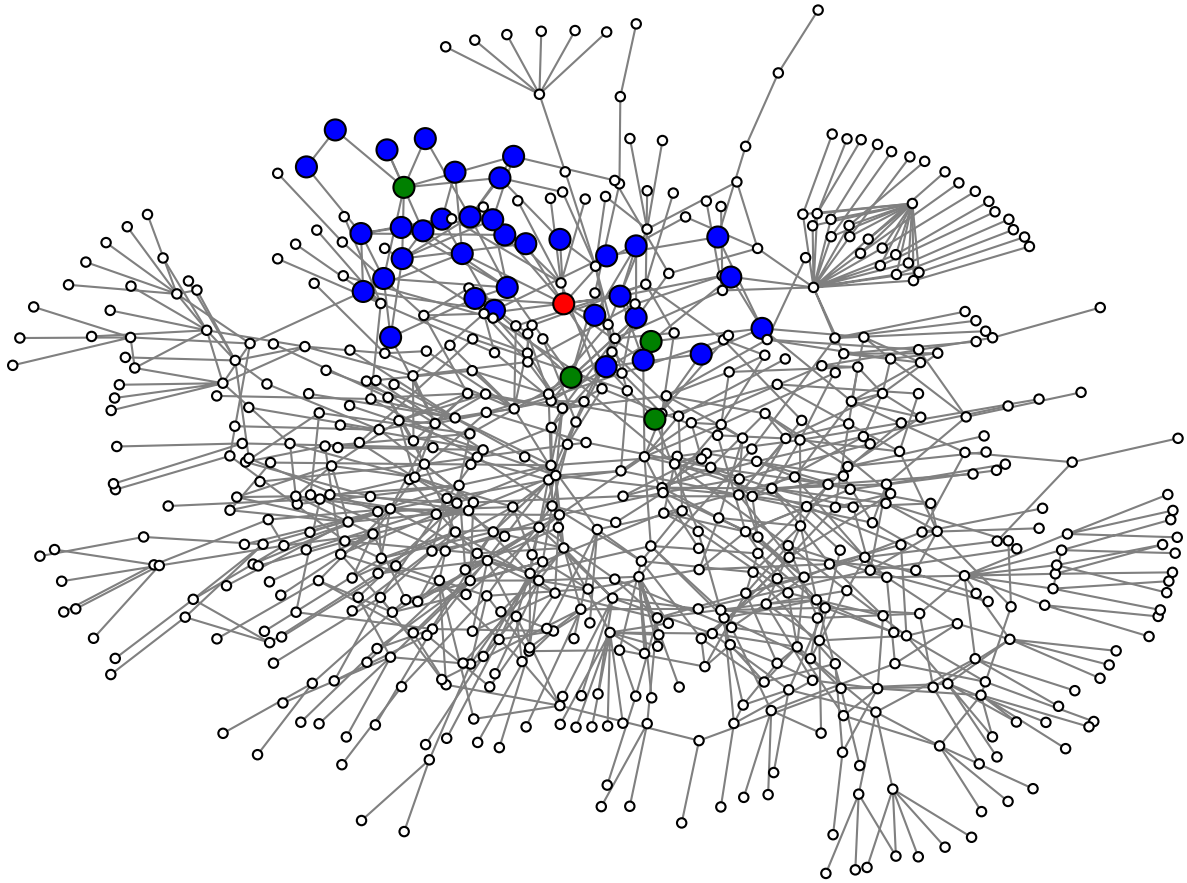


Figure 5.3: The full graph for four instruments: *Acoustic Guitar*, *Electric Guitar*, *Piano* and *Snare Drum*. This gives every possible subject which contains these four instruments to a depth of 4, showing the complexity of linked data stores. Cutting the graph gives a focused scope. The root *Musical Instruments* is red, the source instruments are green and the vertices which link back to *Musical Instruments* in blue.

Depth	Number of Vertices	Number of Edges	Density
1	68	80	1.176
2	136	174	1.279
3	248	355	1.431
4	390	652	1.672
5	567	1036	1.827
6	826	1518	1.838
7	1149	2266	1.972

Table 5.1: The number of vertices and number of edges for the graph after scanning from the root *Violin* instrument. The number of vertices grows nearly exponentially, but the edge density decreases

Depth	Number of Vertices	Number of Edges	Density
1	30	35	1.168
2	58	77	1.328
3	116	159	1.371
4	214	317	1.481
5	379	610	1.609
6	619	1049	1.695
7	1013	1765	1.742

Table 5.2: The number of vertices and number of edges for the graph after scanning from the root *Conga* instrument. There are less vertices in the earlier scans than the Violin but by the 7th level it has a high degree of connectivity.

Not all of these connections are useful for instrument grouping, since they either go out of scope, such as references to construction or culture, or they end up in enclaves of the graph which are disconnected from other entries. So to help focus the graph a root subject is identified as well. This root subject provides an anchor for the graph building process. If a vertex cannot connect to this root it should be removed from the graph since it has no relationship to our chosen subject. For this purpose, the subject root, labelled as v_0, is 'Musical Instruments'. This root is suitable because it provides an entry point through very broad relationships to any instrument-based page on Wikipedia, whilst allowing for detours which would otherwise be overlooked. The power of the root vertex selection also makes this system very adaptable to other name spaces of grouping. For example, an automated grouping system for theatrical performances could be done by selecting v_0 to be 'Theatre'. The system would still behave as expected provided the track meta-data is theatrical data.

With the root vertex v_0 and the instrument vertices v_{inst} known a new graph is extracted from G, called G_1, where $G_1 \subset G$ such that $G_1 = (V_1 \subset V, E_1 \subset E)$. A vertex v_j exists in the new G_1 if a compound edge $\{v_j, v_0\}$ exists in G. A compound edge is an edge which may not be direct between the vertices, but by traversing several vertices there may be a connection. This is done by extracting only simple paths from the graph G that connect between the vertices in v_{inst} and v_0. A simple path is a collection of graphs which form a path between the two given vertices, without repeating an edge or vertex. There may be multiple simple paths between each vertex given. All of the vertices in each of the simple paths, with their edges, are put inside G_1. Figure 5.3 gives a graphical overview of how the pruning process removes all irrelevant vertices from the graph structure. The red vertex is the root 'Musical Instruments' vertex and the green the four given source vertices. Every vertex highlighted blue is a vertex which would be kept. It is clear how many vertices are removed by this search pathway. Now a graph is formed, the instrument similarity part of the process can begin to determine how similar two items are.

5.2.2 Vertex Similarity Measures

Finding the similarity between two vertices in a graph can be achieved through a few methods. This section will explain them before moving on to the Instrument Similarity specifically.

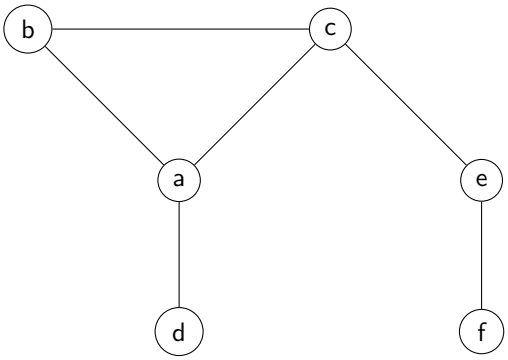


Figure 5.4: An example graph with vertices and edges. This is not fully connected, and shows a structure to the relationships between the vertices.

Node	a	b	c	d	e	f
a	0	1	1	1	0	0
b	1	0	1	0	0	0
c	1	1	0	0	1	0
d	1	0	0	0	0	0
e	0	0	1	0	0	1
f	0	0	0	0	1	0

Table 5.3: The graph shown in Figure 5.4 represented as a matrix, with a 0 indicating no connection and 1 an edge.

Node	a	b	c	d	e	f
a	0	1	1	1	2	3
b	1	0	1	2	2	3
c	1	1	0	2	1	2
d	1	2	2	0	3	4
e	2	2	1	3	0	1
f	3	3	2	4	1	0

Table 5.4: The minimum number of hops required to get to each vertex in the graph presented in Figure 5.4.

A graph not only provides the relationship information, but also the ability to find common ancestors between two or more entries. In Figure 5.4, vertices ‘a’ and ‘c’ are connected to ‘b’ and vertex ‘d’ is connected to ‘a’. Intuitively, vertices which are strongly related to each other will be directly connected. Conversely less related data may be linked but only by traversing through other vertices. The relationship of the data can be calculated by counting how many 'hops' it takes to get between two vertices. If they are strongly related, they would be connected directly or by few hops. Table 5.4 shows the number of hops for the graph in Figure 5.4. As can be seen, there is an identity of 0 when connecting to itself, but vertices ‘a’, ‘b’ and ‘c’ form a cluster together as they are all strongly connected. Whilst the vertices further out (‘d’ and ‘f’) have further jumps to make it to this core group. This can be further expanded upon through weighted graphs. In a weighted graph, each edge has a weight applied to it so not all edges are equal. This graph is more powerful, because it penalises hops which go through a weaker edge. This gives greater flexibility to the model, but increases the complexity.

The hop provides a crude distance measurement, although it is unbounded and would require tuning between different graph sizes. For example, in the demonstrated example the highest distance would be 4, to go from ‘d’ to ‘f’. If the graph had a thousand vertices, a distance of 4 would suggest it is probably quite closely related. Whilst this distance relationship between two vertices is useful, it also does not provide a metric for how similar the two vertices are. The similarity between vertices can be calculated by comparing the features of the vertices. One form of vertex feature is cosine similarity (Newman, 2018, p. 212). This metric uses the dot product between two vectors as its base, which is written as the combined magnitude and angle to represent the new vector. The angle can be found using equation 5.3.

$$\cos\theta = \frac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}|\,|\mathbf{y}|} \tag{5.3}$$

As shown in Table 5.3 the graph can be represented as a matrix. Two vectors of the matrix $\mathbf{x}$ and $\mathbf{y}$, represent the edges for a given vertex. Since the vectors represent the edges of the i-th and j-th rows of the graph A, equation 5.3 can then be rewritten into equation 5.4 (Newman, 2018, p. 213).

$$\sigma_{ij} = \cos\theta = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{\sum_k A_{ik}^2}\sqrt{\sum_k A_{kj}^2}} \tag{5.4}$$

Where k is the index of the element in the row, such that A_{ik} represents the index of matrix A at row i column k. By iterating over it is possible to show the distnace between the two rows A_i and A_j. If the network is unweighted then row A_i will only have 0's and 1's to represent whether an edge exists or not between two vertices. In a weighted network this would not be true as the weights would be represented instead. An unweighted network can be seen as a network with all weights set to one. In this case, where only 0 and 1 are present, then the following condition will be met.

$$\sqrt{\sum_k A_{ik}^2} = \sum_k A_{ik} \tag{5.5}$$

This is because the squared value of 0 and 1 is still 0 and 1, so squaring them would be the same as if no squaring had occurred. A further simplification can be achieved by knowing that $\sum_k A_{ik}$ will return the sum of the row A_i from the matrix. But since the row is only 0's and 1's, the sum actually measures the degree of the vertex, as in how many edges the vertex has. So $\sum_k A_{ik} = k_i$ where k_i is the degree of the vertex. Finally, the sum of the multiplication between the two rows $\sum_k A_{ik}A_{kj}$ will return the number of vertices that both share in common.

$$\sigma_{ij} = \cos\theta = \frac{n_{ij}}{\sqrt{k_i k_j}} \tag{5.6}$$

Another method for measuring the similarity is to use the Jaccard similarity coefficient, shown in equation 5.7 (Schaeffer, 2007). The theory is that vertices which have very similar neighbourhoods will have a high Jaccard similarity coefficient, indicating they themselves are very similar to each other. Conversely, vertices which are not related will have fewer common vertices in their neighbourhoods, giving a very low Jaccard similarity coefficient. Whilst similar to the cosine similarity, this examines if the vertices connect to the same neighbourhood of vertices rather than the number of vertices in the common neighbourhoods. The coefficient works by analysing the ratio of the number of common neighbourhoods of the two vertices, v and u, against the total number of neighbourhood vertices. The neighbourhood of a vertex is defined as $\Gamma(v)$, and comprises of the vertices that vertex v has a direct connection to. The Jaccard similarity details the ratio of vertices that two vertices share $|\Gamma(v) \cap \Gamma(u)|$ against the total number of vertices two vertices can connect to.

$$w(v,u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|} \tag{5.7}$$

Table 5.5 shows the calculated Jaccard similarity coefficients for each vertex in Figure 5.4. It confirms that the core group of inter-connected vertices 'a', 'b' and 'c' have very strong similarity. This gives them all string Jaccard similarity, whilst the less related vertices 'e', 'd' and 'f' have very poor similarity with the other vertices, even if they are directly connected. For it's ease of use as a direct similarity score which will always be bounded between 0 and 1, the Jaccard similarity is used.

5.2.3 Instrument Similarity

The Jaccard similarity coefficient (Schaeffer, 2007), determines how related two vertices are by analysing the overlap of the two neighbourhoods. However, the pruned graph G_1 is directional, meaning the neighbourhoods

Node	a	b	c	d	e	f
a	1	0.75	0.6	0.25	0.25	0
b	0.75	1	0.75	0.5	0.5	0
c	0.6	0.75	1	0.5	0.25	0.33
d	0.25	0.5	0.5	1	0	0
e	0.25	0.5	0.25	0	1	0.66
f	0	0	0.33	0	0.66	1

Table 5.5: The Jaccard similarity coefficient as calculated for each vertex in Figure 5.4

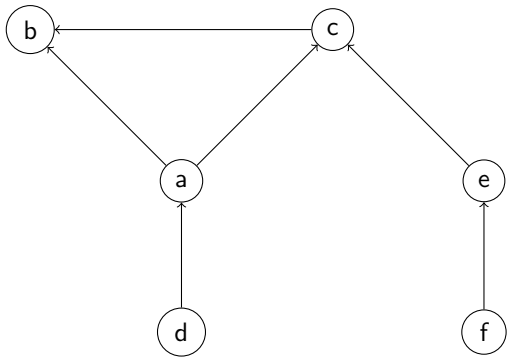


Figure 5.5: The same graph as Figure 5.4 but with directional weightings

Node	a	b	c	d	e	f
a	1	0.33	0.5	0.25	0.25	0
b	0.33	1	0.5	0	0	0
c	0.5	0.5	1	0	0.5	0
d	0.25	0	0	1	0	0
e	0.25	0	0.5	0	1	0.33
f	0	0	0	0	0.33	1

Table 5.6: The Jaccard similarity coefficient as calculated for each vertex in the directional graph in Figure 5.5.

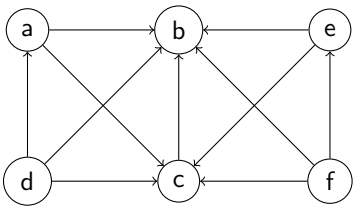


Figure 5.6: Extra edges are added to improve the neighbourhood selection of the directional graph in Figure 5.5. The layout was changed to improve intelligibility.

Node	a	b	c	d	e	f
a	1	0.33	0.5	0.25	0.25	0
b	0.33	1	0.5	0	0	0
c	0.5	0.5	1	0	0	0
d	0.25	0	0	1	0	0
e	0.25	0	0	0	1	0
f	0	0	0	0	0	1

Table 5.7: The Jaccard similarity coefficient as calculated for each vertex in the directional graph in Figure 5.6

of any two given vertices can be extremely limited to only the parent subjects of that entry. The graph G_1 is also poorly connected where each vertex only has a few edges each, so the neighbourhoods are constrained even further. Therefore the overlapped neighbourhood of two given instrument vertices can be poor, or even zero if the two instruments are not immediately related. To emphasise the problem, table 5.6 shows the new Jaccard similarity coefficient calculations for the graph in Figure 5.5, which is now directional. Because the edges can only be traversed one way, the neighbourhoods of each vertex is reduced, to the point most vertices no longer overlap.

To help improve the similarity score the neighbourhoods need to be improved. Instead of the neighbourhood being defined as the vertices directly connected, the neighbourhood should be made of every subject vertex it has a path to in G_1. This flattened graph, called G_2, has the same vertices as G_1, $V_2 = V_1$, minus the root vertex v_0 since that is common to all the source vertices. Now the edge set E_2 can be defined as $\{v_{inst}, v_j\} \in E_2$ if a compound path exists between v_{inst} and v_j in G_1. This effectively states if a path between v_{inst}, representing an instrument source, and vertex V_j representing a subject, even if it is through multiple hops, should be directly connected.

This flattening therefore means instruments which have a similar set of parent subjects will now have a similar neighbourhood, resulting in a higher Jaccard similarity score as this solves the problem of having multiple vertices with no overlapping neighbourhoods, since all vertices in G_1 should have a connection to the root vertex v_0. However this now also proposes a different problem which is the neighbourhoods are all very strongly overlapped. This can happen with multiple highly related instruments. To improve the accuracy of the relationship measurement the edges in E_2 are weighted by the number of hops taken by weighting as $\frac{1}{N}$, where

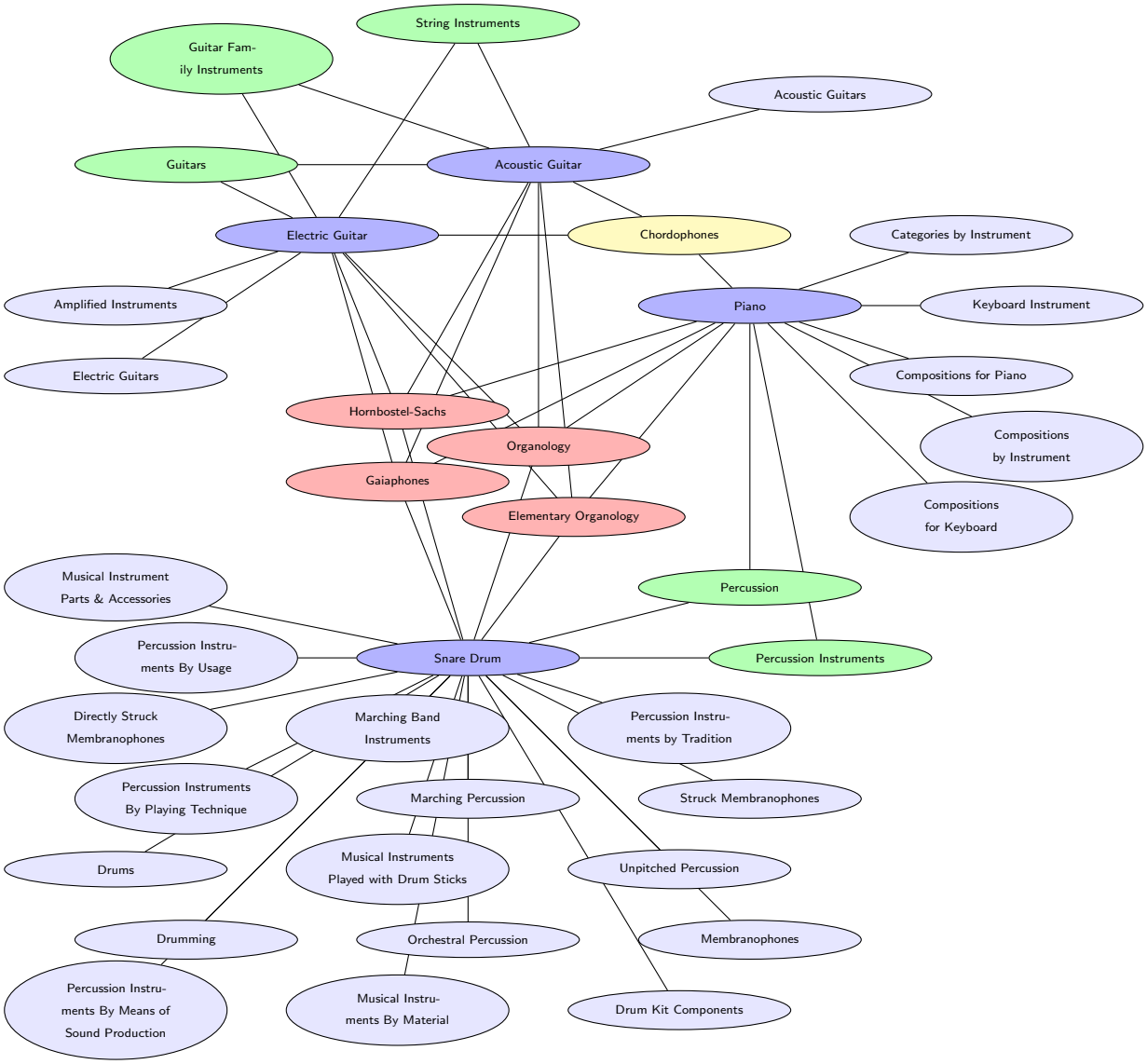


Figure 5.7: The flattened graph G_2 showing the relationships between each vertices. Tightly grouped subjects are nearer the centre, with solitary subjects on the outside, due to the drawing algorithm

N is the number of hops. So a directly connected vertices is given a weighting of 1, vertices an extra hop away a weighting of 0.5 and so on.

Figure 5.7 shows this flattened G_2 graph derived from the G_1 shown in Figure 5.3. The subjects *Organology* and *Gaiaphones* are universally common to the four instruments. These subjects are extremely broad and close to the root subject. The more specific subjects *Keyboard instruments* and *Amplified Instruments* are pushed outwards as they connect to one instrument only (*Piano* and *Electric Guitar* respectively).

With this flattened graph in place, the Jaccard similarity w is calculated for each instrument vertex v_{inst}. For example, following Figure 5.7, the *Electric Guitar* has *Acoustic Guitar* have a Jaccard similarity coefficient of $w = \frac{7}{11} = 0.636$ This similarity measure can be converted into a distance measure by $1 - w$ and stored as an N-by-N matrix D, where N is the number of instrument vertices. This distance matrix for the four instrument example using in figures 5.1 and 5.7 is given in Table 5.8.

	Acoustic Guitar	Electric Guitar	Piano	Snare Drum
Acoustic Guitar	0.000	0.294	0.750	0.875
Electric Guitar	0.294	0.000	0.783	0.886
Piano	0.750	0.783	0.000	0.793
Snare Drum.	0.875	0.886	0.793	0.000

Table 5.8: The distance matrix of the four instruments in figures 5.1 and 5.7.

The distance matrix is symmetric since $w(v_i, v_j) = w(v_j, v_i)$ and $w(v_i, v_i) = 1$. Using this, instruments are then grouped together using hierarchical clustering. The first stage is to extract the distance into a Cartesian space, which can then be sent through a clustering algorithm. This is performed using multi-dimensional scaling (MDS), which aims to convert a given distance matrix into a set of Cartesian co-ordinates, the distance between which reflects the original distance matrix. The number of components to compute N is always 2. This ensures the number of dimensions is suitably reduced for the clustering algorithm, since this is purely a space representation issue. MDS operates to try and find a set of vectors for each point which minimises the global Euclidean distance between the points, to ensure the representation best matches the distance matrix given. Instruments which have a high degree of similarity, have a low distance score, therefore their Euclidean distance should be minimal, since they should share similar relationships to the other instrument vertices. As can be seen in Table 5.8, the Acoustic Guitar and Electric Guitar have low distance scores, but both also have similar scores for the Piano (0.750 and 0.783 respectively) and Snare Drum (0.875 and 0.886 respectively). Therefore, the MDS algorithm will try to minimise their distance in the reduced Cartesian space, increasing the likelihood they would be clustered together. The output of the MDS is a 2-D array of vectors. Each array represents the distance from the neighbouring points.

From this distance, hierarchical clustering can be performed using a linkage criterion (Murtagh and Conreras, 2012). This determines the distance between the sets of observed data as a function of the distance between the observations themselves. An overview of hierarchical clustering is discussed in Section 4.3.4.

Once the hierarchical clustering has completed, a set of clusters are created linking each element together. A cluster flattening algorithm is used to reduce the number of clusters. Since each leaf of the hierarchical cluster holds at least two entries, there are many pairings of cluster to consider. By scanning up the tree at each distance point, a cophenetic distance can be found where an exact number of clusters is extracted. The first task then is to compute the number of clusters, and therefore instrument groups, to create. Equation 5.8 gives the basic algorithm for doing this, where N is the number of instruments.

$$k = \min\left(\lfloor\frac{\mathrm{N}}{2} - 1\rfloor, 1\right) \tag{5.8}$$

This equation tries to create at most 1 group for every one track. Whilst not always possible, depending on the relationship of the groups it is possible for a group cluster to have one instrument because it is completely unrelated to the rest. But by rounding it down it should mean it is forced to build groups from the suitable

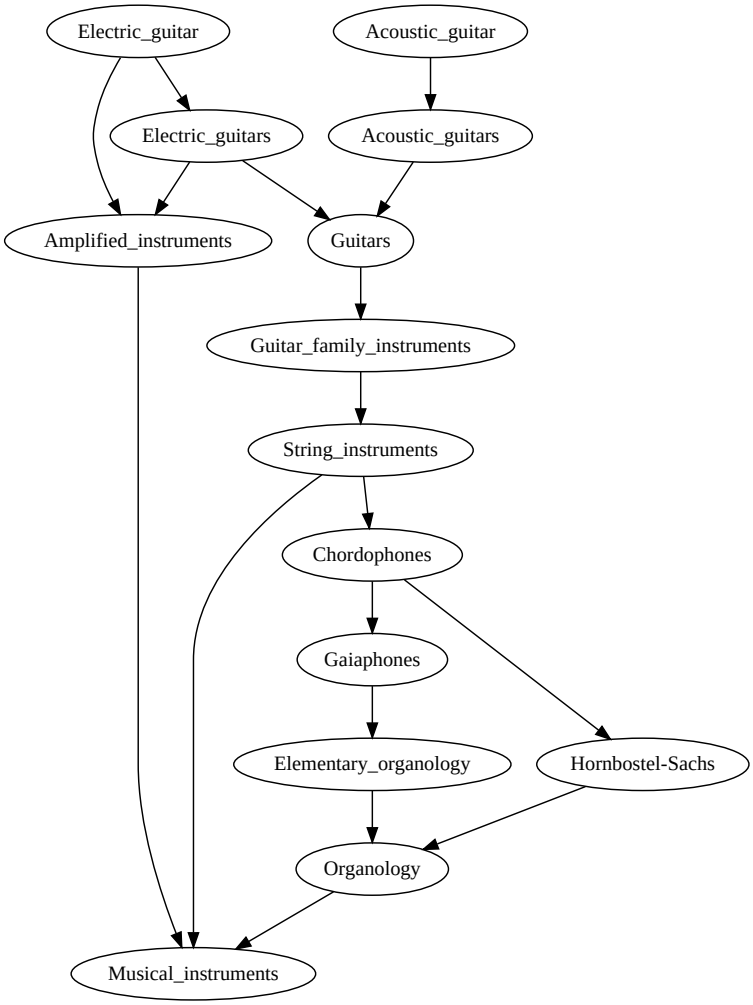


Figure 5.8: The cluster sub-graph G_3^k formed from a cluster holding *Acoustic Guitar* and *Electric Guitar*

.

tracks, but not so few groups that it will force instruments into unsuitable groups. Once the clustering has been performed, the instrument tags are added and output from the system, for example using the same four instruments as before, the system recommended two groups: $C_0 = \{$*Acoustic Guitar*, *Electric Guitar*$\}$ and $C_1 = \{$*Piano*, *Snare Drum*$\}$.

5.2.4 Naming the groups

In a typical production environment, each group is given a name to identify the contents and purpose of that group (Izhaki, 2012, pp. 154-155). Chapter 4 showed in Table 4.22 that the engineers do name based upon the tracks being sent to the group. For instance the engineers would place the Kick Drums, Snare, Toms and Drum Overheads into a group often called 'Drums' or 'Drum Kit'. Further sub-grouping of instruments did occur as well, with engineers placing the entire backing mix into one group called 'Instrumental' to balance against the 'Vocal' sub-group, or placing two 'Kick Drum' sources into a group of its own before routing the common 'Drum' group. The name selected by the automatic system should be representative of the tracks

placed in that group and so the engineer can quickly identify the group and understand its context with respect to the rest of the mix.

This made it obvious to use the already generated graph itself, as constructed in G_1, since it holds all the relationships between the instruments and other subjects. By identifying a common subject, a suitable name could be inferred to assign to the group. This method uses a well established graph search algorithm called Lowest Common Ancestor (Newman, 2018).

The graph G_1 holds the source subjects, root subject and the directly connected subjects between these two To find the common name of the k-th group, a new graph $G_3^k \subset G_1$ is made. This graph contains vertices so long as the subject v_j is in the path between an instrument in the k-th group vik and the root subject v_0. The edges are copied over from the original graph so long as the two vertices are present. Figure 5.8 shows this graph for a cluster containing *Acoustic Guitar* and *Electric Guitar*. As can be seen, the graph holds only the subject vertices which connect the two instruments to the root 'Musical Instruments' subject.

Since the groups are related to the instruments in question, the name should be closely related to that grouping. Therefore a subject which is close to the group is determined as the best choice. This subject is the nearest common parent from the instrument and should be the vertex which has the fewest number of hops to. The most suitable subject is determined to be the nearest common subject vertex, mathematically this can be defined by minimising the total number of hops required to reach the subject from each source subject. The algorithm is defined as follows: first, the distance between two given vertices needs to be established. The distance between two vertices is the number of vertices needed to traverse to reach the target from the source. This is defined as $\delta(v_i, v_j)$. If v_i does not have a path to v_j then $\delta = \infty$. The nearest common subject vertex for cluster C_k is defined as s_k and can be found using equation 5.9.

$$s_k = \arg\min_j \left[\sum_{v_{inst} \in C_k} \left(\delta \left(v_{inst}, v_j \right) \right) \right] \tag{5.9}$$

The vertex with the smallest total distance from every instrument vertex v_{inst} in cluster C_k is the nearest common vertex s_k. The group name is then given as the label attributed to this subject. In figure 5.8 this is the 'Guitars' subject and can be confirmed visually from the two instruments in the cluster *Acoustic Guitar* and *Electric Guitar*.

By following this approach for an entire mix, the system can build and recommend a set of groups for instrument labelled tracks. An output of a 14 track input is depicted in figure 5.9. Tracks with the same instrument have been grouped together into *Electric Guitars*, *Snare Drum*, *Drum Kit* and *Tom-tom drum*. This is called sub-grouping and is performed due to the direct relationship these tracks will have, given they have the same instrument tag. The unique list of instruments to group are then processed to identify the final layer of groups, giving two super-groups which then route to the master output.

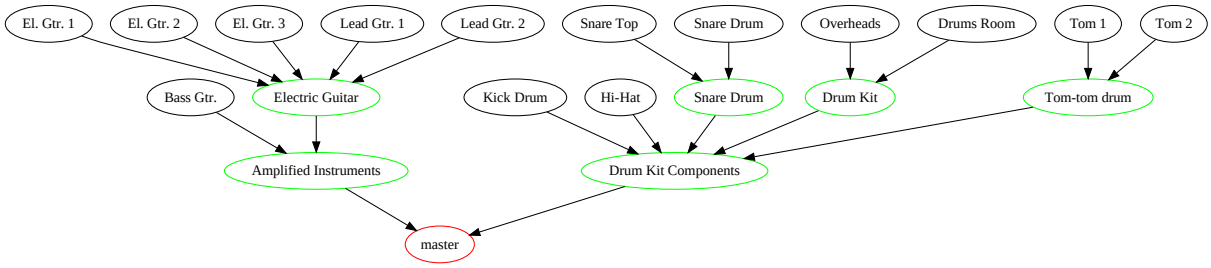


Figure 5.9: Complete output from a set of test tracks. 6 groups are recommended for the 14 tracks, judged only from their instrument labels.

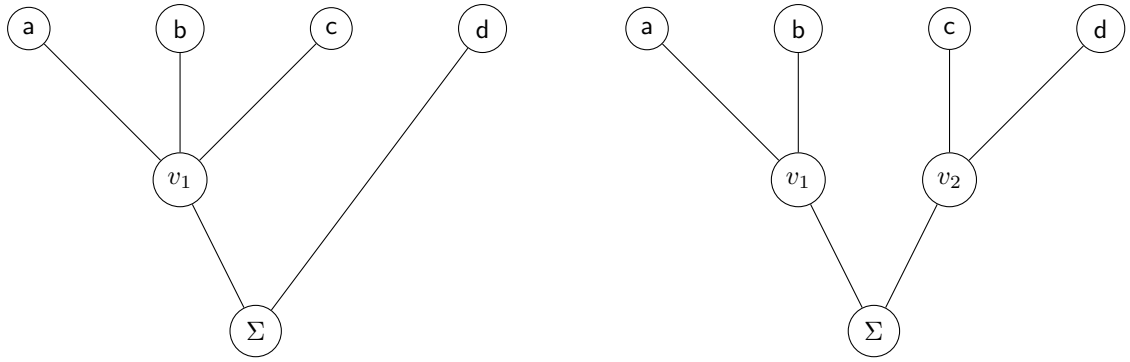


Figure 5.10: Two example graphs to compare, both have the same structure since they contain input tracks (a, b, c and d) and an output track Σ. Throughout this section the graph on the left will be compared to the graph on the right to show the suitability or problems with metrics.

5.3 Evaluation

To evaluate the grouping and naming systems, a data set of groups made by engineers is needed. From the study in Chapter 4 and the previous study by (Ronan et al., 2015a), a total of twelve multi-track mixes were collected across 81 sessions, spread across the mixes. From these, the structure of the multi-tracks can be effectively tested by comparing the generated groups and labels against the multi-tracks produced by the engineers. The details of the combined data set are shown in Table 5.9. The source in Chapter 4 is already contained within the PostgreSQL database and required a query to extract into a JSON format object to pass through the system. The data set by (Ronan et al., 2015a) is in the form of Pro Tools files, which could not be easily queried. These were manually converted into a JSON readable format before any actions were able to be undertaken.

Two parts of the system need to be evaluated to judge its performance as an automatic group creation and labelling tool. Specifically, scrutinise the ability for the system to pick suitable groups and the ability for the system to select a suitable name for a given group. These two evaluations, group creation and group naming are shown below and will cover the methodologies for the evaluation, along with the results.

Name	Number of Tracks	Number of Participants	Source
High Blood Pressure	24	8	Ronan et al. (2015a)
In The Mean Time	24	8	Ronan et al. (2015a)
Lead Me	22	8	Ronan et al. (2015a)
My Funny Valentine	18	8	Ronan et al. (2015a)
No Prize	14	8	Ronan et al. (2015a)
Not Alone	24	8	Ronan et al. (2015a)
Pouring Room	18	8	Ronan et al. (2015a)
Queen's Light	17	10	Jillings and Stables (2017d)
I'm Alright	12	6	Jillings and Stables (2017d)
Sleigh Ride	7	3	Jillings and Stables (2017d)
The English Actor	17	3	Jillings and Stables (2017d)
Left Blind	16	3	Jillings and Stables (2017d)

Table 5.9: Details of the twelve evaluation mixes used

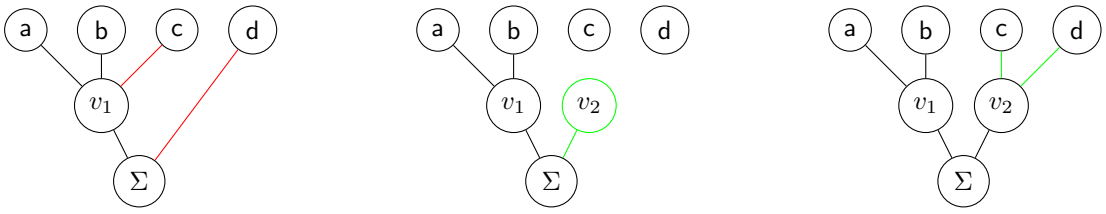


Figure 5.11: The graph edit distance visually demonstrated from Figure 5.10. The first step removes the edges c to v_1 and d to Σ. Then a new vertex is added v_2. Finally edges c to v_2 and d to v_2 are added, making the graph isomorphic.

5.3.1 Selection of groups

The construction of the routing table $k_{m,n}$ (equation 5.2) is the focus of the selection of the groups. This matrix holds the information as to which sub-group m track n should be routed to. As we have demonstrated in Table 5.3 and Figure 5.4 the graphs are just matrix entries underneath for the system to scan through. Therefore, comparison systems can be built to measure the similarity between two graphs.

The success of the system should be determined by the ability for it to select a grouping structure which is representative of what a real-world engineer would do. A method for analysing this lies in graph theory itself, namely graph similarity measures. If the shape of the automated graph is similar to the shape of the engineer graphs then this would provide affirmation that the system can create a set of graphs suitable to the engineer.

Graph Similarity Measures

One metric for graph similarity is the Graph Edit Distance (GED) (Sanfeliu and Fu, 1983). This measurement works on the principle of isomorphism. Two graphs are called isomorphic if they hold the same vertex and edge

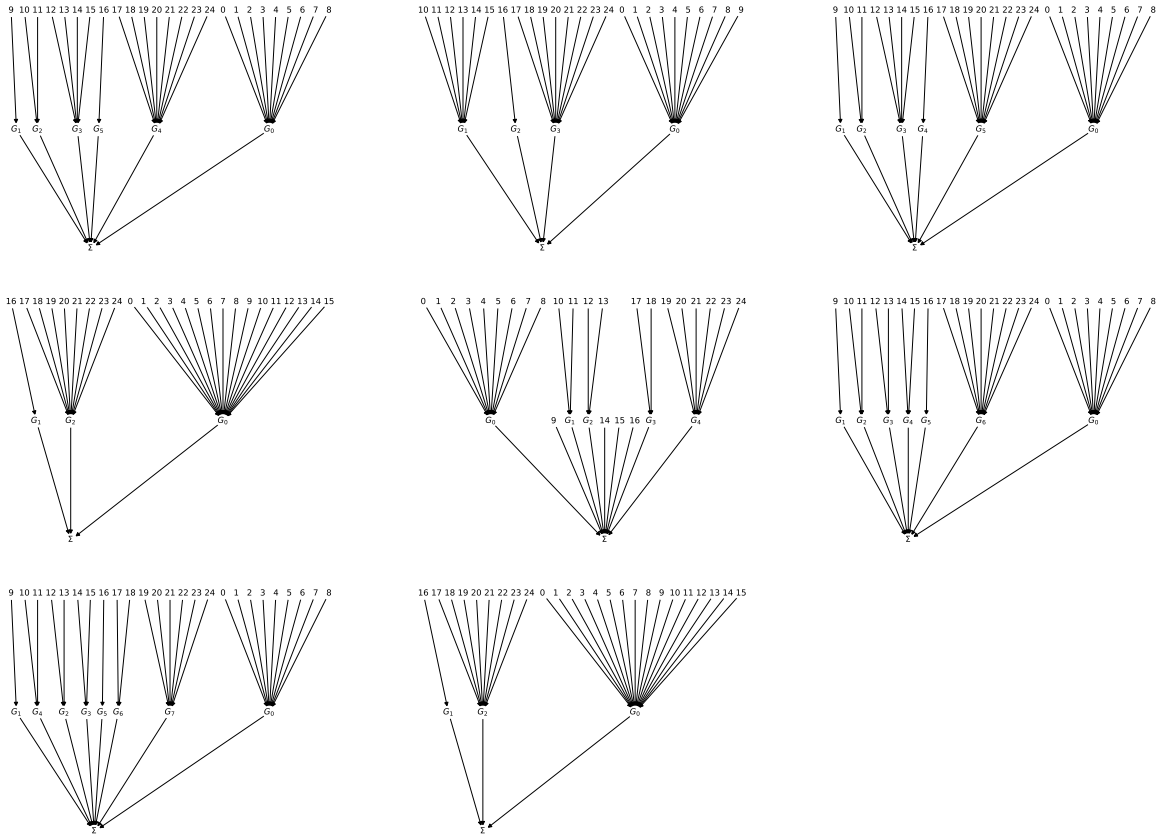


Figure 5.12: The full dataset of graphs from the study by Ronan et al. (2015a). These graphs are used to reference the evaluation methodologies in this section.

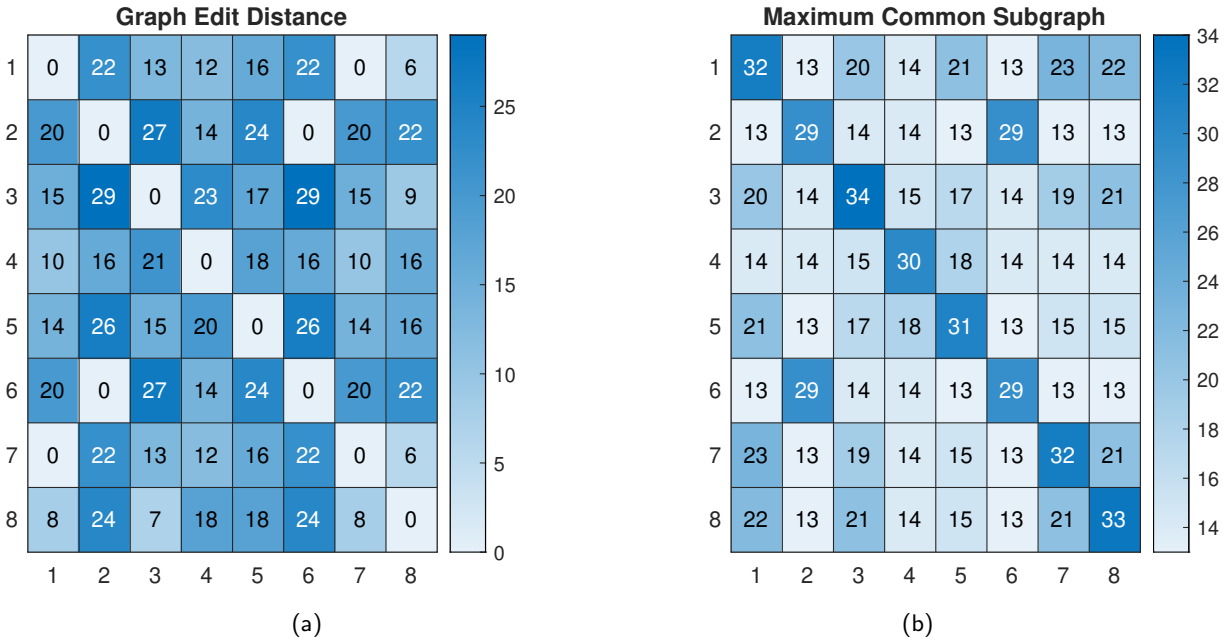


Figure 5.13: The graph edit distance (GED) and maximum common sub-graph (MCS) results between the existing graphs from the study by Ronan et al. (2015a) in Figure 5.12.

structures between them, such that the vertices from one graph can be mapped directly onto another graph. This idea that two graphs can be identified as isomorphic gives way to the graph edit distance. The system attempts to edit the graph, by adding or removing vertices and edges, until the graphs are isomorphic. The number of edits required gives the graph edit distance, with smaller numbers of edits scoring a lower distance metric. This can be tailored to the problem itself, with certain actions being forbidden or associated with a far higher cost depending on the requirements of the distance metric. Figure 5.11 gives a visual demonstration of the graph edit distance in action. The first step removes the edges c to v_1 and d to Σ. Then a new vertex is added v_2. Finally edges c to v_2 and d to v_2 are added, making the graph isomorphic.

The graph edit distances for the engineer created structures of the song In The Mean Time from the study by Ronan et al. (2015a) is given in Figure 5.13. The costs for any edit was set at 1.0, so the number reflects the total alterations made to the graph. There is a diagonal identity of 0 when compared against itself, which indicates the graph isomorphism is correctly detected.

The graph edit distance in Figure 5.13a has the highest amount of interpretation to the result given because it is directly linked to the size of the graphs. This means it will not be directly comparable with graphs of different sizes, in this case source track counts, since large track counts may need more edits to match than smaller track count sessions. For example, if two sessions both had a graph edit distance score of 10, but one session had 10 tracks and the other had 100, the larger session needed fewer edits relative to its size and therefore is of closer distance to the comparison graph. Therefore this should be normalised to edits per source vertex to allow for a more useful comparison.

The costs for the graph edit distance can also be updated to reflect the true cost to the engineer. In graph edit distance scores, there are 6 cost variables that need to be considered: creating, removing or modifying a vertex

Action	Cost Function
Create Vertex	15.3436
Remove Vertex	0
Rename Vertex	0
Create Edge	if$V_j = \Sigma \& V_i = G, 0$else$1000$
Remove Edge	1000
Modify Edge	2.668

Table 5.10: The costs for the graph edit distance

or edge. These six costs should reflect the true cost in the real world for adjusting these parameters. From the study in Chapter 4 the time penalty for these actions can be measured. By taking the time delta from the previous action occurring to the current action occurring, a time cost for performing the action can be estimated. Whilst there is a degree of noise, this method will provide a useful basis to calculate the costs. The actions are all timestamped, so the delta can be found by identifying the row which contains the action in question, then extracting the timestamp of that entry and the preceding action entry. Doing this for the two actions 'Create Track' and 'Change Output Track' will give the equivalent time cost to the engineer for the equivalent costs of creating a group and modifying an edge. Care needs to take place that creating a track will automatically add an edge to the master bus, so this cost should also be removed, since the environment automatically routes all new tracks to the master.

The mean time delta for 'Create Track' is 15.34 seconds and for 'Change Output Track' is 2.67 seconds. Therefore the costs will be set as in Table 5.10. The removal of a vertex costs 0 because if a group is not-used it can remain in the session completely disconnected, therefore this should not be penalised to make the graphs truly isomorphic. Likewise renaming a vertex in the graph should not be a cost because the group structure will be preserved if the input vertices are the same. Because at the fundamental point the only difference between the graphs will be the groups. For the edges, the "create" edge has the case for when the destination is the Master output and the source is a group with no edges connected, in that scenario an edge would always be made so this has a cost of 0. Any other time, spurious edge creation or removal must be penalised since this would not be possible. Finally, the modification of an edge matches the time of 2.668 seconds. So now the graph edit distance value should match the mean time an engineer would take to complete the same transformations.

Instead of measuring the number of edits required to make two graphs isomorphic, which is not a trivial problem for larger graphs due to the number of possible solutions to the problem, another common method is to find the Maximal Common Sub-graph (Bunke and Shearer, 1998). This metric takes two graphs and attempts to find a sub-graph which matches both. If two graphs are very similar, then both graphs will have a sub-graph which encompasses most of the graphs. Dissimilar graphs may only have a single vertex in common. The algorithm also works for labelled graphs, where the vertices or edges contain information about their position and relationships (Bunke and Shearer, 1998). The notation $G = (V, E, \mu, v)$ represents the graph and S is a

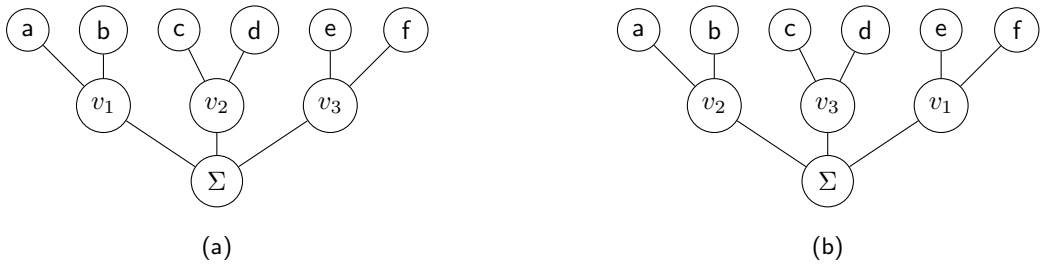


Figure 5.14: In a normal directional MCS score, the highest score would be 0.4, since only the master and the three group vertices are common. The graphs are identical from a signal flow point of view, with only the labelling of the groups is different.

sub-graph of G, such that $S = (V_S, E_S, \mu_S, v_S)$. This means that the sub-graph S is a subset of G, or $S \subseteqq G$. By creating sub-graphs that are isomorphic to the two graphs being compared, it is possible to keep building larger and larger sub-graphs until the largest isomorphic graph has been found. This function is expressed as $mcs(G_1, G_2)$ for comparing two graphs. The actual process for doing this is non-trivial and application specific. With the maximal sub-graph found, the number of vertices is taken $|mcs(G_1, G_2)|$ and forms the first part of the distance metric. Then the largest number of vertices is used as the ratio, shown in equation 5.10.

$$d\left(G_1, G_2\right) = 1 - \frac{|mcs\left(G_1, G_2\right)|}{\max\left(|G_1|, |G_2|\right)} \tag{5.10}$$

This has some useful properties that make it a good distance metric d. The first is that it is always bounded, such that $0 \leqslant d(G_1, G_2) \leqslant 1$, allowing for direct comparison as a single metric. Likewise if two graphs are isomorphic already, then $d(G_1, G_2) = 0$. Secondly, that $d(G_1, G_2) = d(G_2, G_1)$, so that the ordering does not matter and the comparison of one graph with another is always stable. Thirdly, that $d(G_1, G_3) \leqslant d(G_1, G_2) + d(G_2, G_3)$, meaning when comparing multiple graphs together then the distances are related to each other (Bunke and Shearer, 1998). Figure 5.13b shows the results of the Maximal Common Sub-graph before it is converted into a distance metric in equation 5.10.

The act of computing the maximum common sub-graph is fairly trivial when the graphs must match their labelling. A set of graphs is made by iterating over edges and adding the edge with the vertices it connects to if they are common in both. The largest graph from this set is then returned. In this case the labelling is only partially important. Given the graphs in Figure 5.14 the two graphs only have an MCS of 0.4. This is because, with labels, the only common portion is the Master (Σ) and the three group vertices v_1, v_2 and v_3. The structure is identical, in that vertices a and b are joined in one group, c and d another and e and f the final group where the only difference is the labelling of these groups. In this case these graphs should have an MCS equal to 0 to represent that they are functionally identical. To compute this the group vertices need to be re-ordered into different permutations and compared. With three vertices $(1, 2, 3)$ the total number of unique non-repeating permutations is 6: $(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)$. This means the number of permutations is equal to the factorial of the number of groups. For 3 groups this is easy to scan, but with 8 groups this becomes 8! of 40320 and 12! is 479,001,600 permutations to compare.

To solve this, a specialised algorithm is used instead of a brute-force search algorithm. This is based on the assumption that there are N input sources which all eventually lead to one output vertex. This means the extra vertices must be group vertices. This is because generally the structures are very similar, so it would be a waste to search through a large number of the permutations, as the mappings are not useful. For example, if in our twelve group example, one group only ever mapped onto one other possible group, it could be discounted from the permutations leading to a 11! of 39,916,800 possible permutations. This is still a significant amount, but is a reduction of 91.667% of the calculations.

Instead of searching the whole graph at once, it operates on the individual groups first, comparing each group from $G1$ with the groups in $G2$. It does this by taking the sub-graphs of the two groups and comparing them against each other using the MCS. The results are stored in an n-by-m matrix corresponding to the number of groups in both graphs. Then the groups from $G1$ are relabelled based on the maximum scores in this group. The cost matrix is scanned N times, which is how many groups are present in $G1$. Each time, the maximum common sub-graph is extracted. This will correspond to a row and column of the matrix, which is calculated and stored as a mapping since the groups are just index numbers. When it is extracted, the row and column are zeroed in the matrix, so on the next iteration the next highest group is found. If the mapping selects a group that already exists, because $n < M$, then it also creates a mapping for n to $M + J$, where J is the number of out of bounds mappings. This stops a problem where groups could be inadvertently grouped together. The groups in $G1$ are then relabelled as such, and passed to the MCS algorithm with the optimal label mappings applied.

Another way to score the similarity of the structure of two graphs is to take the features of the graph and calculate a distance from these features. Multiple features exist for measuring the shape or structure of a graph (Newman, 2018, pp.168–235). The advantage of using features over direct graph manipulations is that the sizes of the graphs do not impact the measurement. Using graph edit distance, if two graphs are very different in their shape, or very large in size, it can require many edits to make them isomorphic, with multiple iterations required to find the most optimal solution. A feature-based approach measures the features of the two graphs and compares the distance between the feature scores. These are often used in social sciences or computer network analysis. Centrality is one such feature used in social network analysis, which aims to identify if a vertex is important to a network, based on its centrality to the network, however less useful when applied to the graphs used in this Chapter.

$$P(k) = \frac{n_k}{n} \tag{5.11}$$

The degree of a distribution measures the way connections are distributed across the graph (Newman et al., 2001). The degree of a vertex in a graph is the number of edges it has to other vertices. By converting this into a probability density function, by taking a histogram or other binning measurement, the degree of a distribution can be found. A graph with a high degree has many vertices which are connected to each other, whilst a graph with fewer edges would score lower. Equation 5.11 formalises the probability, by counting the

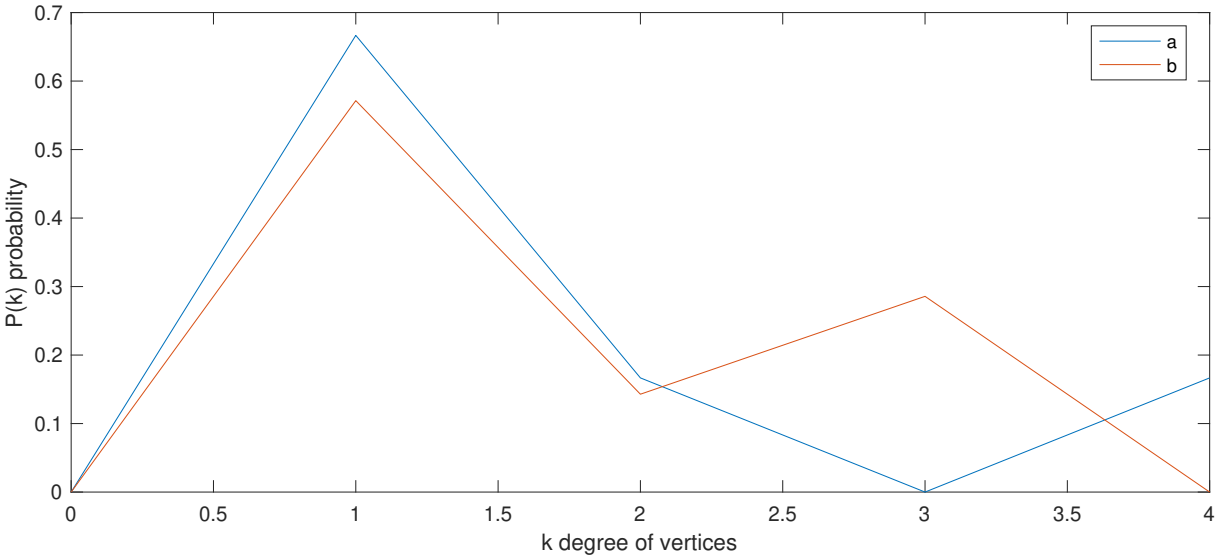


Figure 5.15: Probability density functions $P(k)$ of the graphs in Figure 5.10.

number of vertices n which have k numbers of edges. By dividing this by the total number of edges in the graph, a probability for the likelihood of a vertex having k edges is given as $P(k)$. If two graphs are similar in shape, they would have similar numbers of vertices with the same probability of them occurring.

For example, the two graphs shown in Figure 5.10 can be converted into a probability density function, shown in Figure 5.15. The densities here can be calculated using any conventional distance metric, such as Euclidean distance (Cha, 2007).

Another metric is to look at the features of individual vertices. This is often used in graphs to compare the similarity between two vertices rather than between vertices of different graphs, but the same principles can be applied (Newman, 2018, p. 212). Metrics such as the Jaccard or Cosine similarity can then be applied, as discussed earlier. These are forms of structural equivalence, which measure how structurally similar two vertices are based on their neighbourhoods. Another form of comparison is regular equivalence which measures how similar two vertices are based on the similarity of the connected neighbourhood.

$$\sigma_{ij} = \alpha \sum_{kl} A_{ik} A_{jl} \sigma_{kl} \tag{5.12}$$

Equation 5.12 gives a mathematical definition for what should be achieved, essentially finding the similarity metric of the i-th and j-th vertices by measuring the similarity of the k-th and l-th vertices. If the i-th vertex connects to k and j connected to l and the k-th and l-th vertex have a high similarity score σ_{kl} using the cosine or Jaccard similarity, then σ_{ij} would also be high. This problem is of course extremely intensive for just one level down.

Whilst a metric like this would be useful for the similarity of two vertices, it can be applied to our labelled graphs. Since the graph is effectively known to have an output vertex Σ and a pre-determined number of input vertices the shape of the graph can be compared by examining the vertices between these two. For example, in

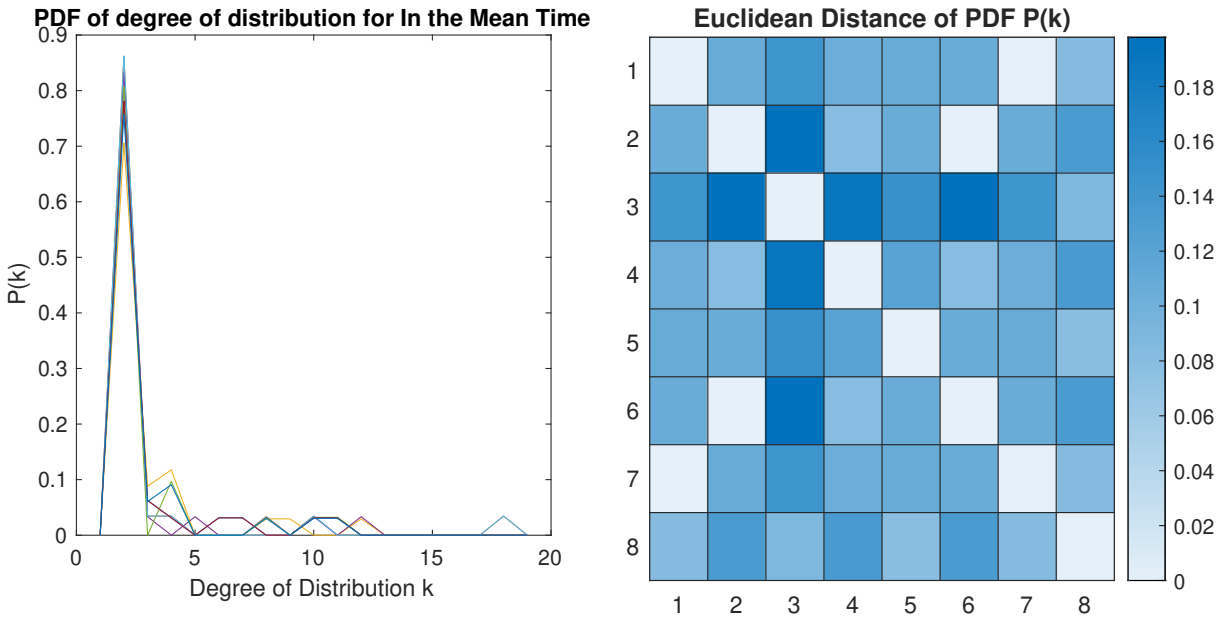


Figure 5.16: Probability density functions $P(k)$ of the graphs from the study by Ronan et al. (2015a) in Figure 5.12. Alongside the euclidean distances between each of the PDF vectors

a completely ungrouped system where tracks a, b, c and d connect straight to the output, then the output vertex would have a degree of $k_i = 4$. This could be compared to a graph with a degree matrix where a, b and c connect to a group v_1 first, and d connects to the master. In this case the degree of $k_j = 2$. Working through this for the cosine similarity, the similarity score for the two graphs as described would be 0.354 as shown in the following equation.

$$\sigma_{ij} = \cos\theta = \frac{n_{ij}}{\sqrt{k_i k_j}} = \frac{1}{\sqrt{4 \cdot 2}} \tag{5.13}$$

This is because, whilst they are from separate graphs, we can map the input and output vectors together to form a new graph to compare the vertices, such as in Figure 5.17.

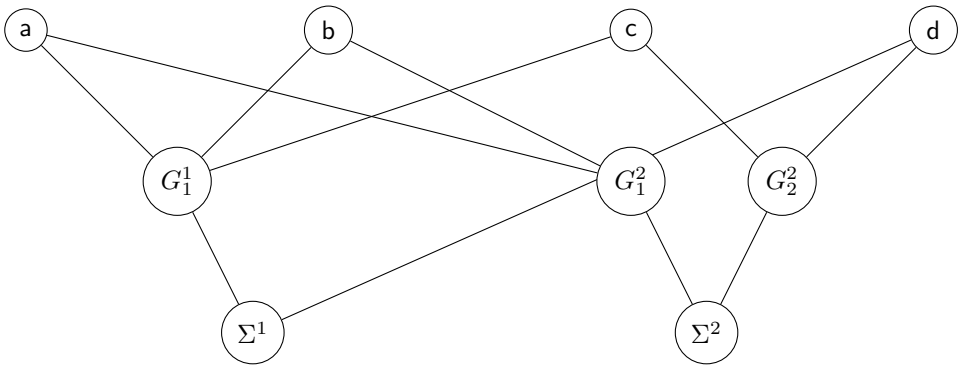


Figure 5.17: The two demo graphs in Figure 5.10 are combined to form a single graph allowing for the comparison between the two structures of the common input and destination vertices.

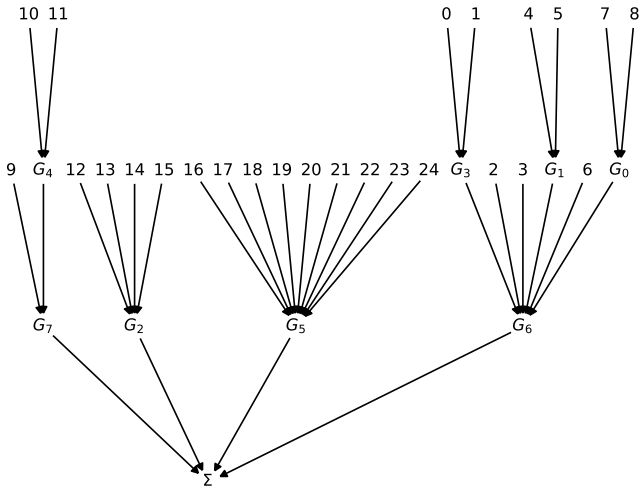


Figure 5.18: The generated plot for 'In The Mean Time' with subgrouping enabled.

Results of Graph Similarity Measures

Each of the sessions are passed through the automatic grouping processes to create a set of grouping structures. The information only contains the track name and instrument label. Figure 5.18 shows the generated structure for the song 'In The Mean Time', and again in Figure 5.19 but without instrument sub-grouping. The engineer created structures for the same song from the study by Ronan et al. (2015a) are shown in Figure 5.12.

To give a baseline performance, the twelve generated graphs were all compared to two fail-cases. One 'Bare' session where no groups were selected, and one 'Full' session where all the sources go into an individual group, which are each individually routed to the master. These are both fail-cases because grouping should be used to collect associated sources together for ease of control. Not using groups, as is the Bare scenario, leads to poorer mixes. Likewise having a full group for each input source also means no grouping has gone on. These are depicted in Figure 5.20.

Figure 5.21 compares the input data set from Table 5.9 against the Bare fail-case. The song 'Sleigh Ride' is the closest to the bare graphs example, along with 'I'm Alright', both scoring low GED scores. These are not normalised by track counts but do use the costs shown in Table 5.10. With a mean of 20.9, this means it would take an average of 20.9 seconds for the engineer to recreate the setup. Whilst there were 3 sessions in this sample, only one of them actually had any groupings applied. This might be because the session is so small that engineers did not create any groups, or there were no suitable structures applied. An interesting observation is that the distributions for the graph edit distance are particularly narrow, showing that there is a high consistency to undo the created structures to this empty style.

Compared to the one-to-one mapping, the empty structure is much closer to the actual structures. Figure 5.22 compares the input data set from Table 5.9 against the Full, or one to one grouping, failure-case. The mean GED for the empty fail-case was 55.9109 whilst the full fail-case is 326.640. This is mostly down to the fact

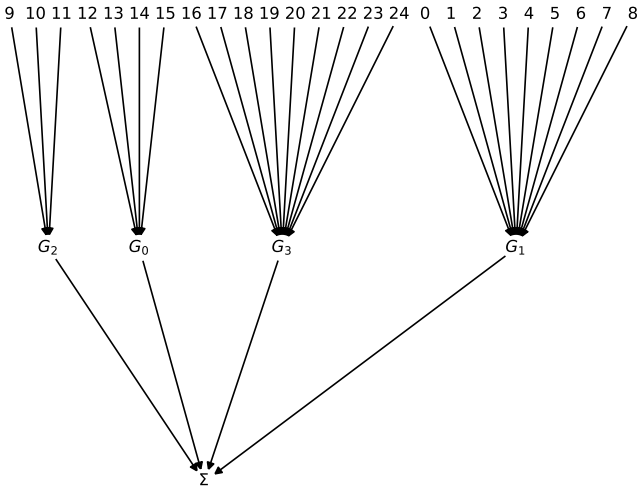


Figure 5.19: The generated plot for 'In The Mean Time' with sub-grouping not enabled.

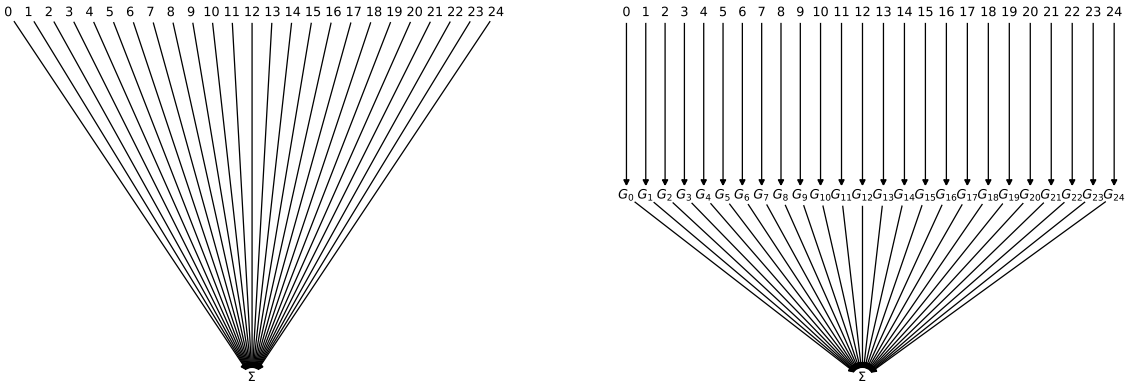


Figure 5.20: The two extreme fail-cases to compare against. The Bare plot (left) has no groupings made, whilst the Full plot (right) has a group for each input track

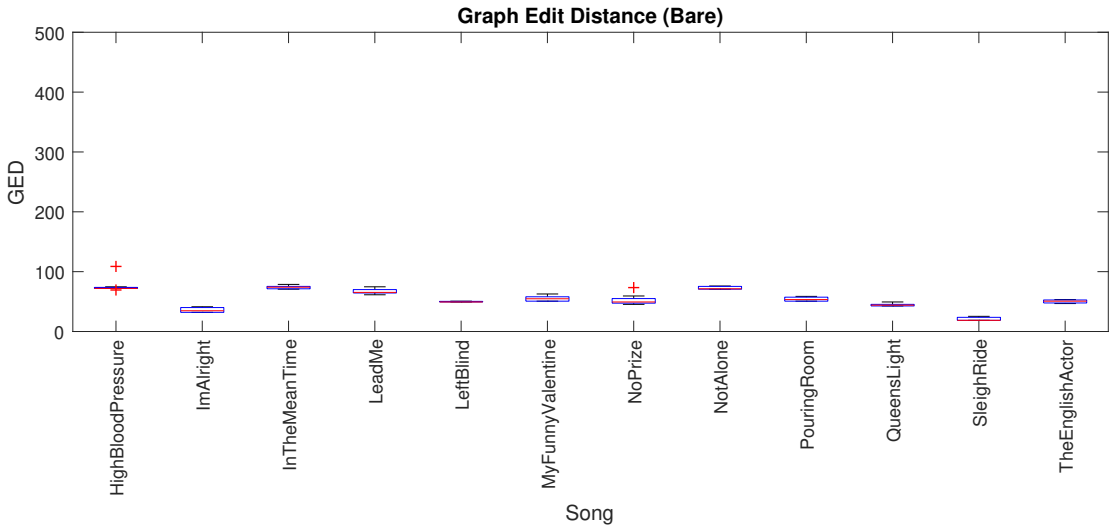


Figure 5.21: The graph edit distance for the user created graphs versus the Bare fail-case.

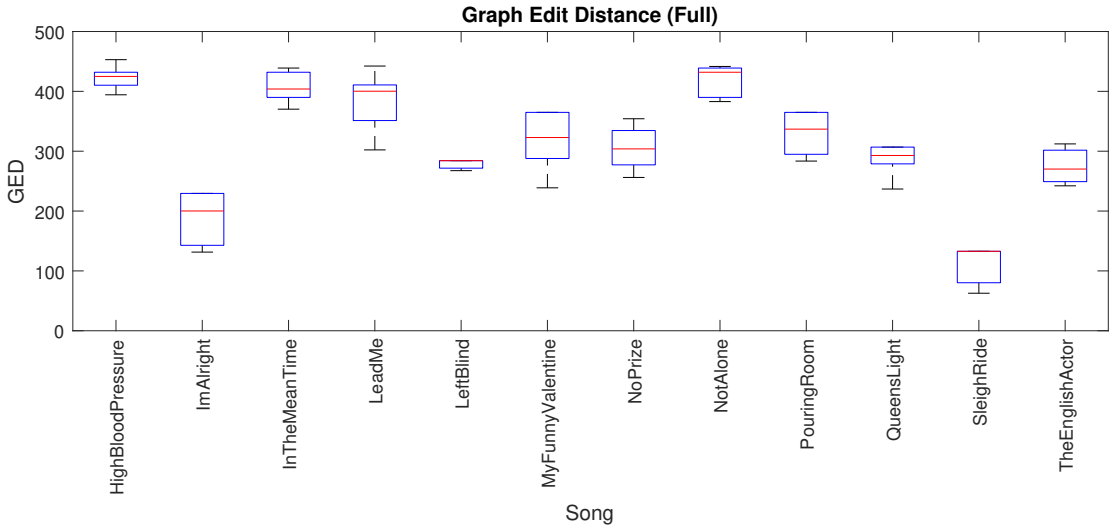


Figure 5.22: The graph edit distance for the user created graphs versus the Full (one-to-one groups) fail-case.

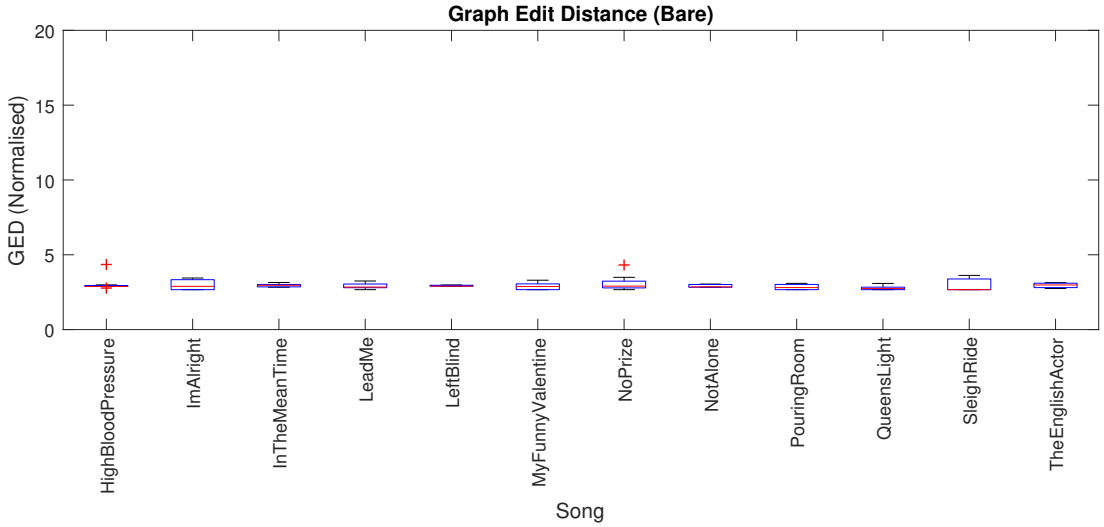


Figure 5.23: The graph edit distance for the user created graphs versus the Bare fail-case normalised by the number of audio tracks in the session.

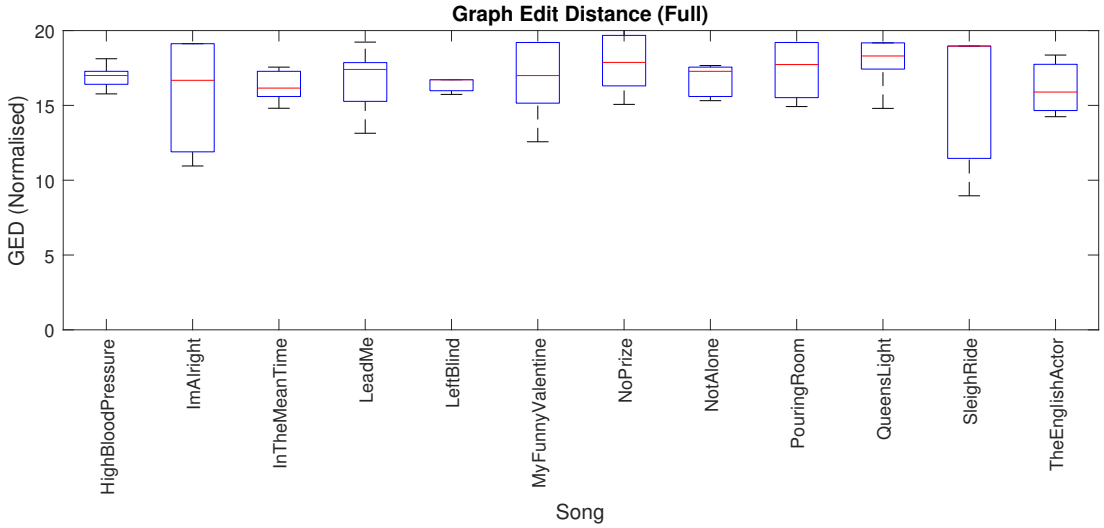


Figure 5.24: The graph edit distance for the user created graphs versus the Full (one-to-one groups) fail-case normalised by the number of audio tracks in the session.

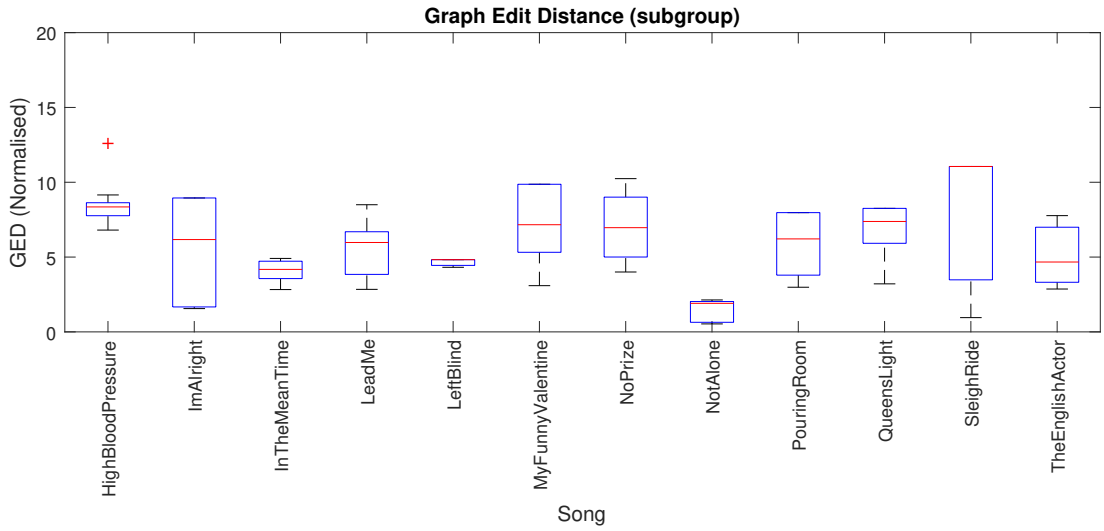


Figure 5.25: The graph edit distance for the user created graphs versus the generated graph with sub-groupings, normalised by the number of tracks in the session.

that adding new busses has a cost associated whilst leaving busses unused, effectively removing them, is given a cost of 0, with the cost of substituting the edges being the main driving factor.

Looking at the track normalised edit distances, shown in Figure 5.23, the tracks all mostly converge on the same GED (Normalised) cost of 2.9171, showing that the distances are indeed correlated with the size of the session involved. Figure 5.24 confirms this for the fully connected one-to-one grouping fail-case, where the GED (Normalised) costs is 17.0817.

The complete graph edit distance rankings for the two generated systems are shown in Table 5.11. This table shows the Bare and Full fail-cases, and the two generated graph structures. The first structure has sub-grouping applied, whilst the second does not. The sub-grouping generated structure is when multiple sources with identical instrument tags are grouped together first, before being passed as a single source to the automatic grouping structure. This results in more nested grouping structures with more groups. Figure 5.25 gives the graph edit distance for the graphs with sub-grouping applied, and normalised by track count.

For the most part, all the tracks with subgrouping applied were between the two fail-cases. This means they were closer to the average group structure as created in the real world than the full case, but were further away than having no groups at all. This would indicate that there are too many groups created and that there are more removals than there are insertions needed for the empty structure to be created. The only time where the means were lower for the Generated with Subgrouping is for Not Alone, where the Mean distance for the Bare was 72.7030 against the Generated score of 36.9381. One of the main factors for this score comes from the fact three of the 8 sessions have sub-grouping applied by the engineer. Therefore the creation of further sub-grouping structures, which is costly as shown in Table 5.10, is significantly lower than starting from the empty example. This also would explain why Not Alone has the second highest cost for the Full fail-case.

Looking at Sleigh Ride, which has the highest normalised graph edit distance of any of the tracks, is an interesting case because of the three submitted graphs only one of them had any group structures created.

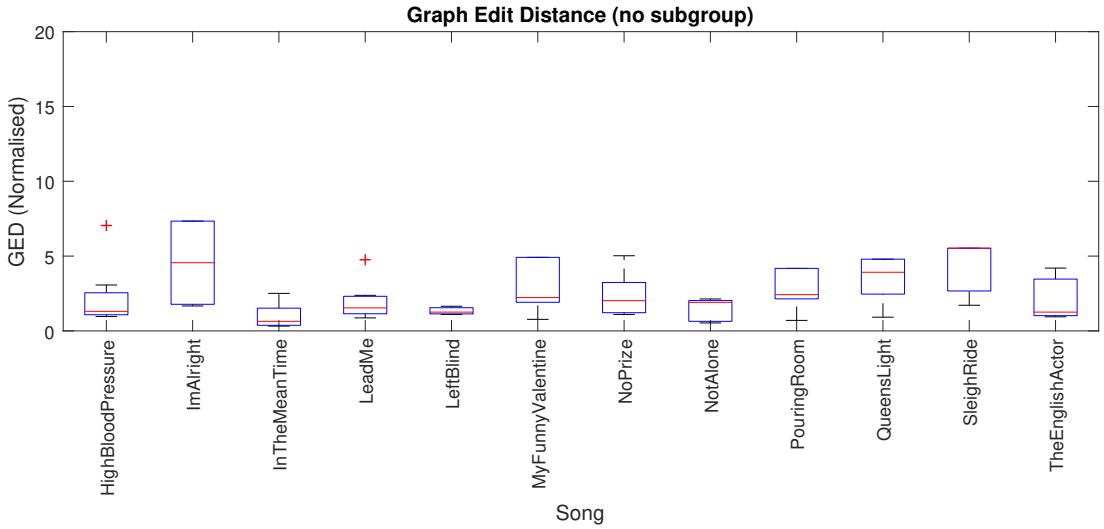


Figure 5.26: The graph edit distance for the user created graphs versus the generated graph without sub-groupings, normalised by the number of tracks in the session.

Song	Bare	Full	Generated	Generated (No Subgroup)
High Blood Pressure	76.4096	432.1238	212.4273	**54.6242**
I'm Alright	**35.7957**	188.8960	64.7059	52.6982
In The Mean Time	73.8702	407.5883	79.3802	**19.6772**
Lead Me	67.0811	382.2397	121.9783	**46.1213**
Left Blind	49.8027	278.6192	74.7109	**22.6780**
My Funny Valentine	55.0275	319.3672	131.9142	**50.5298**
No Prize	52.7773	304.8593	116.0723	**40.1900**
Not Alone	72.7030	417.9288	**36.9381**	**36.9831**
Pouring Room	54.0270	330.2079	107.7322	**50.7789**
Queens Light	**44.2888**	289.9167	110.3343	57.9340
Sleigh Ride	**20.8993**	109.4019	53.8116	29.7961
The English Actor	50.2473	274.8387	84.0507	36.2429
Mean	50.2473	274.8387	84.0507	**36.2429**

Table 5.11: The mean graph edit distances of all the test data from Table 5.9 showing the two fail-cases and the two generated examples. The bolded items are the lowest ranked value

Song	Bare	Full	Generated	Generated (No Subgroup)
High Blood Pressure	0.9663	0.9804	0.9056	**0.8651**
I'm Alright	0.9521	0.9733	0.8305	**0.7393**
In The Mean Time	0.9960	0.9804	0.8143	**0.7513**
Lead Me	0.9783	0.9818	0.9164	**0.8720**
Left Blind	0.9841	0.9840	**0.7600**	0.7918
My Funny Valentine	0.9654	0.9840	**0.8333**	0.8608
No Prize	0.9493	0.9750	0.8365	**0.4383**
Not Alone	0.9961	0.9804	**0.8461**	**0.8461**
Pouring Room	0.9601	0.9840	0.8795	**0.8591**
Queens Light	0.9497	0.9833	0.8143	**0.7901**
Sleigh Ride	0.9167	0.9778	0.9286	**0.7619**
The English Actor	0.9686	0.9714	0.8974	**0.8887**
Mean	0.9648	0.9802	0.8492	**0.7857**

Table 5.12: The mean Euclidean distance of the maximum common sub-graph for the Bare, Full and two generated graphs against the user created structures.

This makes it no surprise that the Bare test was the best performer for this song. Of the one session that did have groups, the mean edit distance was 0.9529, extremely close to the projected graph.

The no-subgrouping generated graphs were the best performers on average. Table 5.11 shows that, over all the evaluation mixes, it scored the lowest distance eight out of twelve times, with the mean total distance of 36.2429 compared to 84.0507 for the generated with subgroups, and 50.2473 for no grouping at all. The four songs which failed to score the lowest were Queens Light, I'm Alright, Sleigh Ride and Left Blind. It is no coincidence that these four all came from the balance mix investigation from Chapter 4. The mixes created in that environment had no overall benefit for the engineers to create busses and groups as there was no extra processing that could be applied, it was just for processing. Compared to the data set by Ronan et al. (2015b) which were fully completed mixes with processors, it just shows that later stage mixes tend to have busses.

The maximum common sub-graph (MCS) shows more confirmation that the generated structures have similarity with those provided by the engineers. Table 5.12 shows that, out of all twelve scenarios, every single one performed better using one of the two generated graphs than the empty or fully connected graphs. Sleigh Ride, which performed poorly for the GED metric, showed a significant improvement over the other two, although the distances are still high. No Prize shows a strong agreement with the generated examples, because in this scenario the majority of the mixes have a grouping structure. As with the GED scores, the best scores generally are when subgrouping by instrument is not enabled. The MCS has a propblem with high variation in the sub-grouping structures, whereby if one aspect of the structure is wrong it can ignore whole leafs of the graph. Therefore groups which are super-groups, whereby they could be split but are not, score significantly lower here than in the graph edit distance relatively. Likewise, lots of smaller groups are also punished. So whilst the

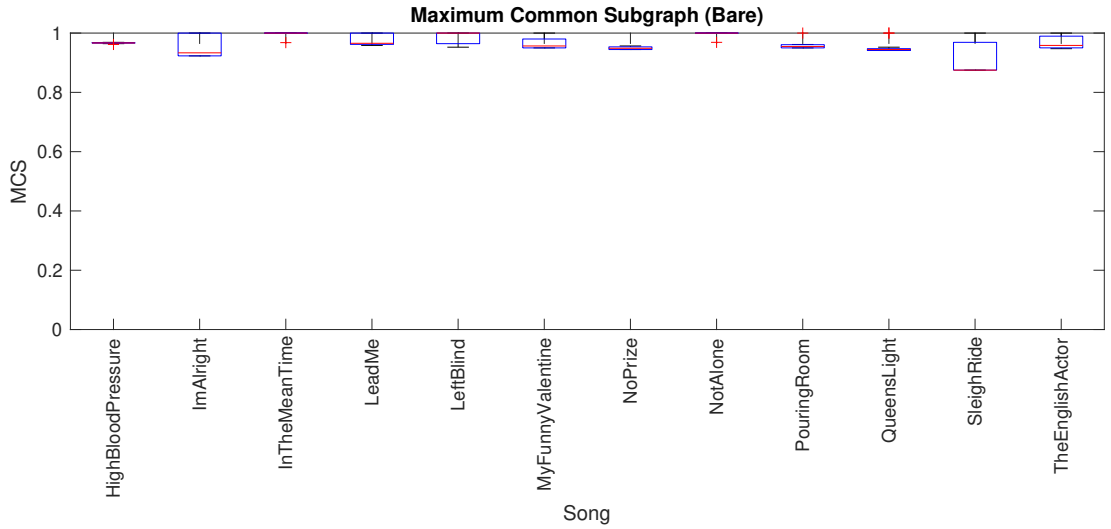


Figure 5.27: The maximum common sub-graph for the user created graphs versus the Bare fail-case.

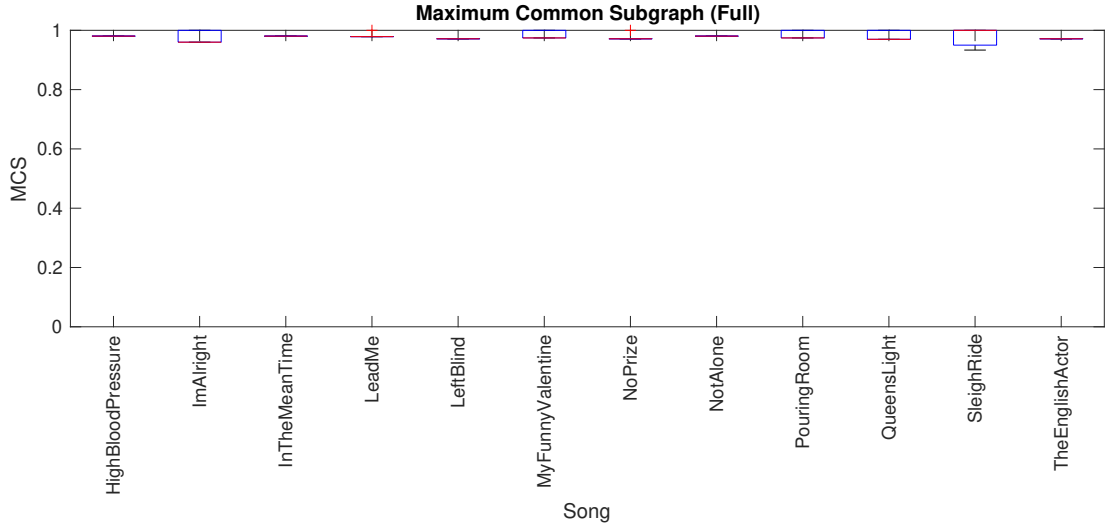


Figure 5.28: The maximum common sub-graph for the user created graphs versus the Full fail-case.

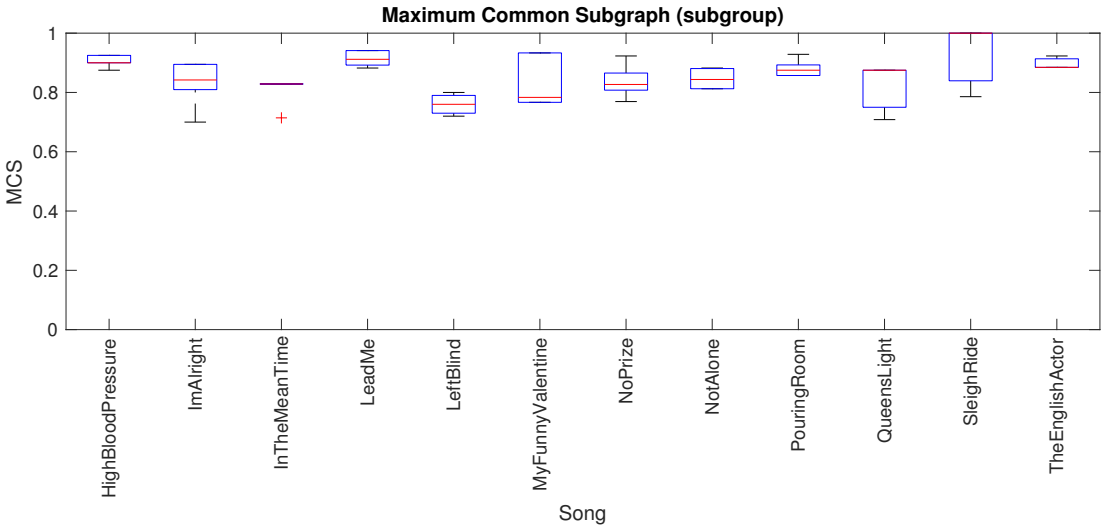


Figure 5.29: The maximum common sub-graph as a distance measure for the user created graphs versus the generated graph with subgrouping

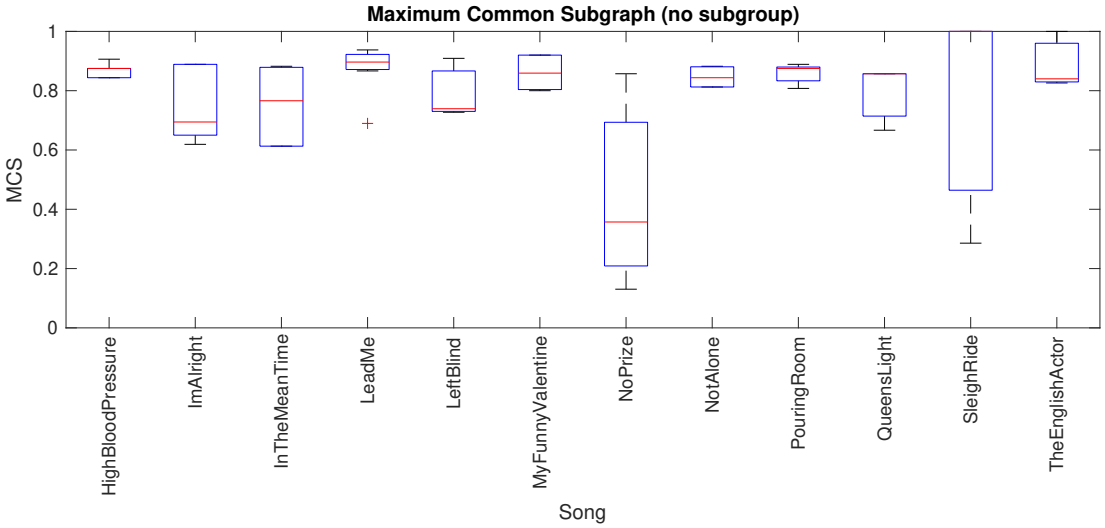


Figure 5.30: The maximum common sub-graph as a distance measure for the user created graphs versus the generated graph without subgrouping

Song	Bare	Full	Generated	Generated (No Subgroup)
High Blood Pressure	0.1605	0.1747	0.2565	0.1045
I'm Alright	0.1798	0.2137	0.2442	0.2143
In The Mean Time	0.1647	0.1791	0.1536	0.1099
Lead Me	0.1582	0.1724	0.2007	0.1440
Left Blind	0.1285	0.1454	0.1704	0.1269
My Funny Valentine	0.1731	0.1980	0.1860	0.1637
No Prize	0.1728	0.1910	0.2147	0.1518
Not Alone	0.1388	0.1530	0.0985	0.1159
Pouring Room	0.1366	0.1599	0.2229	0.1446
Queens Light	0.0934	0.1214	0.2577	0.1733
Sleigh Ride	0.1831	0.2346	0.3591	0.2398
The English Actor	0.1606	0.1764	0.2243	0.1631
Mean	0.1542	0.1766	0.2157	0.1543

Table 5.13: The mean Euclidean distance of the Probability Density Functions for the Bare, Full and two generated graphs against the user created structures.

distances still appear to be high, they show that, on average 21.43% of the graph is of an identical shape to the engineer designed graph.

The Probability density function allows for the comparison of the features of the graph being measured. This is useful for graphs which do not necessarily need the same structure but just a similar one. For example, the MCS could ignore a whole leaf if just one vertex in the chain is incorrect or different, even though the edit distance may be small. Likewise, there may be many edits needed to make a labelled graph match but the overall structure is the same. Therefore the features of the graph can be used to judge similarity. One such feature of the graph is the degree of distribution, which measures the distribution of the connection over the graph (Newman et al., 2001).

Table 5.13 gives the mean Euclidean distance between the four test graphs: Bare, Full, Generated and Generated with no subgroup. The score is a distance metric, since it is the Euclidean distance, so a smaller number means the graphs are similar. Because of the sparse connectivity of the graphs in question, this distance metric will lend itself to the empty graph (Bare) since most vertices only have one output, and those with an input will have multiple inputs. The Generated graph with subgrouping does have a further distance by this feature because of the extra groups being added. These extra groups change the shape of the probability density function since these extra vertices will have at least 2 or more vertices connected to them. This fundamental change in shape confirms that sub-grouping by instrument type is not performed by most engineers. However, just like with the graph edit distances in Table 5.11, the song Not Alone does score very well for this case. Again, this is because for this song several engineers did group by their instrument labels. The Generated with No Subgroup scored smaller distances than the Bare and Full fail tests eight out of the twelve test samples,

with the same songs failing as before. This again provides further evidence that the provided system does produce grouping structures which are suitable for engineers to use.

Using these three distance metrics it is clear that the grouping structures are able to select groups which are sensible to the engineers, based upon what they have done. In each of the three metrics it performed better than the fail-cases almost every time, showing they are better than worst case and no worse than the spread of grouping structures already proposed, which is highly variational.

5.3.2 Group Naming

The naming of the groups is derived from the structure of the groups, and based on the instruments inside them. From the song in the data set In The Mean Time, the engineers often created a bus for the drum kit components in the mix. Naturally this was then labelled Drums, or Drumz by one engineer. Therefore a system needs to be put in place to determine how closely the label picked by the proposed system was to the collection of labels given by the engineers.

One method for measuring the similarity between strings is to look at the edit distances between them. Similar to graph theory, there are two major forms of distance measurements based on string editing. The most simplistic is the Levenshtein distance (Levenshtein, 1966). This measures the number of inserts, deletions and substitutions that must be made to a string to get it to match another string. This measurement suffers from the same base problem as the graph edit distance, in that longer strings which require more edits due to their size would get worse scores than smaller strings requiring the same or even slightly fewer edits. Therefore it is important to normalise the strings chosen. This also metric assumes that there is a root string similarity between the two.

Another metric is the longest common sub-sequence (LCS) (Bergroth et al., 2000). This measures the longest common sequence found between two strings, similar to the graph theory maximum common sub-graph. Unlike the longest common substring, which finds the maximum set of characters in the string that directly match, the subsequence measurement only considers the order of the characters. For example, consider the following two strings ‘Guitars’ and ‘Gtrs’. The last is a common abbreviation where the vowels are removed. In this case, the longest common substring is 2, the ‘rs’ at the end, whilst the longest common subsequence is 4, ‘Gtrs’. This can be normalised by the incoming string length, such that if the comparison was between ‘Guitars’ and ‘Gtrs’ the distance would be 4/7.

Both of these are useful for distances based directly on the string, but they both overlook the important semantic information contained in the string. For example, ‘Guitar’ and ‘String Instrument’ has a really low comparison score between the two using LCS and Levenshtein distance. But semantically these have a common inheritance. Therefore a semantic matching system can be used to compare the two. One such tool is WordNet which holds relationship data between dictionary words using relationship such as Synonymy, Antonymy, Hyponymy and Meronymy (Miller, 1995). These links allow words to be linked in a more human-understandable relationship, such that two strings of various length and physical spelling could be measured and quantified as to how well it matches the meaning of another.

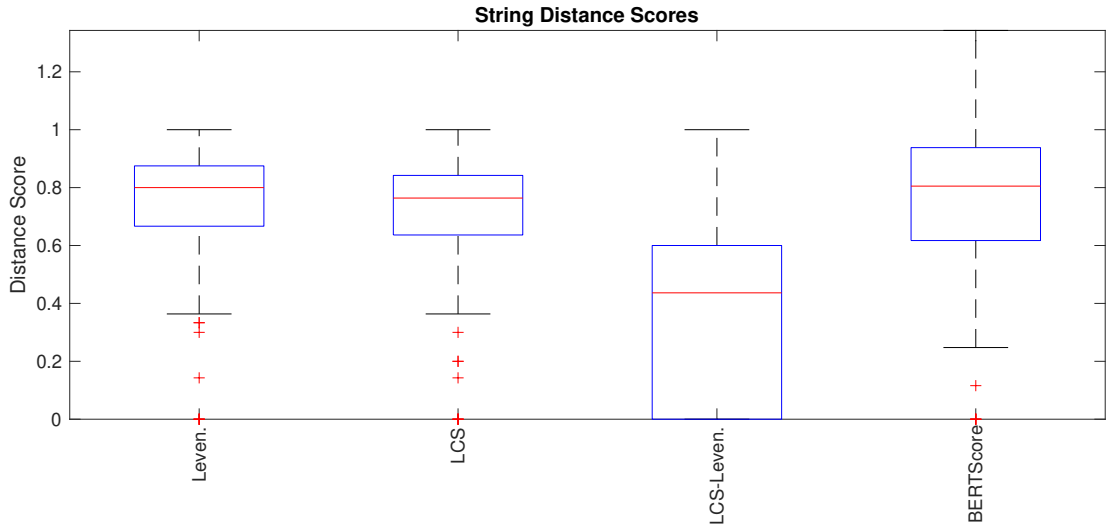


Figure 5.31: Distribution of the string distance scores, using Levenshtein distance, Longest Common Subsequence (LCS), a combination of the two and BERTScore similarity. In all cases lower is better.

Bus Name	Occurrence
Drums	19
DRUMS	11
Bass	10
StSub2	8
Drum Bus	7
StSub1	7
StSub3	7
Bass Bus	6
Keys	6
Piano	6

Table 5.14: The ten most common group names from the 279 groups created across the sessions

Generated Name	Occurrence
Human voice	57
Drum kit components	55
Electric guitar	27
Musical instruments	21
Bass guitar	21
Keyboard instrument	17
Acoustic guitar	9
Classical music instruments	8
Percussion instruments	6
Vibraphone	5

Table 5.15: The ten most common group names generated from the 279 groups in the session

Bus Name	Generated Name	Occurrence
Drums	Drum kit components	17
DRUMS	Drum kit components	10
Bass	Bass Guitar	10
Vox	Human Voice	6
StSub3	Human Voice	6
Drum Bus	Drum kit components	6
Bass Bus	Bass Guitar	5
StSub1	Drum kit components	5
StSub2	Musical instruments	5
BASS	Bass Guitar	4

Table 5.16: The ten most common group names with their highest occurring generated name from the 279 groups

Bus Name	Generated Name	LCS Score	Bus Name	Generated Name	LCS Score
Piano	Piano	0	StSubBus2	Human Voice	1
Guitars	Guitars	0	Squeeze Bus	Accordion	1
Flute	Flute	0	D&B	Musical Instruments	1
Violin	Violin	0	Vox Dbl Bus	Human Voice	1
Harp	Harp	0	StSub3	Human Voice	1
Guitar	Guitars	0.1429	StSub2	Human Voice	1
Snare Bus	Snare Drum	0.3	Keys	Piano	1
Electric gtr1b	Electric Guitar	0.3333	BG	Human Voice	1
Electric gtr2b	Electric Guitar	0.3333	BG's	Human Voice	1
Guitars Bus	Guitars	0.3636	Kick	Bass Drum	1

Table 5.17: The top ten best scoring generated group labels and bottom ten worst scoring generated group labels, using the Levenshtein distance

From the twelve session groups in the combined dataset from Table 5.9, there are 279 groups to be evaluated. The testing shall pass each group structures instrument through the naming portion of the system. Then the output text shall be evaluated using the three metrics explained above. To give an idea of the data variety, Table 5.14 gives the ten most common group labels by occurrence without accounting for any substitutions or case sensitivity. As can be seen, 30 entries alone are for 'Drums', and another 7 for 'Drum Bus'. Table 5.15 gives the ten most common generated names by the system, showing the more clinical naming convention used. This is because the subject of the page is more general than the unique naming conventions used by engineers. The most common source to generated names are shown in Table 5.16. After the removal of duplicated entries, where the same tag was generated with the same label given as an input group, there are 150 unique input to generated name pairs. These are grouped purely on the face of the string provided, showing again the variation in the source data. Use of capitalisation is inconsistent and abbreviations are common throughout the dataset. Likewise some engineers will add the term 'Bus' to the end of string. This occurred 52 times, whilst the suffix 'Aux', shorthand for 'Auxiliary' was only used 2 times.

Of the 150 group names generated, the mean Levenshtein distance was 0.73 after being normalised for the string lengths. This takes the normal Levenshtein edit distance, divides it by the maximum length of the two strings. This value is bounded therefore from 0 to 1. This is converted to a distance 1 subtracting the normalised Levenshtein score. If the duplicated entries are removed and just focusing on the 150 unique combinations the score becomes 0.6055. A score of 0.61 shows that the string matching is quite low, with large edits needing to happen to complete the strings to make them physically match. Table 5.17 gives the top ten performing pairs and the ten worst performing pairs. Of the top performers, these occur because the group is entirely made up of that one instrument, so the lowest common ancestor is itself. The engineer also labelled the groups the same way, giving identical strings. The rest of the matches are very similar with only

Bus Name	Generated Name	LCS Score	Bus Name	Generated Name	LCS Score
Piano	Piano	0	D&B	Musical Instruments	1
Guitars	Guitars	0	Keys	Piano	1
Flute	Flute	0	BG	Human Voice	1
Violin	Violin	0	BG's	Human Voice	1
Harp	Harp	0	Kick	Bass Drum	1
Guitar	Guitars	0.1429	Clap	Percussion Instrument	0.95238
Electric gtr1b	Electric Guitar	0.2	EK	Keyboard Instrument	0.94737
Electric gtr2b	Electric Guitar	0.2	Perc	Musical Instruments	0.94737
Snare bus	Snare Drum	0.3	BCK	Human Voice	0.90909
Guitars Bus	Guitars	0.3636	StSubBus3	Human Voice	0.90909

Table 5.18: The top ten best scoring generated group labels and bottom ten worst scoring generated group labels using the LCS method

mild semantic changes throughout. The string length causing issues can be seen clearly, with 'Guitars Bus' and 'Guitars' scoring 0.3636, despite them being essentially the same meaning.

The LCS scores also show the distance from a common value, with the mean LCS distance score being 0.70. For unique only pairs this does drop to 0.58. This distance score is again high. Part of the problem for both of these can be attributed to the extra words often added by the generation process. Take the most common pairing shown in Table 5.16. The user supplied word of 'Drums' is attributed to 'Drum Kit Components'. This gives an LCS of 5, which coincidentally is 'Drums'. The length of the generated string is 19, so the distance score is $1 - \frac{5}{19} = 0.7368$. This high distance score doesn't actually reflect very well the meaning of the words. Table 5.18 gives the ten best pairings using LCS and the ten worst pairings. For the top pairings, as with Table 5.17 the identical string names provide an ideal comparison scoring a distance of 0. Again here the LCS scores are similar to the Levenshtein because the differences are mostly string substitution scores. 'Guitars Bus' is exactly 4 edits away from 'Guitars' by removing the ' Bus' from the end of the string. Likewise the LCS is 7 ('Guitars') which gives a score of $1 - \frac{7}{11}$ which is equal to the Levenshtein distance of $\frac{4}{11}$.

The two scores can be combined, with the string extracted from the LCS algorithms being used as the generated score and processing the Levenshtein distance. Using the example of 'Drums' and 'Drum Kit Components', the LCS is itself 'Drums'. The 'Drum Kit Components' is substituted for the 'Drums' score, giving a Levenshtein distance of 0. When using this approach the score drops to 0.39 across all uses and 0.33 for unique pairings. This still shows that it does not entirely cover the user supplied names, but they are not random and are usually overly long or specific.

The problem with both of these metrics is the semantic meaning is often lost. Take the example 'Vox' and 'Human Voice' from Table 5.16. This has a Levenshtein distance of 0.82 and an LCS of 0.82 as well, although, the naming itself semantically does make sense. 'Vox' is often used as short-hand for 'Vocals', which means

the Human Voice. Instead the names can be considered using a semantic description such as analysing the WordNet relationships(Miller, 1995).

In this scenario using WordNet would not be entirely appropriate as the engineers and the label generation could pick multiple words. Instead a system called BERTScore can be used (Zhang et al., 2019). This system calculates the semantic relationship between two words as a score into a matrix by using a trained model of words, like WordNet. This gives each word a combination pairing score. The maximum relationship is then taken and multiplied with the importance of each word in the sentence given. The result is a measurement of semantic similarity. For our example 'Vox' and 'Human Voice' scored a similarity score of 0.13, which is still very low. This is not a completely unexpected result, because 'Vox' is a short-hand term and most likely not well captured by the pre-trained models. The expanded term 'Vocal' and 'Human Voice' showed a similarity score of 0.51, far higher than the short-hand 'Vox'.

Figure 5.31 shows the comparison of all four distance metrics for the string selection, showing that the LCS-Levenshtein combination shows how substitution and short-hand labelling affects the scores. Semantically the measurements are similar with a strong focus on the literal meaning, however the labels themselves would need some work to grab a semantic definition for them. This could be gathered through engineer interviews or further data collection.

5.4 Conclusion

This chapter introduced a novel method for automatically assigning tracks to groups and presenting this structure to an engineer. Research from Chapter 4 and from previous studies has shown that engineers who use groups tend to produce stronger mixes, which score higher in perceptual listening tests (Jillings and Stables, 2017d; Ronan et al., 2015b). Previous research into automatically generating groups of sub mixes for engineers used feature extraction to best group their work, but with minimal evaluation except against the same source data (Ronan et al., 2015a).

Therefore a system was built to use the instrument labels often given to a track by engineers in the session, either through the track name, labelling system or using automatic classification systems. With the instrument labels gathered, a knowledge system was needed to find a suitable set of relationships between each instrument. This again is not a consistent study with multiple sources providing different relationships and structures, none of which provide an exhaustive list of relationships. Using Wikipedia, through its SPARQL endpoint DBPedia, common instrument relationships can be gathered. Since this is a public dataset it is often updated with new information allowing for the knowledge tree to grow.

When an instrument relationship is needed, a query is made to the SPARQL endpoint to gather the Subjects of that page. These are scanned for a depth of N times, or until the chosen root subject is found. In this case the root subject was 'Musical Instruments'. This is done for each instrument to consider and placed into the same graph G. The graph is then pruned, with only the vertices and edges connecting the instrument pages to the root 'Musical Instruments' vertex (G_1).

The graph is then flattened so that every vertex has all other vertices in direct paths from themselves to the root in their neighbourhood. This boosts the performance of the Jaccard similarity which allows for the instruments to be compared to each other. Instruments which have very similar neighbourhoods will score a high Jaccard similarity value. The instruments are then clustered together to form groups based on this Jaccard similarity value, placing instruments which are closely related together into the same group.

The labelling of the given group then uses the originally pruned G_1 graph. The label is chosen by finding the lowest common ancestor of all the instrument labels in the given group. This has the advantage of selecting the most closely related Subject as possible.

The system was evaluated by comparing it against 81 mixes created by real-world engineers form the study in Chapter 4, and in previous studies (Jillings and Stables, 2017d; Ronan et al., 2015b). With the 81 mixes, the graph edit distance, maximum common sub-graph and probability density functions of graph features were used to show how similar the generated structures are to structures provided by engineers. Over the twelve songs in the dataset, the generated structures achieved higher similarity to the generated graphs than the fail-cases of no grouping or over-grouping.

The naming of the graphs is harder to quantify. This is because of the language used by engineers is often truncated. For example, 'BG' which is used to refer to 'Backing Vocals'. Using similar measurements such as Levenshtein distance, Longest Common Substring and a combination of the two, showed that the labels generated are not similar to those provided to engineers except in a few scenarios. Using the semantic scoring similarity through BERT the similarity is shown to be there on a semantic level (Zhang et al., 2019). For this to work effectively, data would need to be gathered on the language used by engineers to build a new model to evaluate how similar the use language is.

Chapter 6

Automatic Masking Reduction for Balance Mixing

6.1 Introduction

The mixing phase is performed after all the recording and overdubbing tasks have been completed (Huber and Runstein, 2005, p. 10). Audio mixing involves taking a series of audio tracks and combining them, with additional processing, to produce a single audio stream. This task has evolved greatly from the early phonographs to use completely solid-state, in-the-box mixing tools such as Digital Audio Workstations and other Digital Consoles (Burgess, 2014, p. 16).

By automating this to ensure a suitable starting position, the engineers' subjective interpretation of the initial piece can be minimised, to ensure that all aspects are suitably adjusted for the performance. This will help the engineer produce a rough mix faster, and more consistently, than manually conducting the mixing phase. This chapter introduces a method for offline automatic mixing by discovering a suitable set of values for the gain coefficient g_m to minimise the masking between tracks. In section 6.2 the new model is presented and evaluated in section 6.2.3.

6.2 A Genetic Algorithm for Audio Mixing

6.2.1 Model

Genetic Algorithms and Evolutionary computing are discussed in Section 2.5.3. As a quick recap of the process, evolutionary computing is a solver algorithm which searches a parametric space for an optimal solution. The space is explored using mutltiple possible solutions at once, called chromosomes. The dimensions of the chromosomes represents the dimensionality of the mixing space. Each chromosome is evaluated against the search space using a cost function, also known as a fitness function, giving a numerical ranking of how suitable that chromosomes solution is. The best performing chromosomes are kept, with the poor ones being discarded after each iteration. The best performing chromosomes are then combined using crossover to create a new

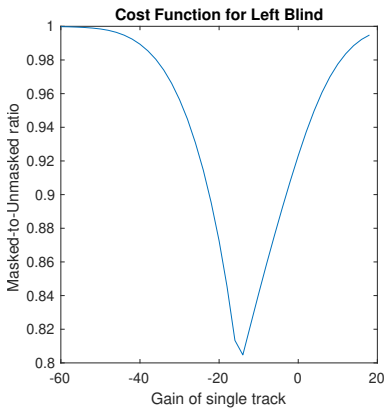


Figure 6.1: The performance of the cost function on two tracks with significant spectral overlap: drum kit overheads and an electric guitar of 'Left Blind'. When modifying the gain of one track a minimum can easily be found.

generation of chromosomes. This entire process is iterated a number of times and is finished if either the number of set iterations has elapsed, or the chromosomes have converged.

Fitness function

For each chromosome to be evaluated, a suitable fitness score must be calculated. This cost function will take the incoming audio streams and calculate a ranking from 0 to 1, where 0 is the most suitable solution and 1 the least suitable. In this case, a score of 0 would indicate no masking between the tracks and 1 a completely masked track. The chromosomes can be ranbked to select the best chromosomes which are used to generate the next population cycle.

The Masked-Unmasked Ratio (MUR) in Equation 2.15, with the implementation explained in Section 4.3.4, calculates the masked to unmasked ratio from a given signal using the Glassberg-Moore (Moore et al., 1997) auditory model. This is shown in equation 4.17. By operating this through each track x_n then a vector r containing the Masked-Unmasked Ratio can be obtained, with each value holding the MUR for each track.

Each chromosome h_i is evaluated using a cost function $f(h_i)$ which returns a cost value c_i as shown in Equation 6.1, where r is the vector of the track MUR responses from equation 4.17 and N is the number of tracks being processed. This takes the minimum MUR for the tracks to return the worst performing MUR value. This is mean-squared to make sure that slightly masked tracks are less impacted than heavily masked tracks which would tend to 1. This is then inverted by subtracting 1, giving it a score of 0 to 1, where 0 would be no masking and 1 would be completely masked.

$$c_i = 1.0 - \frac{\min(r)^2}{N} \tag{6.1}$$

To proove the suitability of the cost function two signals with conflicting spectra, the drum kit overheads and electric guitar track from 'Left Blind', were passed through at various levels of relative gain. Before applying the gain, the tracks were normalised by the loudness model to 70dBSPL (Ward et al., 2012). This means

for the MUR calculation, before the modified gains are applied, both tracks should have equal perceptual loudness. Then the electric guitar track's gain was changed from -20dB to +20dB in 5dB increments, with a new MUR taken at each point. Figure 6.1 shows the curve of this cost function. When the electric guitar's gain is too low, one track is severely masked which causes an increase in the cost making it inappropriate as a solution. Likewise a gain too high causes the same phenomenon. But most importantly the cost function shows a specific minima to be found using this methodology.

With a fitness function established, the chromosomes can be passed through to calculate their fitness. The size of each chromosome equals the number of tracks to be mixed, so four tracks would require four gain variables, giving a chromosome with four values. To simulate a suitable range of control, the number range of the chromosomes is limited to be between -96 and +24, to represent the decibel value of a typical DAW fader range. Initialising the chromosomes for the first run requires randomnly selecting a floating point number between this range. The process for calculating the cost function for each stage of a multi-track session is as follows. The gains from the chromosome vector h_i are applied to each track x_n, and are summed together to create the 'master bus' mix y_i an engineer would hear, as in equation 6.2.

$$y_i = \sum_n h_i[n] x_n \tag{6.2}$$

This master mix y_i is used to normalise the specific loudness of the mix to 70 dBSPL using the same loudness normalisation model as before. This ensures that a louder mix, or quieter mix, does not perform better due to the change of the sensitivity of the human ear (Fletcher and Munson, 1933). Each audio track, x_n, is multiplied by the gain coefficient stored in chromosome h_i. Because the gain stages are linear equations, the Δ gain variable for the loudness normalisation is applied to each mix stage as well, thereby normalising the individual tracks before further processing as shown by Equation 6.4.

$$\Delta = 70 - \mathrm{g}\,(y) \tag{6.3}$$

$$\hat{x}_n = \Delta h_i[n] x_n \tag{6.4}$$

The $\hat{x}_n$ are then passed through the MUR calculations in equation 4.17 to get the vector of results r, which are then passed through the cost function in equation 6.1 to give the cost function of c_i of chromosome h_i. The chromosomes are then sorted in rank order of the best performing chromosomes. The worst performing are discarded with the highest performing then combined using a crossover function to create a new generation of chromosomes.

For the selection stage, elistism is used to help preserve the best performing generation and provide resistance to over convergence on sub-optimal solutions. Each stage 25% of the population is preserved. The remaining 75% of the population is generated using crossover techniques. The parents are selected from the population

using Roulette wheel selection (Katoch et al., 2021). This has an advantage due to the desireability to find a strong performing candidate quickly, whilst ensuring enough randomness with lower performing members.

For crossover and population generation, the single-point crossover is the simplest to implement and other crossover implementations that exist are specific for chromosome types and properties. For the four chromosome pairs, 2-point and k-point crossover implementations are not appropriate due to the smaller size. Uniform cross-over methods would provide some implementation, along with shuffle, but these would need some method of blending for the floating point nature of the chromosome structure.

The combination of elitism, single point cross over using roulette wheel selection and blending of the crossover point will create an algorithm that should explore the space quickly whilst being resistant to local minima. Since the cost function itself returns the worst performing selection, its space should not be too complex to find a suitable minima point.

The final stage is the mutations, this models the biological principle of gene-copy errors. There is a probability $P(m)$ that a gene in a chromosome will be mutated, called the mutation rate, A random number is generated for each chromosome, if this number is less than mutation ratethen that chromosome must experience a mutation. Then a random entry of the vector is randomised to be a number within the defined space of -96 to $+24$.

6.2.2 Methodology

To test the performance of the genetic algorithm mixing function, the following methodology was used. Four songs were obtained from the open multi-track test bed (De Man et al., 2014b):

- **I'm Alright** by Angels in Amplifiers (IA)
- **The English Actor** by James Elder & Mark M Thompson (TEA)
- **Queen's Light** by Dino on the Loose (QL)
- **Sleigh Ride** by The Funny Valentines (SR)

The songs were truncated to only be 30 seconds in length, centred around the chorus to ensure the most consistent musical activity between samples. To compare the performance of the model, four mixes were created to provide stimuli for objective and subjective performance evaluation. The mixes were:

- A unity mix where all values are set to 0dB. This is the default used by most Digital Audio Workstations when a new session is created.
- A the human engineered mix, taken from the study in Chapter 4. The highest ranked mix from the listening test was used.
- An equal loudness mix, as shown by Mansbridge et al. (2012b) to be a suitable automatic mix process. This was modified to be a time-invariant model.
- A mix generated using our proposed genetic algorithm.

The mixes were all evaluated objectively by measuring their masking levels for each track to determine the total inter-track masking level. To perform subjective evaluation, a listening test was also performed to confirm if the masking minimisation technique is actually preferred by listeners as a target.

The genetic algorithm ran 100 iterations with a population size of 32 and a mutation rate of 0.100 (10%). The chromosomes represent a the gain value for each track and are held as a vector of real numbers of size N. The gains are limited to a range of -96dB and $+$24dB, giving a suitable range similar to a mixing console or digital audio workstation.

The short-term loudness (STL) vectors for each track are computed ahead of time. Each track is normalised to 70dBSPL (RMS) to simulate their free-field loudness when computing the STL_n. With the initial population and STL vectors computed, the algorithm can then perform the following steps for each iteration:

1. Evaluate the cost of each chromosome, see section 6.2.1.
2. Rank each chromosome based on its cost value.
3. Perform the cross-over and mutations to repopulate the next generation, see section 2.5.3.
4. Decrement the number of iterations by 1, $M = M - 1$. If $M < 0$ then exit.

6.2.3 Results

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human	Anchor 1	Anchor 2
Drums	-4.54	-5.32	1.36	1.30	-3.56	-∞
Bass	-4.54	-4.04	-7.45	-10.71	-3.56	-3.19
Electric Guitar	-4.54	-8.67	-3.42	-4.71	-∞	-∞
Vocal	-4.54	-3.28	-4.58	-1.71	-3.56	-3.19

Table 6.1: Mix gains for the song 'I'm Alright' for the four mix evaluations and two hidden anchors.

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human	Anchor 1	Anchor 2
Drums	-9.20	-13.25	-5.53	-14.27	-9.08	-∞
Electric Guitar	-9.20	-17.11	-3.47	-2.27	-9.08	-8.68
Lead Guitar	-9.20	-20.00	-5.80	-5.27	-∞	-∞
Bass	-9.20	-8.30	-12.99	-11.27	-9.08	-8.68

Table 6.2: Mix gains for the song 'Left Blind' for the four mix evaluations and two hidden anchors.

The four mixes under evaluation give four different possible balance mixes an engineer could achieve. The four mixes are Unity, where all the track gains are set to 0dB, Genetic Algorithm, EBU based mix from Mansbridge et al. (2012b) and Human Mix. The unity mix is where all the gains are set to 0dB, indicating no boosting or attenuation from the recorded materials. The EBU based mix from Mansbridge et al. (2012b) takes each track and passes it through the ITU-R BS.1770 loudness filter (International Telecommunication Union, 2011). This

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human	Anchor 1	Anchor 2
Bass	-6.41	-5.38	-5.53	-10.97	-∞	-∞
Lead Vocals	-6.41	-13.17	-3.47	-4.53	-0.54	-∞
Piano	-6.41	-13.25	-5.80	0.27	-0.54	2.79
Drums	-6.41	-10.51	-12.99	-0.42	-0.54	2.79

Table 6.3: Mix gains for the song 'Sleigh Ride' for the four mix evaluations and two hidden anchors.

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human	Anchor 1	Anchor 2
Synthesiser	-0.40	-1.44	3.25	-6.48	-∞	-∞
Acoustic Guitar	-0.40	-1.07	-2.48	-0.48	0.23	-∞
Drums	-0.40	-3.05	3.47	2.52	0.23	3.94
Lead Vocals	-0.40	-1.25	-4.69	-0.48	0.23	3.94

Table 6.4: Mix gains for the song 'The English Actor' for the four mix evaluations and two hidden anchors.

gives a loudness score in LUFS (Loudness Unit Full Scale). The recommendation document R128 specifies that a good loudness score for broadcast material is -23LUFS (European Broadcast Union, 2014). So each track's gain is set to equal -23LUFS over the program material. The human mix is the highest scoring mix from each of the four materials presented in Chapter 4. The two anchors are also unity mixes, but with one and two tracks muted respectively. Tables 6.1, 6.2, 6.3 and 6.4 give the gains for each of the four mixes, plus anchors, for the four songs under test. All the gains are the normalised mix values such that the LUFS measurement is -23LUFS to allow for equal loudness between the tracks when performing the listening test.

Listening Test

The listening test was conducted using the Web Audio Evaluation Toolbox (Jillings et al., 2015). The toolbox allows for the creation of easy listening tests to be deployed on the web or in laboratory conditions. The toolbox has several different user interfaces included, along with multiple standardisation factors such as automatic loudness normalisation, randomisation and data collection scripts. The listening test was chosen to be based upon the the Multiple Stimulus and Hidden Reference testing standard (MUSHRA) (International Telecommunication Union, 2011). Unlike in the ideal MUSHRA standard, there is no appropriate reference mix. This is because for the genetic algorithm auto-mixer, there is no known ideal mix. Therefore the normal MUSHRA scale is not appropriate, and a scale which has an open-top for interpretation should be used. The Comparison Category Rating (CCR) Method has been used in previous studies for such a known problem. Naderi et al. (2021) uses the CCR due to the fact the scale is suitable to systems that improve the quality of the input. In their study they were investigating the validity of using croud sourced participants versus laboratory studies. Just like previous studies by Schoeffler et al. (2013) and Cartwright et al. (2016) the performance of lab and croud sourced studies can show the same level of performance so long as there is appropriate filtering, training and pre-processing of the tests. The CCR scale is defined as follows (International Telecommunication Union, 1996):

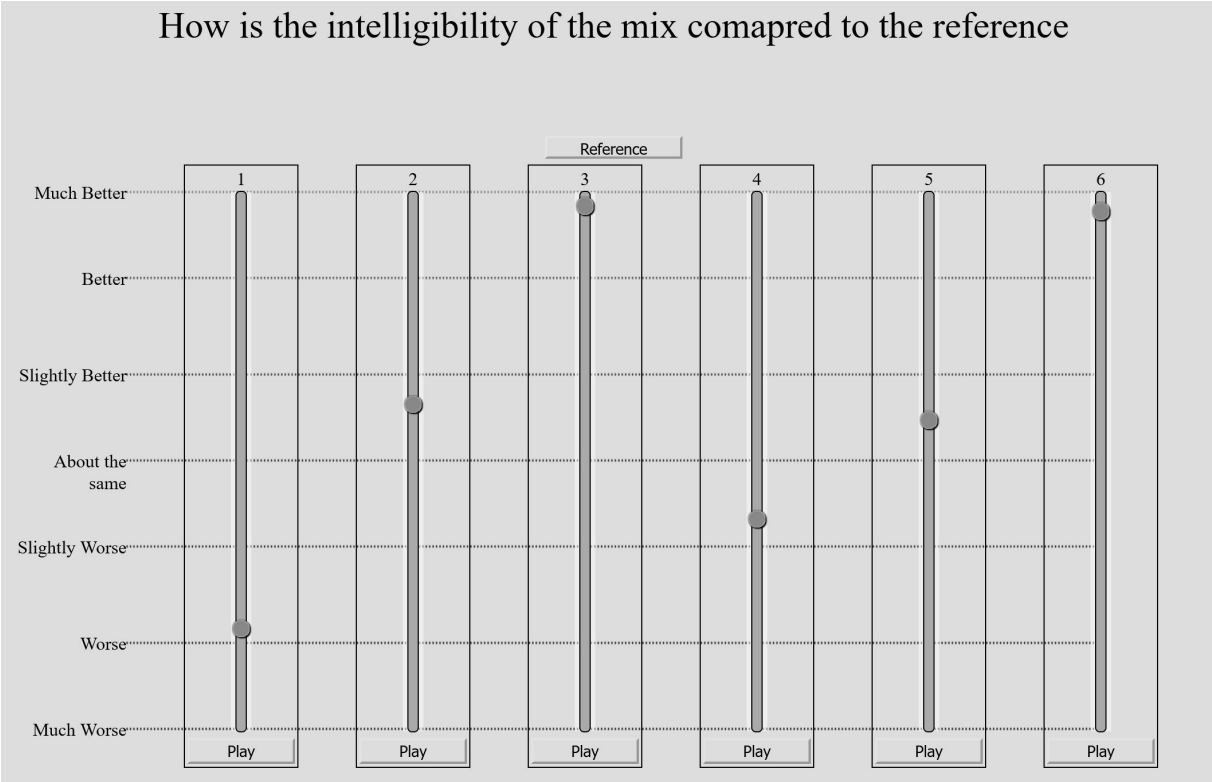


Figure 6.2: The listening test interface for the evaluation of the genetic mixes, using the Web Audio Evaluation Toolbox (Jillings et al., 2015).

- Much Better
- Better
- About the same
- Worse
- Much Worse

This scale, when compared with a reference, allows for the subject to determine if a song has improved above the reference. This then allows for an inferred relationship between the tracks where they can still be placed above or below each other in terms of their mix performance.

The question being posed is also important to gathering the right form of information. The aim of the balance mix process is to make the mix intelligible and set up the foundation for how the mix is to progress (Izhaki, 2012). Therefore the question posed instead was "How is the intelligibility of the mix compared to the reference?".

A total of 27 users participated in the study, 9 of who completed all 5 pages. Across the 27 subjects, there was 55 pages completed giving a total of 359 fragment evaluations made. The average test time per page was 2 min 17 seconds. Table 6.5 gives the total times for each of the pages under test. The training page is always shown first to the users, and this correctly shows a larger amount of time spent. This would indicate

Song	Duration	# subjects	# fragments
Training	3 min 34 s	13	5
The English Actor	1 min 26 s	11	7
Left Blind	1 min 46 s	9	7
I'm Alright	2 min 8 s	12	7
Sleigh Ride	2 min 9 s	10	7

Table 6.5: The average time spent on each page along with the total number of subject submissions after filtering

.

Years Experience	Number of Subjects
0	1
1	1
2	4
3	5
4	4
5	1
More than 5	10

Table 6.6: The years of experience as reported by the listeners in the test survey

.

the subjects are learning the interface at this point, and when they engage with the test each page takes only a few minutes to complete.

Tables 6.6 to 6.9 give the results of the pre-test survey. The individuals removed from the study based on the survey were individuals with known hearing impairments (2 subjects), individuals using personal hi-fi's in untreated rooms or not with sufficient frequency range (3) and finally individuals who had said they had already completed this listening test (2).

As can be seen, most of the participants in this study have fewer years of experience mixing, most being under 5 years. This is because most of the subjects were acquired from the Sound Engineering undergraduate degree course at Birmingham City University. This should not change the results because, whilst musically untrained, the subjects are not being asked to perform or complete mixing tasks, but to listen to the quality of the mix

Experience in Listening Tests	Number of Subjects
Yes	20
No	7

Table 6.7: The number of subjects who have performed a listening study before

.

Listening Environment	Number of Subjects
Studio	1
Private Space (Hi Fi)	9
Headphones	17

Table 6.8: The declared environment of the participants

.

Headphone Types	Number of Subjects
Supra-aural	10
Circumaural Open Backed	4
Circumaural Closed Backed	2
In-ears	1

Table 6.9: The headphone type of the 16 headphone using subjects

.

presented. Most participants used their headphones, and most seem to have higher quality headphones than in ears. Only one participant took place in a treated studio environment.

Figure 6.4 shows the results for the listening test across all pages. The genetic algorithm mix did not perform as strongly when combined, which is most likely due to the poor performance experienced in two of the listening test conditions. The Wilcoxon rank sum test results in table 6.15 confirm it was significantly less than the Human, EBU loudness based mix by Mansbridge et al. (2012b) and unity mixes. However the results are un-related to each other due to the varying scales and potential failure of the test environment. The first action is to removes any instances where the 'Unity' mix was ranked as the highest or lowest performing mix, as this would indicate that a participant was either unclear on the test or unable to distinguish the results properly. Table 6.14 shows the 6 removed entries from the combined data with the combined data plotted in figure 6.5. These 6 submissions also indicate times when the listener could not identify the hidden reference correctly, either over ranking or under-ranking the 'Unity' position.

Track	Anchor 1	Anchor 2	Mansbridge	Genetic	Human	Unity
Anchor 1	1.000	<0.001	<0.001	<0.001	<0.001	<0.001
Anchor 2	<0.001	1.000	<0.001	0.014	0.003	<0.001
Mansbridge	<0.001	<0.001	1.000	0.184	0.627	0.554
Genetic	<0.001	0.014	0.184	1.000	0.456	0.137
Human	<0.001	0.003	0.627	0.456	1.000	1.000
Unity	<0.001	<0.001	0.554	0.137	1.000	1.000

Table 6.10: Wilcoxon ranked sum test for the listening test reults of 'I'm Alright' in figure 6.3a.

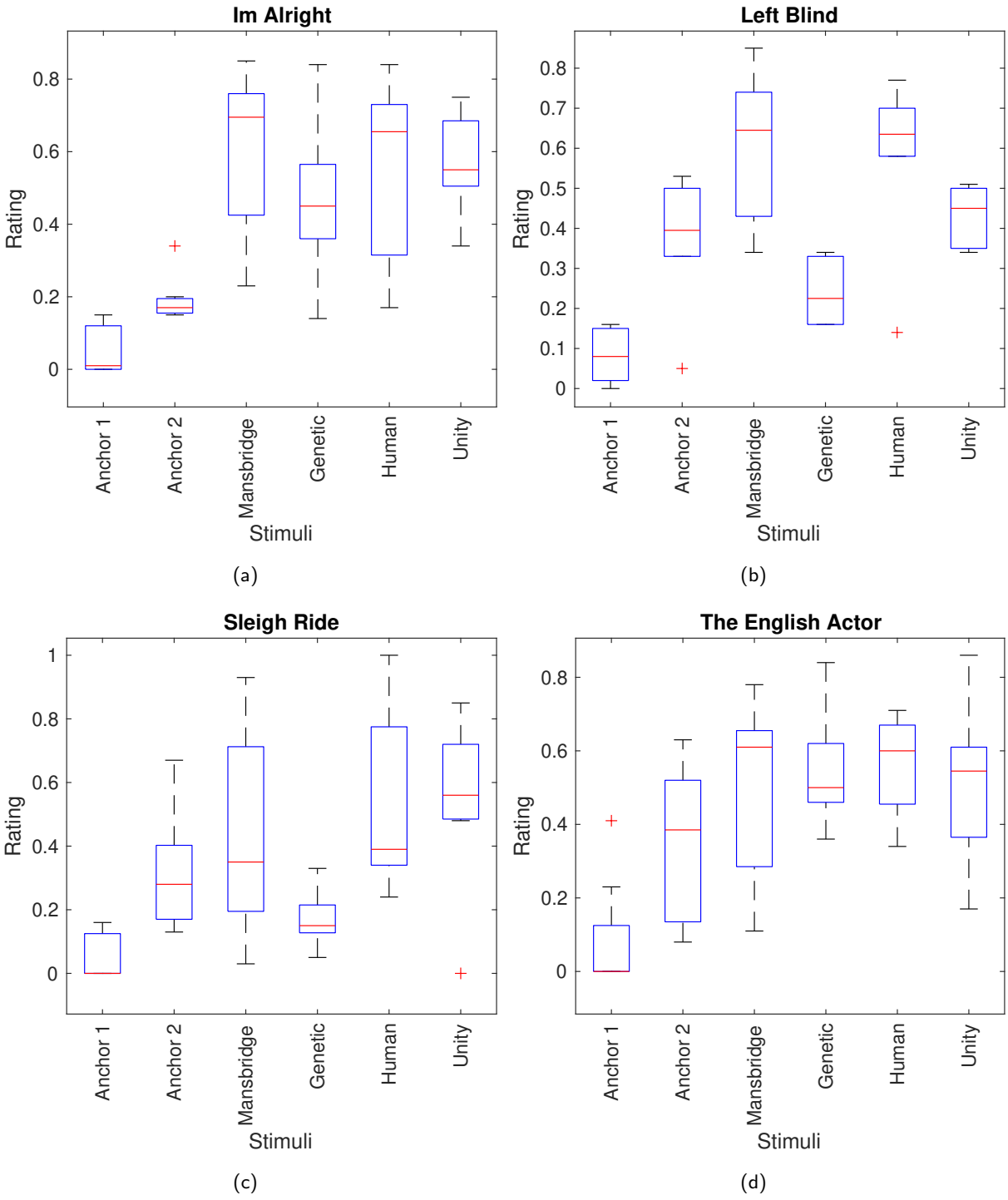


Figure 6.3: The box plot of the results for the four mixes under evaluation for 'I'm Alright' 6.3a, 'Left Blind' 6.3b, 'Sleigh Ride' 6.3c and 'The English Actor' 6.3d.

Track	Anchor 1	Anchor 2	Mansbridge	Genetic	Human	Unity
Anchor 1	1.000	0.026	0.002	0.006	0.008	0.002
Anchor 2	0.026	1.000	0.071	0.123	0.041	0.561
Mansbridge	0.002	0.071	1.000	0.004	0.853	0.139
Genetic	0.006	0.123	0.004	1.000	0.061	0.004
Human	0.008	0.041	0.853	0.061	1.000	0.065
Unity	0.002	0.561	0.139	0.004	0.065	1.000

Table 6.11: Wilcoxon ranked sum test for the listening test reults of 'Left Blind' in figure 6.3b.

Track	Anchor 1	Anchor 2	Mansbridge	Genetic	Human	Unity
Anchor 1	1.000	0.004	0.006	0.024	0.001	0.010
Anchor 2	0.004	1.000	0.513	0.103	0.137	0.097
Mansbridge	0.006	0.513	1.000	0.104	0.513	0.644
Genetic	0.024	0.103	0.104	1.000	0.001	0.024
Human	0.001	0.137	0.513	0.001	1.000	0.777
Unity	0.010	0.097	0.644	0.024	0.777	1.000

Table 6.12: Wilcoxon ranked sum test for the listening test reults of 'Sleigh Ride' in figure 6.3c.

Track	Anchor 1	Anchor 2	Mansbridge	Genetic	Human	Unity
Anchor 1	1.000	0.006	0.002	$<$0.001	$<$0.001	0.001
Anchor 2	0.006	1.000	0.137	0.087	0.040	0.129
Mansbridge	0.002	0.137	1.000	1	0.663	0.739
Genetic	$<$0.001	0.087	1.000	1	0.627	0.777
Human	$<$0.001	0.040	0.663	0.627	1.000	0.487
Unity	0.001	0.129	0.739	0.777	0.487	1.000

Table 6.13: Wilcoxon ranked sum test for the listening test reults of 'The English Actor' in figure 6.3d.

Song	Unity	Genetic	Mansbridge et al. (2012b)	Human	Anchor 1	Anchor 2
I'm Alright	0.75	0.14	0.33	0.27	0.15	0.01
Sleigh Ride	0.74	0.33	0.93	0.67	0.28	0.00
Sleigh Ride	0.85	0.15	0.15	1.00	0.35	0.15
The English Actor	0.86	0.43	0.17	0.51	0.63	0.00
The English Actor	0.17	0.36	0.40	0.64	0.43	0.41
The English Actor	0.23	0.70	0.40	0.71	0.58	0.23

Table 6.14: Summary of the removed entries from the listening test data based on the improper usage of the scales, where the 'Unity' mix was not correctly identified as being the same as the reference.

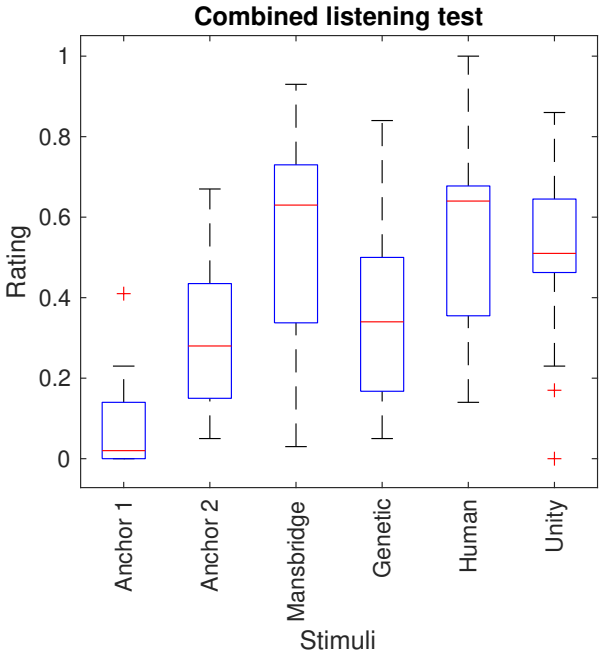


Figure 6.4: The box plot of the results for all the tests combined

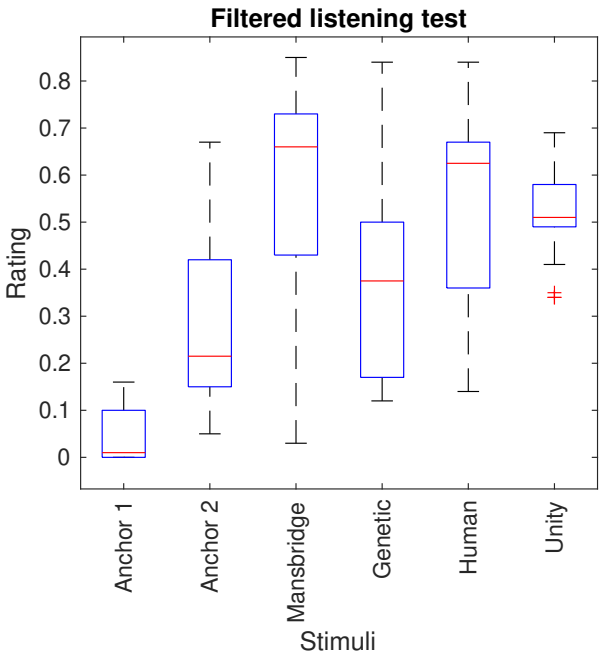


Figure 6.5: The box plot of the results for all the tests combined filtered for when the Unity mix was ranked highest or lowest.

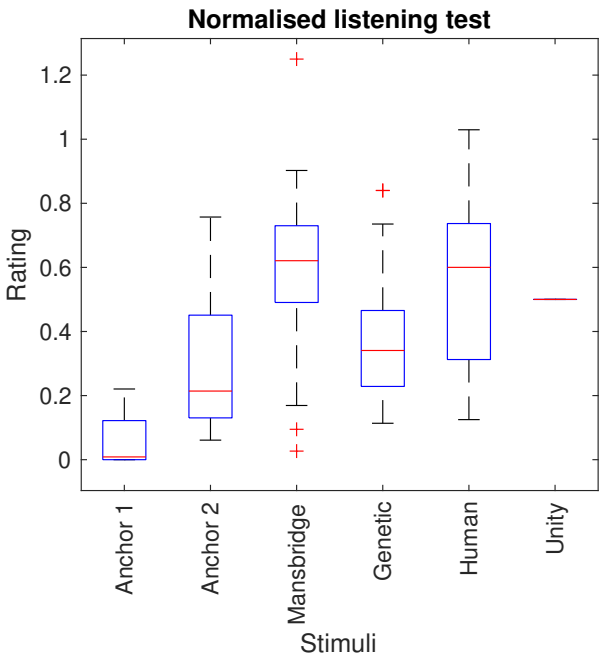


Figure 6.6: The box plot of the results for all the tests combined filtered and unity was normalised to equal 0.5

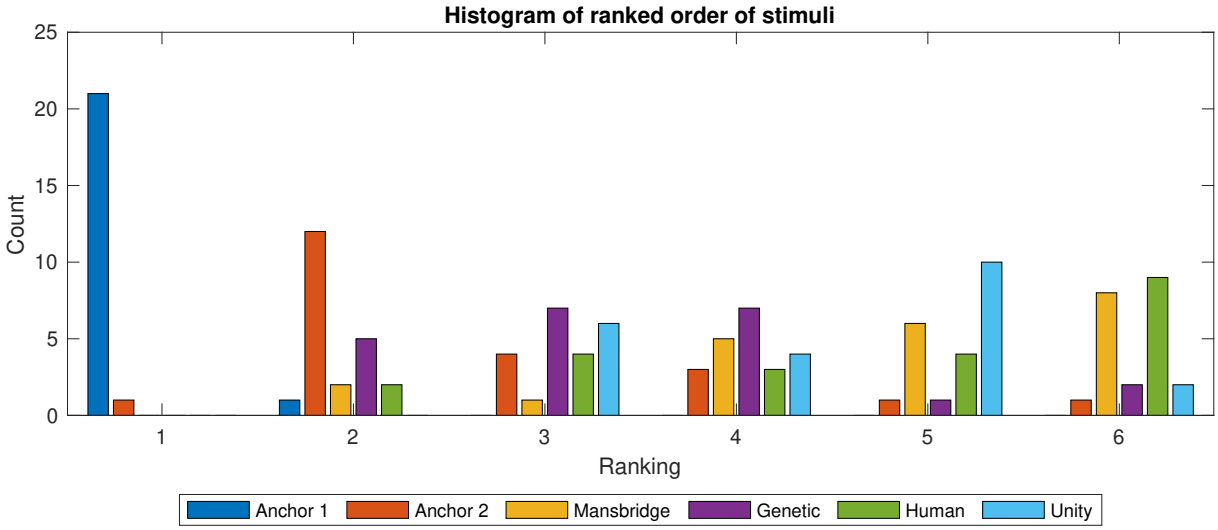


Figure 6.7: Histogram ranking of the 6 mixes by their individual ranking scores as given by each subject

Track	Anchor 1	Anchor 2	Mansbridge	Genetic	Human	Unity
Anchor 1	1.000	<0.001	<0.001	<0.001	<0.001	<0.001
Anchor 2	<0.001	1.000	<0.001	0.262	<0.001	<0.001
Mansbridge	<0.001	<0.001	1.000	0.008	0.895	0.489
Genetic	<0.001	0.262	0.008	1.000	0.002	0.002
Human	<0.001	<0.001	0.895	0.002	1.000	0.362
Unity	<0.001	<0.001	0.489	0.002	0.362	1.000

Table 6.15: Wilcoxon ranked sum test for the combined listening test results in figure 6.4.

Track	Anchor 1	Anchor 2	Mansbridge	Genetic	Human	Unity
Anchor 1	1.000	<0.001	<0.001	<0.001	<0.001	<0.001
Anchor 2	<0.001	1.000	<0.001	0.093	<0.001	<0.001
Mansbridge	<0.001	<0.001	1.000	0.006	0.605	0.046
Genetic	<0.001	0.093	0.006	1.000	0.016	0.004
Human	<0.001	<0.001	0.605	0.016	1.000	0.274
Unity	<0.001	<0.001	0.046	0.004	0.274	1.000

Table 6.16: Wilcoxon ranked sum test for the filtered combined listening test results in figure 6.5.

Whilst each of these will not have a significant impact on the data being removed, it will show the importance of checking the data for erroneous results. Table 6.16 confirms there is no major change in the statistical relationship between the results. Because the scale indicated the value 0.5 should equal a mix which was perceptually the same as the reference, and with the 'Unity' mix being the hidden reference, this mix should have been around the 0.5 mark. The results were then normalised to make the 'Unity' mix equal to 0.5 Figure 6.6 goes one step further and normalises the data such that the 'Unity' mix is equal to 0.5. This normalisation stage will give a more representative score by narrowing the spread of the system to account for noise generated by subjects inadvertently placing the reference mix higher or lower, thus biasing their axis ranges. There is no statistical difference to the distributions, with genetic algorithm failing the null-hypothesis test of the Wilcoxon rank sum test compared to the other 3 methods under test (excluding the anchors).

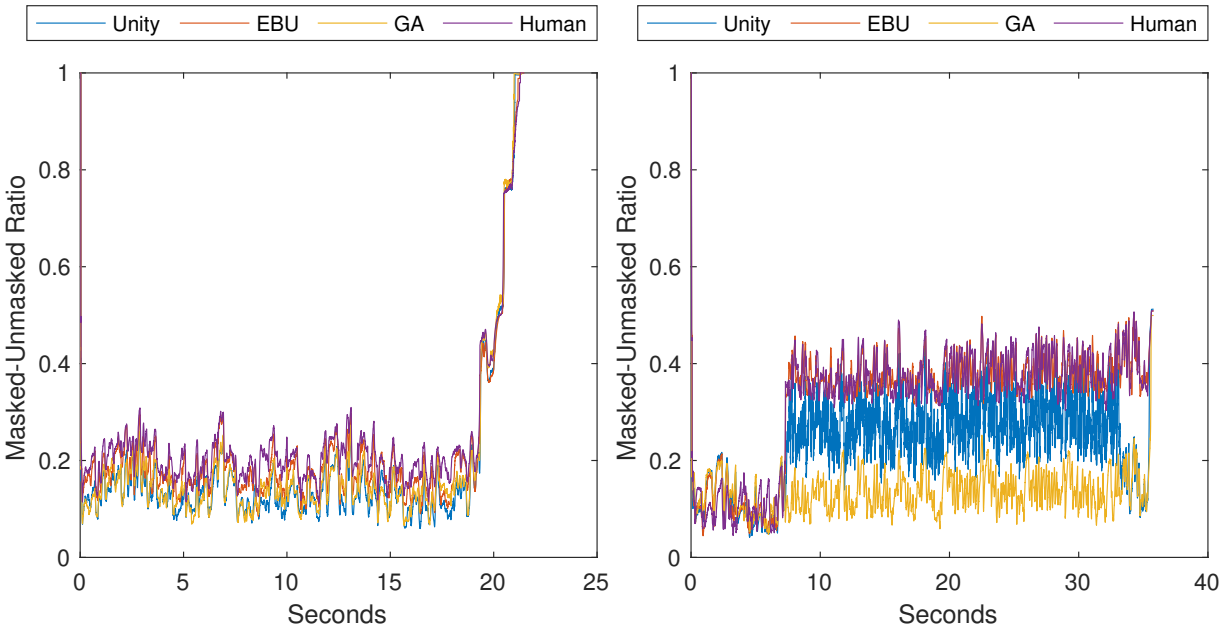
Figure 6.7 gives the data as a histogram showing the ranked position of each of the mixes based on the subject page ranking. Clearly the two anchor mixes are positioned heavily at the bottom, often scoring 1 and 2 rank positions. Then the Genetic Algorithm mix scores 2 and 3 rank positions, with the Unity, Human and EBU mixes scoring average higher ranking positions respectively.

6.2.4 Discussion

This section will present the the analysis of the results from the listening test when combined with the masked to unmasked ratio of each song. This will explain the performance of the system relative to the existing solutions based on the subjective results. This section then discusses the effect of the cost function and how this was improved to the final version. Finally, the section will present the efficiency of the algorithm from a computational point of view and how the system can be improved to increase the performance.

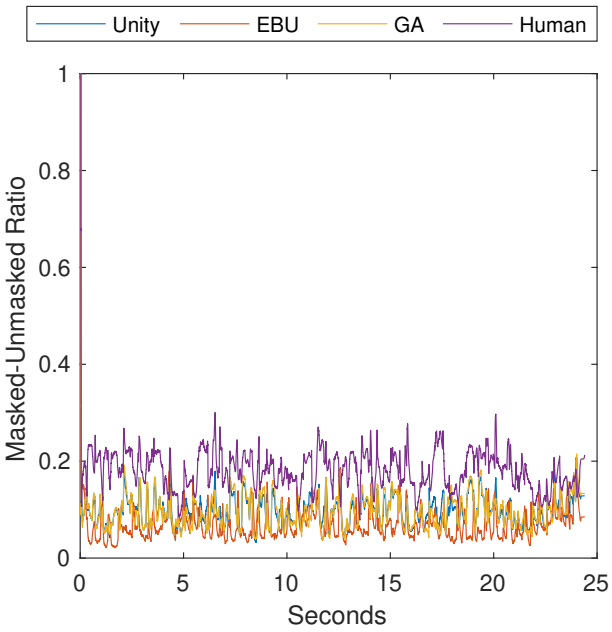
Subjective Performance

The song 'I'm Alright' and 'The English Actor' both had the Genetic Algorithm mix performing comparably to the other mixes. Tables 6.10 and 6.13 shows the results of the Wilcoxon rank sum test p-values (Wilcoxon, 1945). For 'I'm Alright' the four mixes under test perform similarly well with no clear significan difference betweent the four, and all performing significantly better than the anchor. Whilst 'Sleigh Ride' and 'Left Blind' had the Genetic Algorithm performing significantly worse than the other mixes, with the p-values in tables 6.12 and 6.11 showing the p-value scores. Whilst not significantly worse than the others in these two, it is clear the

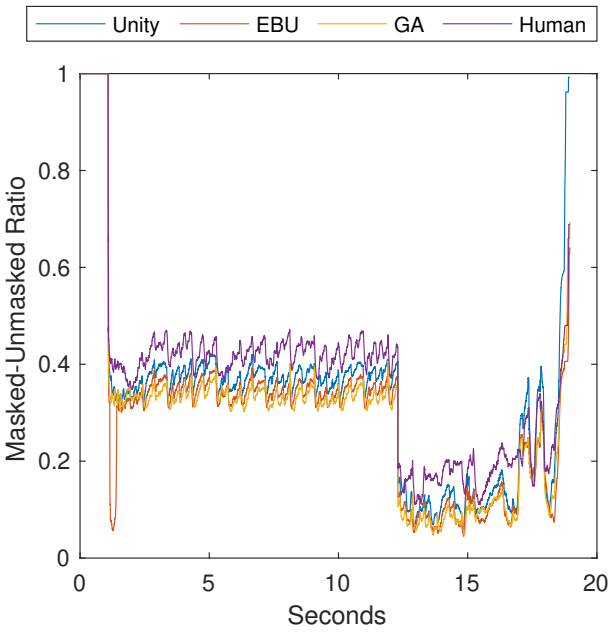


(a) MUR over time of the mixes for 'I'm Alright'.

(b) MUR over time of the mixes for 'Left Blind'.



(c) MUR over time of the mixes for 'Sleigh Ride'.



(d) MUR over time of the mixes for 'The English Actor'.

Figure 6.8: The Masked-Unmasked ratio (MUR) of the four songs. A higher MUR indicates more masking. In 3 of the experiments, the GA outperformed all other mixes.

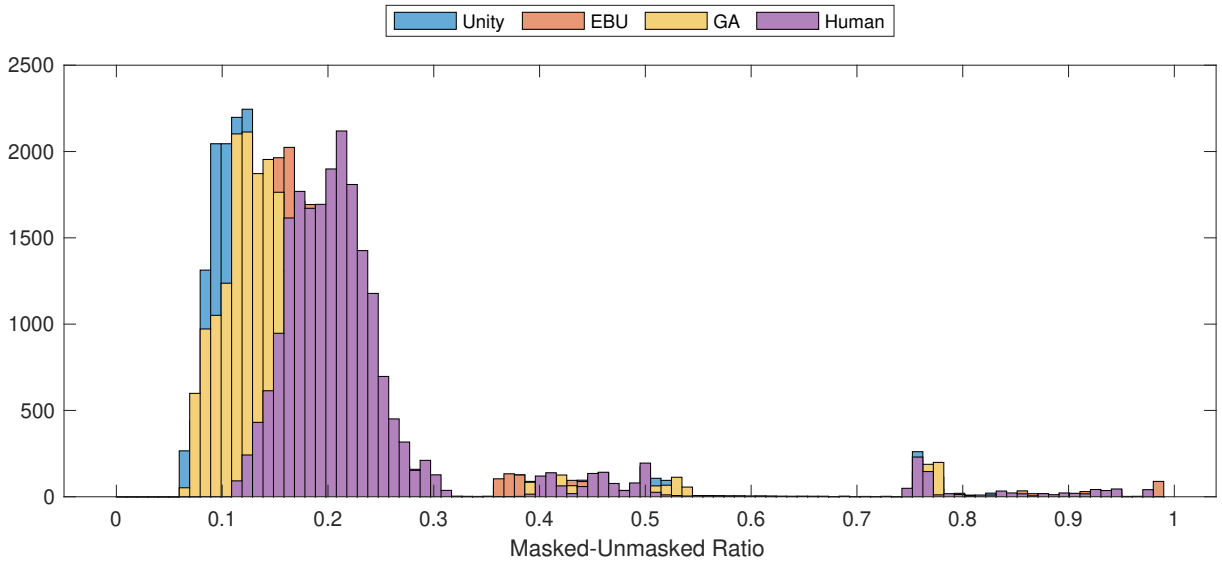


Figure 6.9: Histogram of the Masked-to-Unmasked ratio of the song 'I'm Alright' in Figure 6.8a.

Mix	Unity	Mansbridge et al. (2012b)	Genetic	Human
Median	0.1300	0.1743	0.1431	0.2075
Mean	0.1831	0.2202	0.1937	0.2471
Standard Deviation	0.1757	0.1631	0.1780	0.1557

Table 6.17: Median, Mean and Standard Deviations for the Masked-to-Unmasked ratio of the song 'I'm Alright' in Figure 6.9.

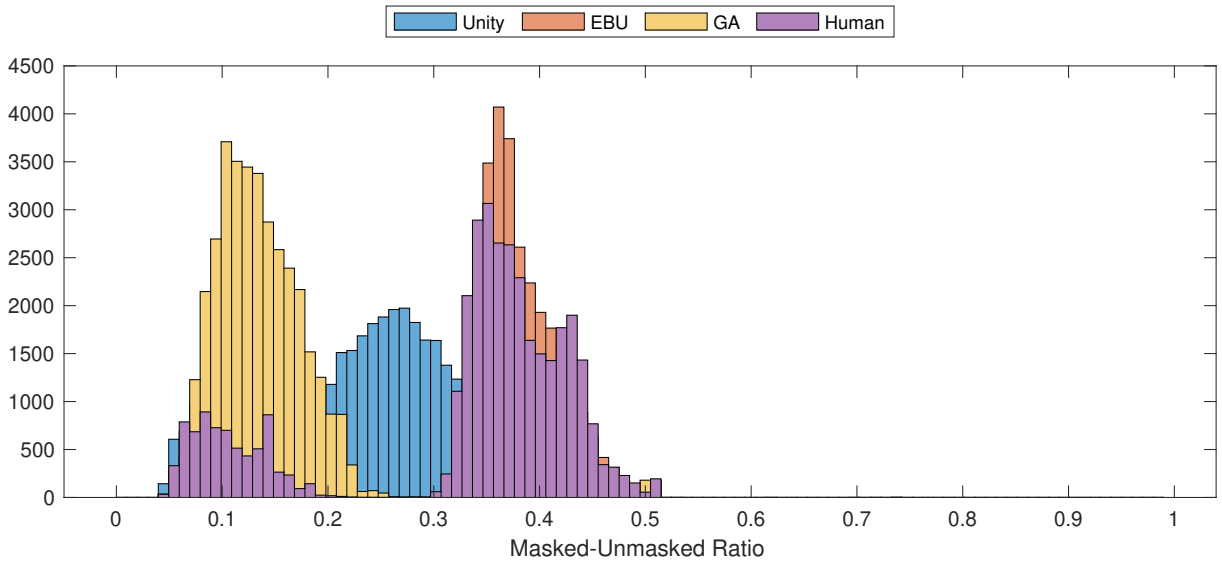


Figure 6.10: Histogram of the Masked-to-Unmasked ratio of the song 'Left Blind' in Figure 6.8b.

Mix	Unity	Mansbridge et al. (2012b)	Genetic	Human
Median	0.2507	0.3658	0.1307	0.3612
Mean	0.2390	0.3288	0.1372	0.3275
Standard Deviation	0.0912	0.1182	0.0536	0.1194

Table 6.18: Median, Mean and Standard Deviations for the Masked-to-Unmasked ratio of the song 'Left Blind' in Figure 6.10.

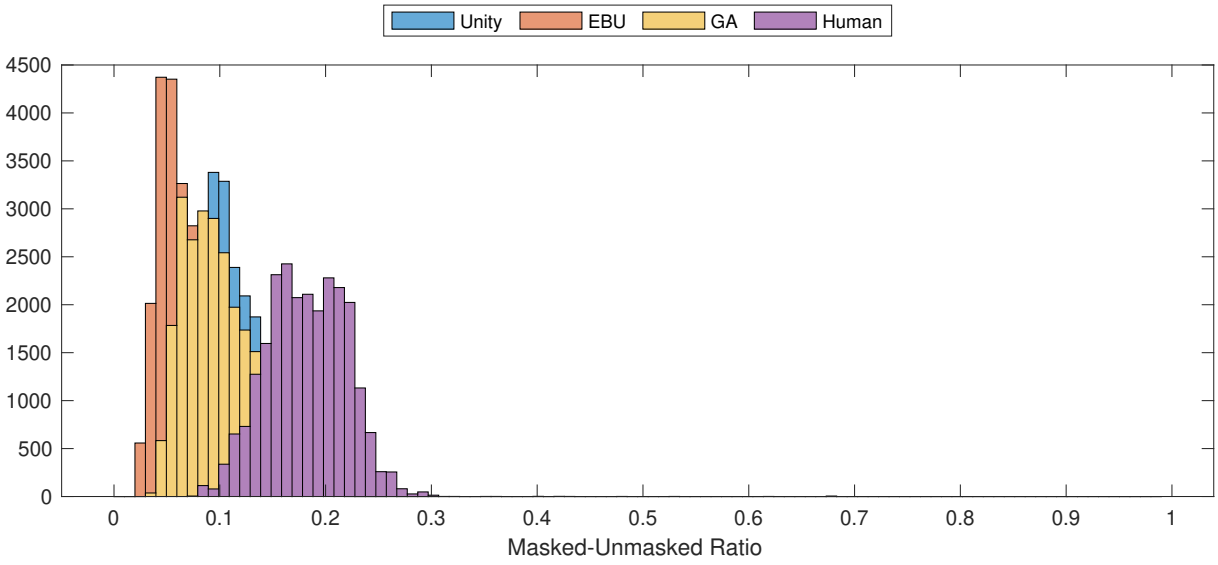


Figure 6.11: Histogram of the Masked-to-Unmasked ratio of the song 'Sleigh Ride' in Figure 6.8c.

Mix	Unity	Mansbridge et al. (2012b)	Genetic	Human
Median	0.0967	0.0620	0.0930	0.1811
Mean	0.0992	0.0705	0.0980	0.1828
Standard Deviation	0.0428	0.0452	0.0450	0.0481

Table 6.19: Median, Mean and Standard Deviations for the Masked-to-Unmasked ratio of the song 'Sleigh Ride' in Figure 6.11.

Mix	Unity	Mansbridge et al. (2012b)	Genetic	Human
Median	0.3605	0.3300	0.3243	0.4043
Mean	0.3436	0.3056	0.3038	0.3788
Standard Deviation	0.2103	0.2059	0.2058	0.1907

Table 6.20: Median, Mean and Standard Deviations for the Masked-to-Unmasked ratio of the song 'The English Actor' in Figure 6.12.

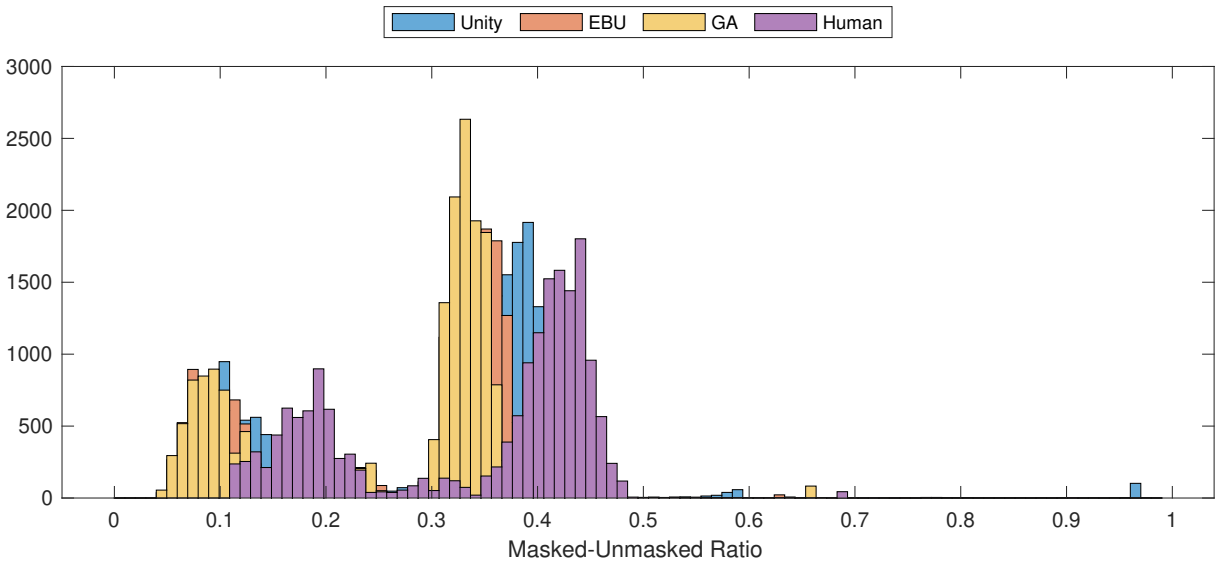


Figure 6.12: Histogram of the Masked-to-Unmasked ratio of the song 'The English Actor' in Figure 6.8d.

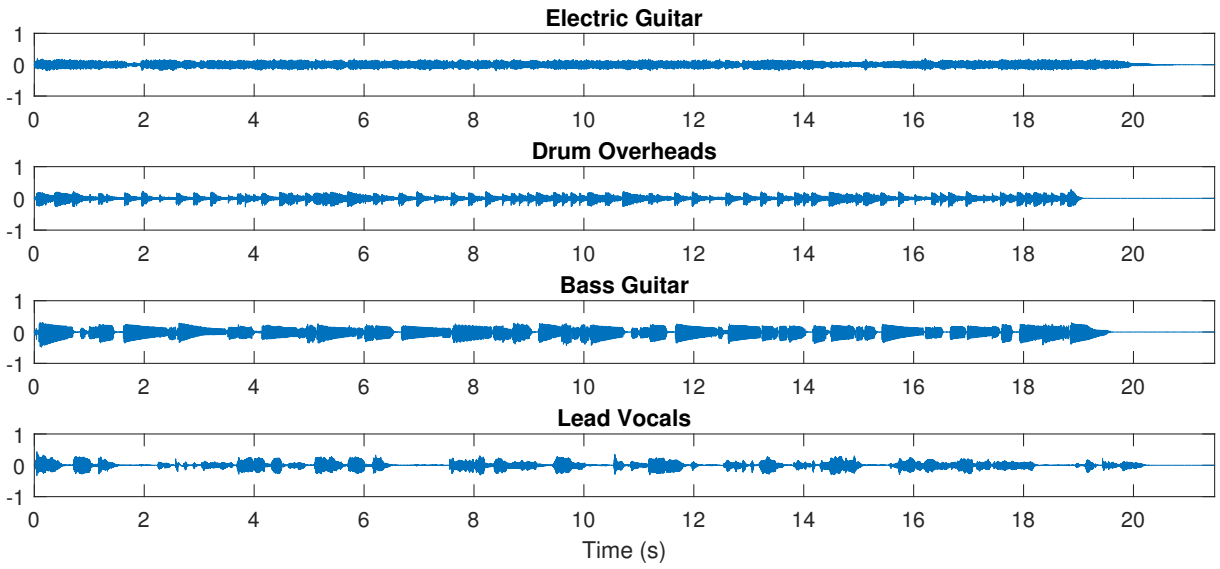


Figure 6.13: Timeline of the tracks used in the song 'I'm Alright'

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human
Electric Guitar	-6.9055	-11.0313	-5.7800	-7.0752
Drums	-11.6840	-12.4576	-5.7800	-5.8437
Bass Guitar	-2.8762	-2.3689	-5.7800	-9.0460
Lead Vocals	-5.7441	-4.4849	-5.7800	-2.9138

Table 6.21: The Relative LUFS of the song 'I'm Alright' compared to the end mix LUFS after normalisation.

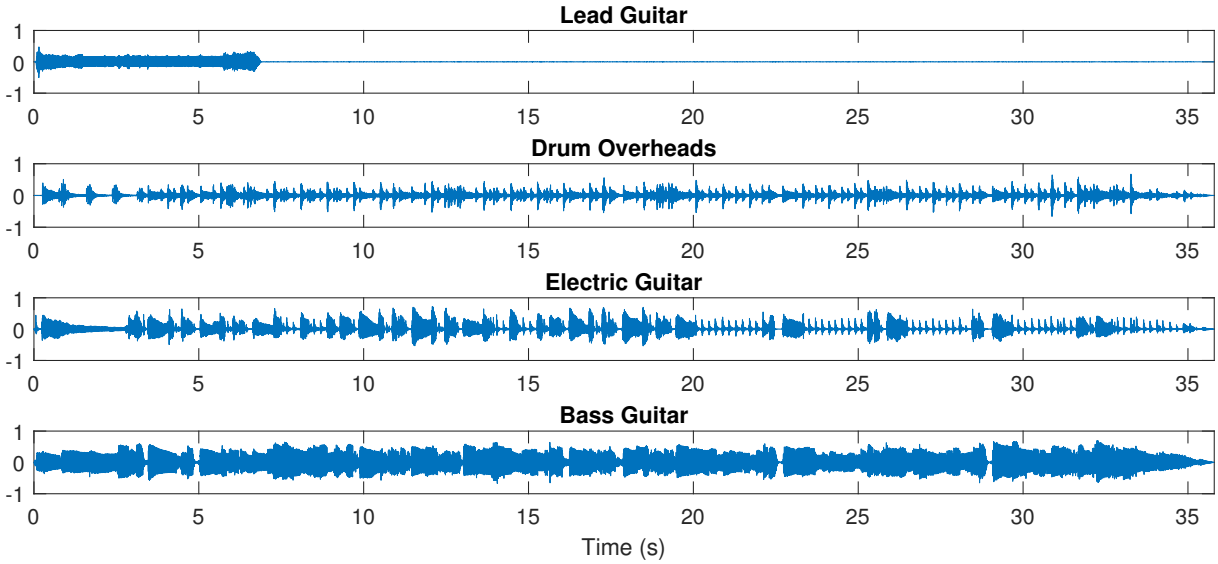


Figure 6.14: Timeline of the tracks used in the song 'Left Blind'

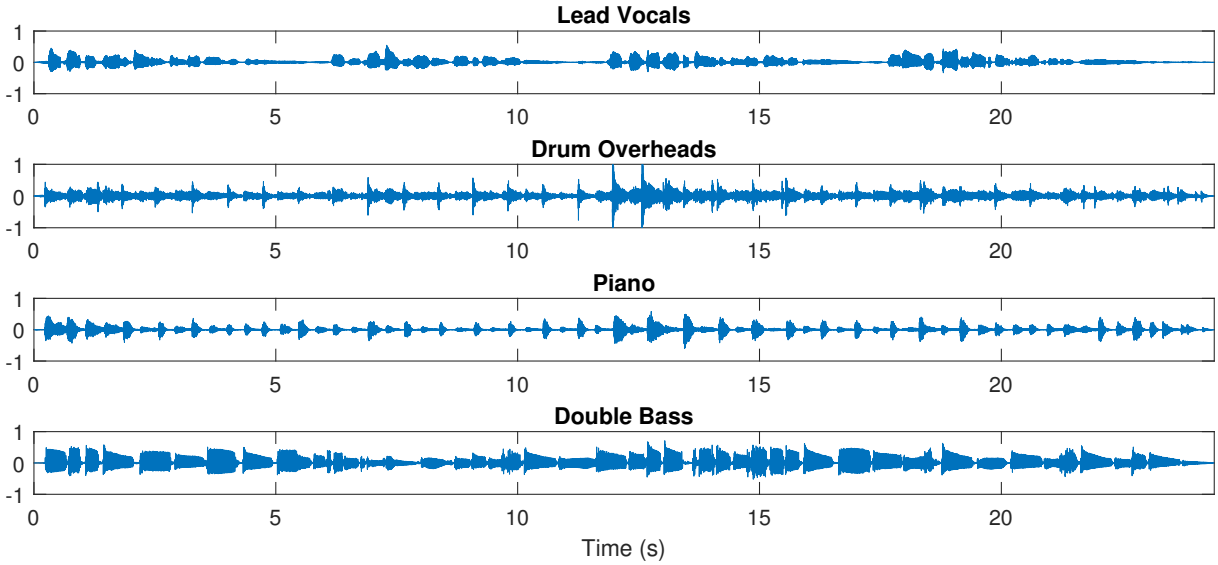


Figure 6.15: Timeline of the tracks used in the song 'Sleigh Ride'

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human
Lead Guitar	-8.3570	-19.1547	-4.9509	-4.4228
Drums	-10.6812	-18.5890	-4.9509	-3.7470
Electric Guitar	-8.6201	-12.6714	-4.9509	-13.6859
Bass Guitar	-1.1617	-0.2670	-4.9509	-3.2275

Table 6.22: The Relative LUFS of the song 'Left Blind' compared to the end mix LUFS after normalisation.

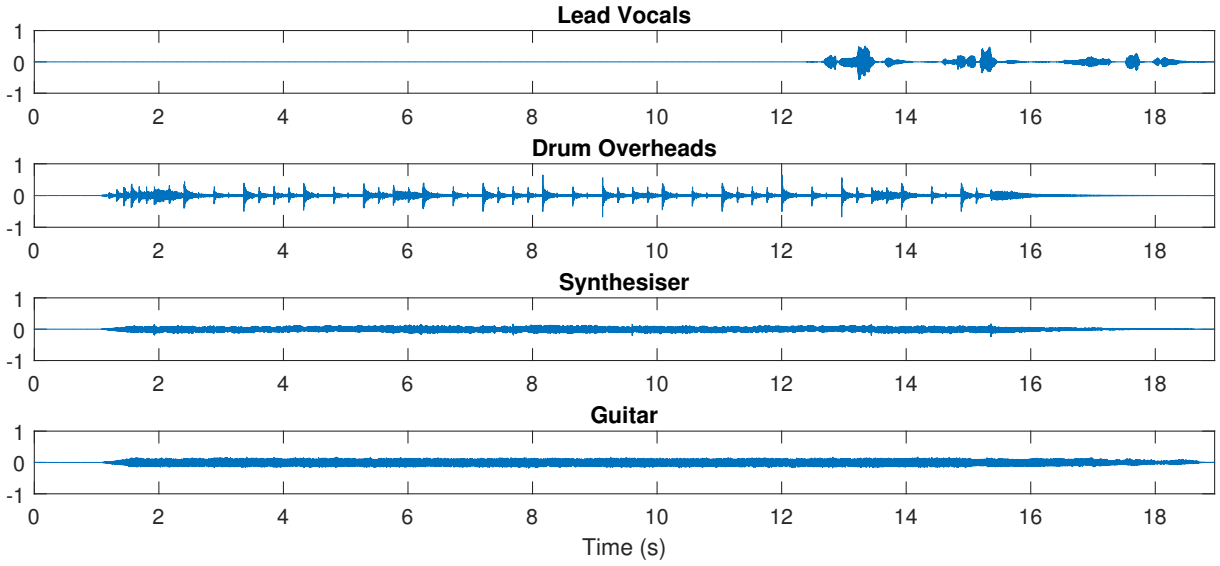


Figure 6.16: Timeline of the tracks used in the song 'The English Actor'

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human
Double Bass	-1.3622	-0.3388	-5.9229	-6.9154
Lead Vocals	-7.8020	-14.5657	-5.9229	-1.3651
Piano	-12.5984	-19.4436	-5.9229	-15.1616
Drums	-11.9096	-16.0174	-5.9229	-8.4627

Table 6.23: The Relative LUFS of the song 'Sleigh Ride' compared to the end mix LUFS after normalisation.

Track	Unity	Genetic	Mansbridge et al. (2012b)	Human
Synthesiser	-8.5103	-9.5528	-4.8584	-14.5942
Acoustic Guitar	-2.7774	-3.4474	-4.8584	-2.8613
Drums	-8.7317	-5.2792	-4.8584	-5.8156
Lead Vocals	-0.5706	-1.4217	-4.8584	-0.6544

Table 6.24: The Relative LUFS of the song 'The English Actor' compared to the end mix LUFS after normalisation.

mixes are not significantly better than the existing methodology by (Mansbridge et al., 2012b), even though it was modified to be a steady state system, or providing a Unity mix.

To understand the differences, the MUR was calculated over time for each of the four mixes under test to examine if there was a failure of either the genetic algorithm or the principle under test. To achieve this the MUR was calculated over time rather than as an average of the entire signal. These are plotted in figure 6.8. To help further analyse these signals, a histogram of the plots are taken, presented in figures 6.9, 6.10, 6.11 and 6.12.

As can be seen the four mixes all share a similar distribution of masked to unmasked, and the 'Unity' mix gives the lowest median Masked to Unmasked Ratio score. Table 6.17 shows the statistical measurements of the mix. The Human mix scores a mean MUR of 0.2471, showing a significant amount of the energy in this song is mixed. The lowest score in the song 'I'm Alright' came from the Unity mix at 0.1831, with the Genetic mix close behind at 0.1937. This can be attributed to the higher relative loudness of the vocal track in the mix space for 'I'm Alright'. The gains in Table 6.1 show the Vocal track gain for the 'Genetic' mix is -3.28dB, higher than any other track in that mix. It is already known that higher vocal mix gains produce a perceptually better quality mix (Wilson and Fazenda, 2015b; De Man and Reiss, 2013a). The relative loudness of the tracks for each mix of 'I'm Alright' is presented in Table 6.21. This shows that the vocal track is placed higher in the mix for the 'Genetic' and 'Human' mixes. This can also lead to a perceived better intelligibility, since the vocals themselves convey information and a masked, muted or buried vocal track will be considered a unintelligible. 'The English Actor' also shows this same approach, with the vocal mix being perceptually quite high in the mix. Second to this, in 'The English Actor' the vocals are only present in the last 5 seconds of the song. Figure 6.16 shows this in the timeline plot with the lead vocals only present in the tail of the song, when other tracks are fading out. This leads to a situation where the song is mostly instrumental in nature. The relative loudness of the tracks for each mix of 'The English Actor' is presented in Table 6.24. Again the lead vocals are relatively loud, at only -1.42 LUFS compared to the mix loudness and clearly above the other tracks. Interestingly, the other mixes also placed the vocals quite high in the mix which would explain the relative lack of separation between the mix scores.

Only biasing by the instrumental mix, removing any vocals, does not explain all the performance issues of the mix generator. 'Left Blind' has no vocal mix, as shown in Figure 6.14. The three main tracks and one partial lead track, only present for the first 6.5 seconds, created another problematic mixing scenario for the multi track mix. In this cases, the 'Human' and Mansbridge et al. (2012b) mixes both did the best, whilst the 'Genetic' mix scored extremely poorly. Looking directly at the relative loudness of each track in Table 6.22, it is quite clear that the genetic mix over-biased towards the bass Guitar compared to the other three tracks. In the 'Human' mix, the drums and bass are both favoured, with the Electric Guitar providing background bedding to the mix. The lead, only present at the start, provides little impact on the mix overall, altough it would be easy to spot if it was missing or substantially quiet. In this case, the mix would sound overly bass-heavy. Table 6.26 gives some high-level features of the mixes for 'Left Blind'. Spectral centroid is one of many high-level features that can be used to analyse an audio stream (Grey and Gordon, 1978). The centroid calculates the barycenter

of the spectrum by weighting each bin X_k with the centre frequency of that bin F_k, given in Equation 6.5 (Peeters, 2004). The discrete version is given in 6.6.

$$\mu = \int F_k \frac{X_k}{\sum X}\, dk \tag{6.5}$$

$$\mu = \frac{\sum_{k=0}^{K} (|X_k|\, F_k)}{\sum_{k=0}^{K} X_k} \tag{6.6}$$

$$\text{SFM} = \frac{(\prod_k X_k)^{\frac{1}{K}}}{\frac{1}{K}\sum_k X_k} \tag{6.7}$$

$$\sigma^2 = \int (F_k - \mu)^2 \frac{X_k}{\sum X}\, dk \tag{6.8}$$

$$\gamma_1 = \frac{m_3}{\sigma^3} m_3 = \int (X_k - \mu)^3 \left(\frac{X_k}{\sum X}\right) dk \tag{6.9}$$

The spectral centroids, listed in Table 6.26 for 'Left Blind', confirms this would be a bassier mix compared to the other three mixes. The 'Genetic' mix scoring 1.941kHz, which is significantly lower than all the other four mixes. This on it's own does not statistically mean the energy is inherently bass heavy. Spectral Flatness in equation 6.7 (Peeters, 2004), is designed to show how noisy the spectrum is, but can be used to show if there is any bias to bassier or brighter tones. A score of 1 would indicate a perfectly flat spectrum and a score of 0 a perfect sine wave. For this metric, all four of the mixes scored similar measurements of 0.30, indicating there was a fair amount of band-limited activity experienced in all the songs.

Spectral Skewness gives a measurement of the asymmetry of the spectrum around the mean value. This will show if the spectrum itself is balanced in terms of its energy distribution. A score of 0 means the energy is perfectly symmetric about the mean. A score less than 0 means more energy to the right, indicating more energy is held above the mean. A score greater than 0 means more energy to the left, indicating more energy is held below the mean. To help calculate it, the spectral variance, σ is also needed. Equation 6.8 gives the Spectral Standard Deviation, or σ^2. This is used, along with the centroid μ to calculate the Skewness γ_1. The σ is known as the spectral spread and represents the width of the distribution about the Spectral Centroid measured in Hertz. A larger number represents a wider distribution, smaller would indicate a narrower, approaching sinusoidal, spectral envelope. Lower spectral centroids should naturally correlate with narrower spectral spreads, as there is less room for the distribution to naturally occur, so on its own it is not indicative of a difference unless the centroids are similar. The spectral skewness, listed in Table 6.26 for 'Left Blind' do give more information as to how much energy is above or below the mean. The 'Genetic' mix has a significantly higher Skewness score than the 'Human' and 'Mansbridge et al. (2012b)' scores, showing there is significantly more energy below the centroid score of 1.941kHz.

Feature	Unity	Human	Mansbridge et al. (2012b)	Genetic
RMS amplitude	-22.8915	-23.2753	-23.2849	-22.6989
Spectral Centroid μ	3.160kHz	3.972kHz	3.946kHz	3.164kHz
Spectral Flatness	0.2862	0.3636	0.3559	0.2950
Spectral Spread σ	4179	4578	4522	4332
Spectral Skewness γ_1	1.9352	1.5081	1.5108	1.8826

Table 6.25: Analysis of the four mixes created for the Song 'I'm Alright'.

Feature	Unity	Human	Mansbridge et al. (2012b)	Genetic
RMS amplitude	-22.4583	-22.8483	-22.7662	-22.3408
Spectral Centroid μ	2.806kHz	3.893kHz	3.461kHz	1.941kHz
Spectral Flatness	0.2239	0.3155	0.2815	0.1450
Spectral Spread σ	3854	4332	4179	3221
Spectral Skewness γ_1	1.9465	1.4124	1.6050	2.6500

Table 6.26: Analysis of the four mixes created for the Song 'Left Blind'.

Feature	Unity	Human	Mansbridge et al. (2012b)	Genetic
RMS amplitude	-22.2991	-22.9088	-22.8117	-22.1227
Spectral Centroid μ	2.764kHz	3.097kHz	3.522kHz	2.361kHz
Spectral Flatness	0.2332	0.2538	0.2991	0.2050
Spectral Spread σ	4146	4159	4585	4071
Spectral Skewness γ_1	2.1445	2.0261	1.7474	2.3267

Table 6.27: Analysis of the four mixes created for the Song 'Sleigh Ride' .

Feature	Unity	Human	Mansbridge et al. (2012b)	Genetic
RMS amplitude	-23.4681	-23.3679	-23.8126	-23.4841
Spectral Centroid μ	3.228kHz	3.383kHz	3.634kHz	3.467kHz
Spectral Flatness	0.2511	0.2720	0.2885	0.2791
Spectral Spread σ	3762	3912	4025	3959
Spectral Skewness γ_1	1.9274	1.8407	1.7850	1.8173

Table 6.28: Analysis of the four mixes created for the Song 'The English Actor'.

Following from the song 'Left Blind', the next song to perform poorly was the song 'Sleigh Ride'. Figure 6.3 shows the 'Genetic' mix scoring the worst out of the four mixes, and only just above the Anchor. The algorithm on 'Sleigh Ride' de-emphasised the vocals. This is a Jazz piece being recorded near-live, meaning all the performers are together and interacting with each other. This makes the interaction between the four extremely locked and interdependent. By removing the lead instrument it makes the mix distant and lost, emphasised by the disproportionate amount of vocal bleed being picked up by the other microphones. Table 6.23 also shows the relative loudness for each track relative to the output mix. It is clear the mix had not just removed the vocal (-14.566 LUFS) but had again emphasised the bass instrument, the Double Bass (-0.339 LUFS). Table 6.27 gives the extracted features for the mixes. The Spectral Centroid for the 'Genetic' mix was the lowest, at 2.361 kHz. The best rated 'Human' mix had a centroid of 3.097 kHz. Along with this, the spectral skewness of 2.3267 was the highest of the four. This clearly shows the bass heavy focus of this mix, given the lower Centroid, compared to the other four mixes. The anchor 1 mix for sleigh ride did so much better because the Double Bass was removed from the mix. Not having this instrument contributed to higher intelligibility of the mix. Looking at the timeline plots for 'Sleigh Ride' in Figure 6.15, the Piano has a highly transient response, showing it also providing a rhythmic track. This tempo may contribute to the double bass not being needed to support the mix in the ears of listener.

Figure 6.11 shows the histogram of the Masked-to-Unmasked ratios over time for 'Sleigh Ride' given in Figure 6.8c. By taking the histogram and analysing the distribution, the properties of the four mixes are revealed. It is clear that the 'Human' mix actually scores the worst for Masking out of the four for 'Sleigh Ride'. But more interestingly the mix by Mansbridge et al. (2012b) had the lowest Masked-to-Unmasked ratio over time. Table 6.19 gives three statistical measurements of the histograms in Figure 6.11. As can be seen, the median of the distributions for 'Unity' and 'Genetic' are very similar at 0.0967 and 0.0930 respectively. The 'Human' mix scored 0.1811, indicating far higher levels of masking occurring than the for the other two mixes. 'Mansbridge et al. (2012b)' scored the lowest median at 0.0620 showing it had the lowest amount of masking overall. All four mixes scored similar standard deviations of 0.0428 to 0.0481 respectively This would indicate the 'Genetic' algorithm failed to correctly converge the cost function and did get stuck in a local minima. When testing against the whole song, rather than frame by frame, the 'Genetic' mix did score the lowest. Table 6.23 also shows the relative loudness for all of the songs were extremely varied, with all except the 'Mansbridge et al. (2012b)' mix having at least one track more than 10 LUFS below the relative loudness of the mix. For the 'Unity' mix both the Piano and Drums were significantly reduced. Fore the 'Human' the Piano was heavily reduced by -15.16dB, and the 'Genetic' had the Piano, Drums and Vocals reduced. This amount of variance shows why the anchors were quite varied as well, since most of the mixes had at least one track being removed or reduced, that anchor 1 which had one removed deliberately could compete. This explains why the mix by 'Mansbridge et al. (2012b)' scored so highly compared to the others, because its mix had all four tracks included.

Cost function	Improved	Original
Drums	-5.32	-6.74
Bass	-4.04	-2.72
Electric Guitar	-8.67	**-42.27**
Vocal	-3.28	-5.29

Table 6.29: Mix gains for the song 'I'm Alright' for the Original (Equation 6.10) and Improved (Equation 6.1) cost functions.

Cost function	Improved	Original
Drums	-13.25	-10.36
Electric Guitar	-17.11	-5.26
Lead Guitar	-20.00	-5.87
Bass Guitar	-8.30	-4.40

Table 6.30: Mix gains for the song 'Left Blind' for the Original (Equation 6.10) and Improved (Equation 6.1) cost functions.

Optimising the Cost Function

The system shows that genetic algorithms can be used to successfully create a mix suitable mix which minimises the masking of the tracks compared to human engineered mix. The first problem encountered in the system was the development of the cost function, which determines the environment the chromosomes are evaluated. The original cost function was defined as Equation 6.10. The cost function takes the Masked-to-Unmasked Ratio of the n-th track as $r[n]$ and squares it. Instead of returning just the maximally masked track, it will return the average masking over all the tracks by dividing by N of the sum of the $r[n]^2$.

$$c_i = \frac{\sum_{n=0}^{N} \left(1.0 - \min\left(r[n], 1.0\right)\right)^2}{N} \tag{6.10}$$

Because of the evolutionary style of the network, the algorithm will exploit weaknesses or failures in that cost function to maximise its fitness. The problem is to do with the use of an average in the cost function. In an ideal situation, the average masked to unmasked ratio would show the average amount of masking occurring in the system, which is a plausible metric. But the algorithm exploited the fact that, if a single track was muted, the average masking of the entire system could drop too. This is because, whilst for that particular track the masking ratio would approach zero, indicating it is entirely masked, its influence as a maskee on other tracks would also be removed, resulting in a higher average and therefore a better fitness score. In the best performing examples, the system selected the track which has the highest total amount of masking across the input tracks. If the algorithm could find this track and remove it from the mix it could inflate its scores to a point that the intended solutions would never be selected.

Cost function	Improved	Original
Double Bass	-5.38	-9.68
Lead Vocals	-13.17	-4.96
Piano	-13.25	-3.87
Drums	-10.51	**-38.77**

Table 6.31: Mix gains for the song 'Sleigh Ride' for the Original (Equation 6.10) and Improved (Equation 6.1) cost functions.

Cost function	Improved	Original
Synthesiser	-1.44	-9.23
Acoustic Guitar	-1.07	**-53.05**
Drums	-3.05	-5.14
Lead Vocals	-1.25	-2.25

Table 6.32: Mix gains for the song 'The English Actor' for the Original (Equation 6.10) and Improved (Equation 6.1) cost functions.

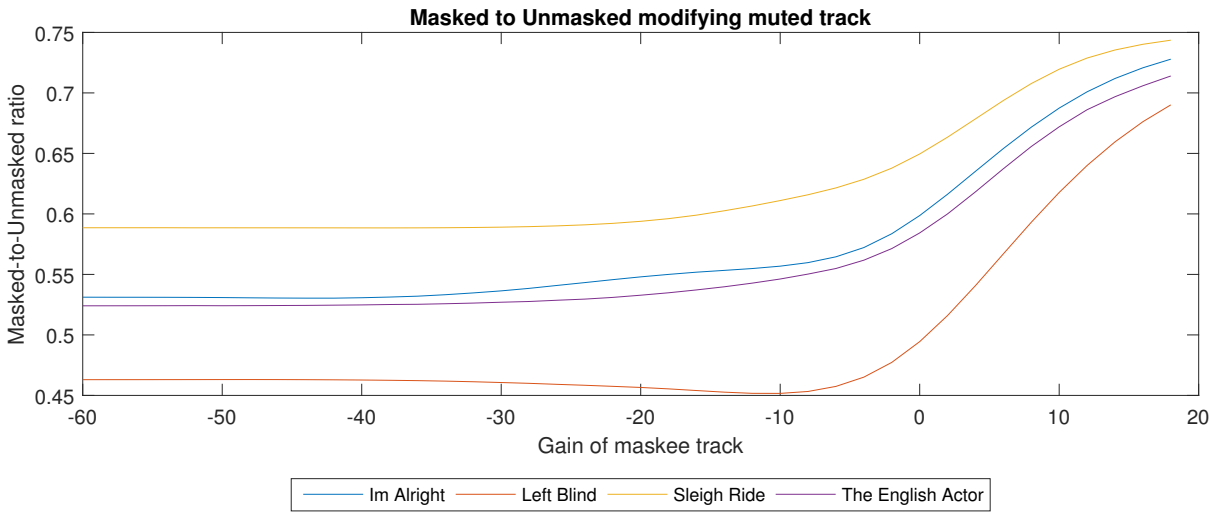


Figure 6.17: Performance of the cost function of each of the four songs. The gains used are the same as those given in Tables 6.29 to 6.32 with the most reduced track being adjusted to show the MUR curve.

Tables 6.29, 6.30, 6.31 and 6.32 give the normalised gain comparisons for the two cost functions across the four mixes. In a situation where a track is effectively muted, the gain is highlighted in bold. In 'I'm Alright' (Table 6.29), the Electric Guitar is set to -42.27dB, in 'Sleigh Ride' (Table 6.31), the Drums are set to -38.77dB and in 'The English Actor' (Table 6.32), the Acoustic Guitar is set of -53.05dB.

The only song which did not exhibit this pattern of behaviour was 'Left Blind'. In this case the Lead Guitar track ends early on, as shown in Figure 6.14. Because this track goes to silence, the weighting for most of the other tracks is only $1/3$, and therefore removing one track would not be beneficial to the average masked to unmasked ratio. As can be seen in Figure 6.17, for each of the four songs, the given track were selected as the normal track gain, but the x-axis shows the 'muted' track to be removed. For all the tracks, the system performs better as the track itself is removed from the mix, except for 'Left Blind' which does have a minima point around the Drums at -10dB. Once a track is nearly fully masked, it no longer has any benefit to the system to keep reducing the gain and therefore the cost functions plateau.

An improved cost function must therefore discourage the genetic algorithm from muting one track to improve its score. The method was to redefine Equation 6.1 as Equation 6.11. This instead returns the MUR of the track with the most amount of masking applied. This would penalise the muting of one track, since its MUR would be 0, giving a cost value $c_i = 1.0$, the worst score possible. With the modified score this would be the worst cost result possible. In short, it would have to ensure each track is audible.

$$c_i = 1.0 - \min(r) \tag{6.11}$$

The modified cost curve, using Equation 6.11, is given in Figure 6.18 for the four track test **Left Blind**. As shown in the preliminary tests, well defined minima help the algorithm converge on a solution space quickly. Running it in the four track experiment environment shows the performance of the cost function against the original mean based cost function. Where the old method would continue to minimise as the track approaches zero, the new function has a clear minima area defined for the search function to successfully explore.

Computational Efficiency

The software was written in Python3 to take advantage of the rich ecosystem of audio manipulation packages available, as well as the loudness extraction package Loudness by Ward et al. (2012). Throughout, the loudness model used is the Glassberg Moore model, updated in 2002. The Masked-To-Unmasked Ratio algorithm in Equation 6.1 uses the Short-Term Loudness (STL) and Short-Term Partial-Loudness (STPL) to determine the rate of masking occurring. The Short-Term Loudness is an individual analysis of the incoming audio stream, and therefore can be calculated ahead of time. The following code block does this, by loading in all samples, using the Loudness library to read the audio files into memory. The number of Ears was set to one because the system was not going to be modifying the panning of the audio signal. Therefore, both ears will be hearing the same source making the calculations redundant.

With the STL extracted for each track, the Genetic Algorithm could then begin. Each generation had a size of 32 chromosomes, with a parent ratio of 25%, meaning after each generation 8 chromosomes were saved

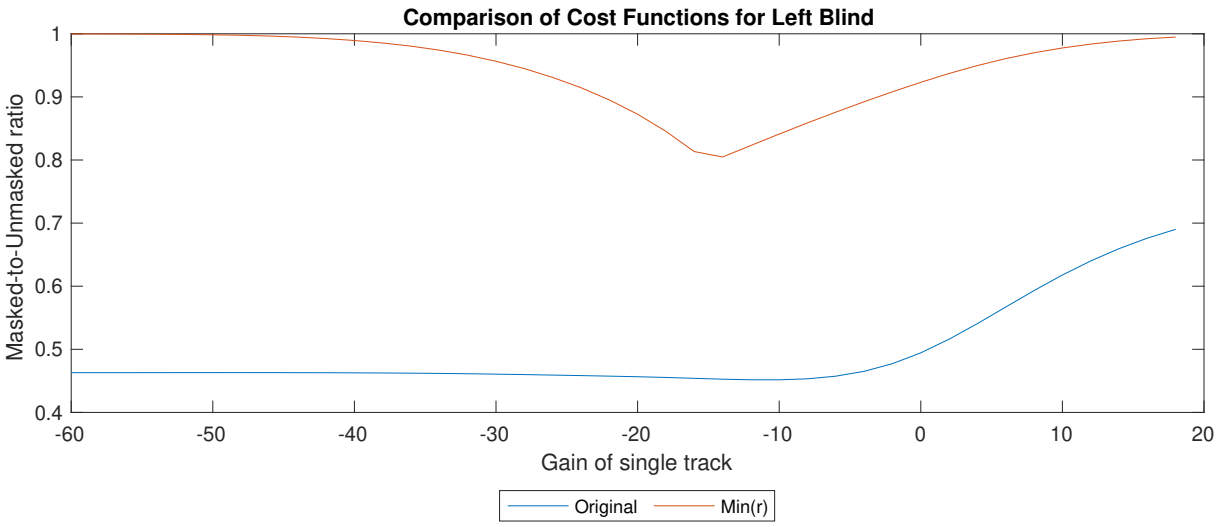


Figure 6.18: Comparison of the two cost functions, the original one and the improved for four tracks. As can be seen the original has a shape which can optimise for non-ideal situations if a track is removed.

to seed the next generation. Before the MUR could be extracted from the system, the mixes needed to be normalised. The normalisation function is there to ensure that the ratio between the gains is used. An overly loud mix could, in theory, achieve a better score by having flatter loudness curves rather than achieving a better mix. To minimise this possibility, the mixes were normalised so that they all equalled 70dBSPL. Another method, as used by Wilson and Fazenda (2017) is to have the chromosomes represent the delta coefficient of the gains themselves. This would still need normalising since it just says one track might have a higher gain compared to another, but the tracks themselves might have very different loudness levels anyway. Before entering this function, each `sound` mix has already been applied with the relevant gain value. As can be seen in the following code-block, the sounds are first down mixed to a mono file source. Then the Root-Mean-Squared is taken of the output audio file, converted to a dB Power value and the difference is found between 70 and the RMS. This difference in the SPL measurement is used to modify the gain of the individual tracks again such that the mix itself is normalised.

With the mix normalised, the calculation of the solver solution can be evaluated. The same loudness library is used, with the same parameters, except this task is to extract the Short-Term Partial-Loudness (STPL) from the model. With the two arrays, STL and STPL, for each track, the Masked-To-Unmasked Ratio, as defined in equation 6.1 is calculated.

Each generation of processing required 32 evaluations of the mix to be evaluated. This required 32 iterations of the cost function to be calculated. Since each chromosome represents the gain of 4 tracks, each tracks individual masked-to-unmasked ratio (MUR) needed to be calculated. This means each iteration required 4 loudness models to be executed 32 times, or a total of 128 loudness models. This placed significant strain on the system since each model is highly complex, as discussed in Section 2.5.1. On the test machine (2x Intel Xeon X5535 at 3.0GHz with 8GB of DDR2 ECC RAM) each chromosome took an average 4.46 seconds of CPU time to complete. Each generation took an average total of 15 minutes 32 seconds of CPU time to complete. The average execution time per chromosome is 28.13 CPU seconds. With each complete solution

requiring up to 1,000 iterations, this would mean an execution time of 258 hours 45 minutes 20 seconds to complete. To speed up the processing time, the work was performed in parallel across the multiple CPU cores.

Because the loudness model was an external package by Ward et al. (2012) it was not trivial to re-write the model to take advantage of parallel architectures found in the system. Instead, the genetic algorithm itself was implemented with parallelism in mind. Algorithms can be most effectively parallelised when the work loads are independent of each other, for both memory access and execution dependency. Each cost function for each chromosome is completely independent from the other cost function executions. The audio files can be loaded into shared memory for the processes to use as well. Once each chromosome is loaded, the gains are applied to the files and copied into the parallel process memory. The loudness model is then initialised, as shown above, and the cost calculated. On our 8-core test rig, with 8 threads running each thread would need to operate 4 loudness models. This dropped the execution time for each thread from 15 minutes 32 seconds of CPU time to 2 minutes 05 seconds. An average execution time per chromosome of 3.91 CPU seconds. Over 1000 executions this would equate to 34 hours 43 minutes 20 seconds. This should not that the total CPU time per chromosome is still 31.25 CPU seconds, similar to the above, except that this method is slightly slower per iteration. This can be explained by the increased cost required for creating and destroying the threads on each iteration, as well as locking resources to ensure atomic operations.

This time is still significantly higher than the unity mix, which requires no pre-loading, and the 'Mansbridge et al. (2012b)' mix, which requires minimal processing per track to calculate the decibel scale LUFS measurement to be applied to each track.

Evolutionary computing should converge as it approaches the optimal solution. This means the chromosomes should start to match over time, with more of the chromosomes more likely to match another chromosome as the evolutionary steps continue. Plus a certain number of the top performing chromosomes from previous generation would be preserved for use in the next generation. In this experiment the parents were made up of 25% of the previous generation, or 8 parents, to spawn another 24 chromosomes. Therefore, on each iteration, $1/4$ of the results are already known from past executions so long as they are stored. Therefore a further saving can be done by storing each chromosome in a lookup table along with its cost result. If a chromosome is encountered from a previous run, the execution time could be saved by simply looking up the previous results, thereby removing unnecessary redundant calculations. The lookup table also has the advantage of showing all the chromosomes that have been evaluated, giving an extra view of the mix space explored.

The lookup table is a map style structure, within the chromosome being examined stored as the key and its associated cost the value. Using this method is only useful if the cost of looking up the result is smaller than re-computing the result. In this example, each calculation requires nearly a full minute of CPU to compute. So as long as the compute cost of the lookup table remains under one minute for each chromosome it is worth doing. Given the cost of the lookup table is directly related to its length, the size of the table also has a bearing on this result. In one of the experiments given, after 1000 iterations the Lookup table had a size of 3,342 elements. Obviously this is not deterministic and could change each run, but this is an interesting value. After 1000 iterations, a total of 32,000 chromosomes should have been evaluated. So if only 3,342 are stored in the

Population Size	0.0	0.1	0.2	0.3	0.4	0.5
16	60.8333	88.2500	123.4167	145.1667	180.5000	200.0833
32	274.0000	334.6667	377.9167	406.6667	440.6667	502.0000
48	660.8333	664.5000	712.4167	700.6667	759.1667	781.1667
64	975.8333	922.5000	1033.0000	1014.3333	1109.9167	1149.4167
80	1128.5833	1334.5833	1337.5833	1394.6667	1452.0833	1445.0833
96	1626.7500	1470.0833	1647.5000	1730.3333	1703.1667	1768.0000
112	1868.0833	1922.0000	1907.8333	2023.6667	2053.6667	2044.7500
128	2186.4167	2080.8333	2198.2500	2290.0000	2363.1667	2383.7500
144	2231.0000	2665.0000	2671.5833	2645.7500	2694.1667	2704.8333
160	2761.9167	2791.7500	2943.1667	2911.5000	2964.5833	2989.7500
176	3141.2500	3280.7500	3193.2500	3283.9167	3294.8333	3311.0833
192	3563.4167	3580.5833	3590.4167	3594.4167	3607.0833	3610.0833
208	3765.1667	3823.7500	3851.5000	3907.0833	3892.4167	3922.6667
224	4177.7500	4192.5000	4203.0833	4210.2500	4218.2500	4222.9167
240	4428.5000	4373.3333	4504.8333	4510.8333	4519.0833	4528.5833
256	4791.3333	4605.0000	4813.5833	4821.7500	4781.0833	4834.7500

Table 6.33: Average lookup table length for the evolutionary computing using various mutation rates and population sizes

lookup table, this shows that only 10.44% of these 32,000 were unique. To calculate all 32,000 a total CPU time of 258 hours 45 minutes 20 seconds would have been needed. Instead, only 27 hours 1 minute 40 seconds is needed.

Table 6.33 gives the results of a test execution of the evolutionary computing algorithm using the Rosenbrock test function (Rosenbrock, 1960). The test executed the test for various population sizes, from 16 to 256 rising in increments of 16 against 6 mutation rates from 0% to 50%. The executions were stopped after 25 iterations and repeated 12 times. As can quickly be seen, an increase in the population size gives an increase in the table lookup size. The table size not only shows the efficiency saving but also the number of unique chromosome solutions encountered. For example, a generation size of 32 with a mutation rate of 0% gave a total lookup table size of 274. 32 chromosomes over 25 generations should equate to 800 chromosome evaluations. This gives a repeatability score of 65.75%, where each chromosome has this much of a chance of having already been evaluated. The relative lookup table size against the population size multiplied by the number of iterations is given in Table 6.34. As the mutation rate increases, the lookup table increases as well for the same chromosome size. This makes intuitive sense since on each generation there is more likely to be a completely new set of chromosomes generated. In theory with the number generation of chromosomes, every new chromosome would be unique since the crossover involves a random number generation as well to splice the chromosomes together. This number generator either is predictable or as the generations converge on a

Population Size	0.0	0.1	0.2	0.3	0.4	0.5
16	0.8479	0.7794	0.6915	0.6371	0.5488	0.4998
32	0.6575	0.5817	0.5276	0.4917	0.4492	0.3725
48	0.4493	0.4462	0.4063	0.4161	0.3674	0.3490
64	0.3901	0.4234	0.3544	0.3660	0.3063	0.2816
80	0.4357	0.3327	0.3312	0.3027	0.2740	0.2775
96	0.3222	0.3875	0.3135	0.2790	0.2903	0.2633
112	0.3328	0.3136	0.3186	0.2773	0.2665	0.2697
128	0.3167	0.3497	0.3130	0.2844	0.2615	0.2551
144	0.3803	0.2597	0.2579	0.2651	0.2516	0.2487
160	0.3095	0.3021	0.2642	0.2721	0.2589	0.2526
176	0.2861	0.2544	0.2743	0.2537	0.2512	0.2475
192	0.2576	0.2540	0.2520	0.2512	0.2485	0.2479
208	0.2759	0.2647	0.2593	0.2486	0.2515	0.2456
224	0.2540	0.2513	0.2494	0.2482	0.2467	0.2459
240	0.2619	0.2711	0.2492	0.2482	0.2468	0.2452
256	0.2514	0.2805	0.2479	0.2466	0.2530	0.2446

Table 6.34: Relative size of the lookup tables against the total number of chromosome evaluations from Table 6.33

Population Size	0.5	0.25	0.125	0.0625
16	150.3333	94.0833	63.9167	60.1667
32	361.1667	352.2500	204.5833	122.3333
48	597.6667	637.6667	516.1667	233.7500
64	787.0833	1013.6667	1097.9167	449.0000
80	1018.9167	1212.5833	1059.1667	593.0833
96	1224.1667	1600.9167	1680.7500	1034.2500
112	1437.5000	1879.5000	1860.2500	1553.0833
128	1642.1667	2188.7500	2038.7500	1685.0833
144	1850.8333	2587.7500	2302.0833	2651.5000
160	2058.8333	2954.7500	2869.5000	2640.0833
176	2268.2500	3277.4167	2903.7500	2939.9167
192	2476.2500	3455.2500	3683.0000	2286.5000
208	2685.1667	3882.0833	3816.7500	3676.9167
224	2893.6667	4194.5000	4435.8333	4411.7500
240	3104.2500	4494.5833	4693.0833	3963.3333
256	3310.6667	4783.5000	4951.2500	4882.9167

Table 6.35: Average lookup table length for the evolutionary computing using various parent ratios and population sizes

Population Size	0.5	0.25	0.125	0.0625
16	0.6242	0.7648	0.8402	0.8496
32	0.5485	0.5597	0.7443	0.8471
48	0.5019	0.4686	0.5699	0.8052
64	0.5081	0.3665	0.3138	0.7194
80	0.4905	0.3937	0.4704	0.7035
96	0.4899	0.3330	0.2997	0.5691
112	0.4866	0.3287	0.3356	0.4453
128	0.4868	0.3160	0.3629	0.4734
144	0.4859	0.2812	0.3605	0.2635
160	0.4853	0.2613	0.2826	0.3400
176	0.4845	0.2551	0.3401	0.3318
192	0.4841	0.2802	0.2327	0.5236
208	0.4836	0.2534	0.2660	0.2929
224	0.4833	0.2510	0.2079	0.2122
240	0.4826	0.2509	0.2178	0.3394
256	0.4827	0.2526	0.2264	0.2370

Table 6.36: Relative size of the lookup tables against the total number of chromosome evaluations from Table 6.35

solution the sample spaces become so small that there is no numerical difference between the chromosomes. It should also be noted that the mutation rate is only applied to the newly generated chromosomes, not those that are already generated.

Table 6.35 is the same test function as Table 6.33 except it varies the parent population ratio for each generation. The parent population size can be found by taking the ratio and multiplying it by the population size. For example, a population size of 256 with a parent ratio of 0.125 gives a parent population size of 32. These parent chromosomes are the best performing chromosomes and are used to seed the next generation. In this case the mutation rate was fixed at 0.1 (10%). As the parent size increases relative to the population size, the lookup table decreases. This is because more of each generation is carried over and implicitly must be in the lookup table from the previous iteration. There is also less chance of a brand new chromosome being made each time. For smaller population sizes, under 112 in this case, the relationship actually reverses, where a larger parent ratio results in larger lookup tables. This is most likely because when there is a small number of parent chromosomes to preserve, with only a dimensionality of 2 for this example case, each chromosome is multiplied multiple times resulting in a larger likelihood that it will experience the same chromosome twice. So having a larger parent ratio can help preserve more of the population to carry forward. As more iterations occur, and the chromosomes converge, it is expected that this would follow the trend of lower parent sizes resulting in larger tables and therefore more solutions experimented.

Each iteration also has a variable number of chromosomes that need computing, so the total execution time is less predictable. If only 4 chromosomes are unique for that iteration, only 4 threads would be needed. On the next cycle, 16 might be unique, requiring 8 threads to do 2 runs each. This shows the limitation of the parallelism at the chromosome level. This could be expanded by parallelising the toolbox, although this would require significant time to achieve, or increasing the number of chromosomes used per generation, since only 10.44% of all execution times are needed.

Searching through this table is also a non-trivial process, each chromosome item needs to be compared against each item in the lookup table to make sure it matches using strict equality. In a 1000 long lookup table, with chromosome sizes of 4 double floats long, this would require 4000 equality checks across 32KB of memory. If checking against 32 chromosomes per iteration, this would require 1024KB or 1MB of memory loads per invocation for a full check. In the real-world test, at 3342 with 32 chromosomes, the last 4 iterations spent an average of 11.6 CPU seconds per invocation processing the lookup table requests. This averages to 0.3625 seconds per chromosome. Whilst this is significantly shorter than the 31.25 seconds to execute the loudness model, it clearly is not a small amount when multiplied over many iterations.

This lookup process is optimised naturally by programming languages in a few ways. Firstly, once a match has been found there is no need to continue processing the operations, therefore the total amount of memory scanned will shorten. This can be taken advantage of by sorting the table by most-commonly accessed arrays. Whilst the sorting algorithm itself would not be trivial either, the savings may outweigh the length of scanning. Secondly, the costs of the parents from the previous iteration could be accessed immediately, removing them from ever needing to scan the table since they should have their costs readily available. This would remove

25% of the total scan time from 11.6 CPU seconds to 8.7 CPU seconds. Finally, duplicated chromosomes in the search pool should not scan the table twice.

As explained in the earlier sections, evolutionary computing algorithms work by reducing the search space by rejecting parts of the system which would cause the cost to increase, by favouring those chromosomes who show good solutions to the problem. Over-fitting is still a concern, where the chromosomes become identical, or very similar, such that the search space is reduced to a very small area. This is called convergence, and indicates the evolutionary algorithm is reaching a consensus on the solution. To ensure the algorithm does prematurely converge into a minima, a random event is applied on the chromosomes. The randomisation is referred to as the Mutation Rate and can help influence the learning rate by randomising a portion of the populations chromosomes on each iteration. No mutation means the evolutionary algorithm will be exposed to converging on a local minima as once it starts to reduce the space it will continue reducing into that space. A higher mutation rate will mean the algorithm never converges. For this project a mutation rate of 10% was selected, meaning on each iteration of 32 chromosomes, 3 should experience a randomisation or mutation. Selecting this variable, along with the population size, are the only two parts of the algorithm which require extensive tuning. A very high population count increases the amount of execution per step, a smaller amount reduces it but means more steps must be taken otherwise it may converge on the incorrect space.

6.3 Conclusion

This chapter introduced a novel method for exploring the mix space to generate a balanced mix for engineers. As was shown in Chapter 4, the starting position of a mix does have an impact on where the engineer will take the mix artistically, favouring traits which are more prominent. Therefore a system which can improve upon the starting position of the default Unity state of Digital Audio Workstations would be of benefit to an engineer.

To navigate this mix space, a set of metrics were identified to determine a suitable mix score, mostly to minimise the masking between tracks. The study in Chapter 4, and previous works by Ward et al. (2012) show that masking-minimisation as a target would provide an improved mix. To identify this mix, a form of Machine-Learning would need to be employed. Whilst neural networks and other Deep-Learning and Feature-Learning tool kits could be used, there was no suitable target score. Previous work in auto-mixers have relied upon real-time, feature driven calculations such as Mansbridge et al. (2012a), Mansbridge et al. (2012b) and Perez-Gonzalez and Reiss (2009). Building upon these generic automatic, cross-adaptive processes for real-time signal flow, an offline approach was instead used. This gives an advantage to real-time systems in that all the audio is available to the solver.

Now the requirement has shifted from autonomous control onto a search problem, where the search parameters were identified through a cost function to describe the space. Solvers are algorithms which iterate through a mathematical space and aim to minimise or maximise a cost function. Several solver algorithms exist, including Sampling, Gradient Descent and Evolutionary Computing. Testing these algorithms with several test cost functions showed Evolutionary Computing should provide the most efficient method for searching the space, since it was more resilient to local minima and able to reach a solution in the fewest number of

total steps. These systems sample the solution space with a given number of solutions, called chromosomes. These chromosomes together make up a genetic population. Each iteration the chromosomes are compared against the cost function to evaluate how well they solve the problem, giving them a fitness score. The best ranked solutions are saved and are used to populate the next iteration through a process called crossover. This is when two chromosomes are split at a random intersect and recombined to create two new chromosomes. This process is repeated until a new generation is created. The system takes its cues heavily from biological evolution, hence the term Evolutionary Computing. This is often more efficient than other space solvers since it can quickly isolate parts of the solution space for possible good solutions, preserving the knowledge gained from previous generations and quickly converge on a high quality solution. Tuning this algorithm proved paramount, as the cost of execution is expensive due to the inherent cost of calculating auditory models.

The cost function was described in Equation 6.1. The cost function takes the Masked to Unmasked ratio for each of the three tracks and returns the maximum number (Aichinger et al., 2011). Initially the cost function took the mean MUR of all the tracks, but testing showed the Genetic Algorithm would remove one of the tracks from the mix by muting it, thereby dropping that track to an MUR of 0, but boosting all other tracks due to the decrease of masker energy. This was not always the case, but edge cases aside was definitely undesirable.

To evaluate the performance of the generated mixes, a listening test was conducted using the Web Audio Evaluation Toolbox (Jillings et al., 2015). A total of 27 participants provided ranking information for each of the 16 tracks under test. It showed the performance of the mix was itself not as good as the Human engineered mix. It was not statistically under-performing compared to the Unity mix and the mix performed by Mansbridge et al. (2012b). Certain improvements to the algorithm would include an ability to prioritise tracks, such as vocals, which are known to be placed above the mix (Wilson and Fazenda, 2015b). Further improvements to the system would involve improving the efficiency of the auditory model such that computations are not so expensive. Using more of the available system resources and sharing computations in look-up-tables to aid faster response when recalculating the model would improve the model execution time significantly.

With this cost function in place, the algorithm could then perform the calculations to minimise the maximum amount of masking experienced by any one track. The algorithm is already more resilient as shown by the gains for the tests in Tables 6.1, 6.2, 6.3 and 6.4. The algorithm could be further improved from the results of this study. It is known that certain instruments should be placed more prominently in the mix (Jillings and Stables, 2017d; Wilson and Fazenda, 2015b). Therefore, the algorithm could be altered in one of two ways. The first method would be to apply a pre-defined ranking or relationship based on the instrument type of the tracks coming in. For traditional sessions, it is clear the vocal loudness has an impact on the mix scores, therefore vocal tracks could have a cost function which overly penalises them if they are reduced. Such a function could either come from the cost function itself, by biasing the vocal track a given amount, or from the chromosome structure which would limit the range of the vocal track.

Another method of improvement to the algorithm is to use relative gains instead of absolute gains. A relative gain system would remove one dimension from the chromosome, thus reducing the dimensionality of the space. Currently the system behaves in this reduced environment, because of the mix normalisation that occurs after

each mix is built. For example, given chromosome A with values $[-10, -10, -10, -10]$ and chromosome B with values $[-5, -5, -5 - 5]$. After normalisation, these will both equal the same chromosome value since the underlying mix will be the same. For the Lookup Table, these are difference entries meaning if these do get the same score, because they are ultimately the same mix, it can cause the system to not converge correctly.

Relative gains instead take the difference between the two mixes. Chromosome A and B would both be equal to $[0, 0, 0]]$, since the delta between each entry is 0. Applying this to the tracks would require one of the tracks to always be at gain 0, and the other tracks to be derived from the delta. The delta's can either be derived from the neighbour track or from this root track. By using this delta, a degree of dimensionality is reduced and the cost function would be more highly optimised to solve to the mix space.

Further to this, there is only so much masking optimisation can be done with just the volume controls at play. Using panning, whilst not always important to masking, can improve the intelligibility of the mix (Mansbridge et al., 2012a). Likewise using spectral effects such as Equalisers would allow the genetic algorithm to not only mix the tracks but to also remove problematic bands from interfering with each other.

Chapter 7

Conclusions

The aim of this thesis was to uncover how the role of the junior engineer could be automated to create assistive tools for mixing engineers in audio production. Therefore the first objective of this thesis was to understand the operations an engineer currently undertakes to complete this task. The output should be a better knowledge of the roles of the studio engineer, such that assistive technologies can be developed which aid the workflow of music production. The approach taken to achieve these goals is divided into several key stages: data collection, engineer operations and novel techniques.

7.1 Data Collection

A literature review was conducted, highlighting the evolution of the role of the engineer in the music production workflow, and the impact technology has had on the production life cycle. It was understood that the junior engineer would assist the senior engineer in the mixing roles, such as organising and configuring the session, creating a balance mix and transitioning from the recording phase to mixing phase. Whilst several studies have been conducted based on understanding the mixing process and how engineers perform in a studio, these are generally highly focused on specific actions and processes (Wakefield and Dewey, 2015; King et al., 2010). Therefore the first step of the process was to collect this important engineering data.

Traditional data collection techniques are not entirely suited to understanding how an engineer performs the mix. Surveys and interviews provide highly subjective information, based on interpretation and are usually specific to certain engineers. This has led to a set of texts which are conflicting in their stated workflows (Izhaki, 2012; Senior, 2019; Eargle, 2002). Recording of engineers performing a mix would be able to collect the data level that would be needed, by capturing the actions of the engineer in a physical studio. This style of working would be difficult to transcribe and would require a large amount of researcher time to complete.

Thanks to enabling technologies, such as the increased distribution of the internet, the web audio API and web development toolboxes, creating a set of web-powered data collection systems is feasible. The Web Audio Evaluation Toolbox was built to conduct web-distributed subjective listening tests whilst ensuring conformity with published standards (Jillings et al., 2015). The toolbox included resources for designing, running, collecting

and processing the results from subjects all through their device browsers. Since it's creation it has been cited over 126 times with numerous research outputs using it to evaluate the performance of their research.

Another developed data gathering system is through plugin interactions. Based upon the SAFE plugins, which collected semantic descriptors along with feature information from users through an audio plugin (Stables et al., 2014), a system for building and deploying plugins in the browser was developed (Jillings et al., 2016d). This system defined host and plugin specifications and will work with multiple other web plugin standards, deployed as a collection of JavaScript and npm modules. This was not eventually used in the context of the research for this thesis, but was used for a plugin recommendation system run in parallel to this research (Stasis et al., 2017a).

The web based DAW was then developed to gather data from engineers through a familiar environment. The DAW presented engineers with a 'timeline' and 'mixer' view which performed similar functions to production software such as Avid Pro Tools or Apple Logic. The DAW was built to perform a subset of traditional functions, focusing on providing the key processes require to allow the engineer to perform a balance mix. The features missing were focused on editing of the audio content and MIDI support, neither of which should be needed for this stage of work. Multiple production workflows could be explored by the engineer through familiar controls such as 'Mute' and 'Solo' functions, grouping, sends, panning and volume controls.

All the data from the DAW was collected and stored in a PostgreSQL relational database. This provided static tables, analogous to traditional DAW session files by allowing the session to be rebuilt from the end position. It also included a history table for the session, track and sends interfaces. This replay table logged every actionable change made to a given item from creation to completion. Each action was time-stamped as well as holding action specific information. This allows for fine-grained analysis of the engineer information, tracing from start to finish of the mix.

7.2 Engineer Operations

In Chapter 4 a study was conducted on how engineers would approach the balance mix using the software created in Chapter 3. The online data collection platforms was used with five multi-tracks. Each multi-track had two starting points that alternate engineers were presented with. This was done to try and confirm that the initial perception of the mix had an impact on the final mix (Wilson and Fazenda, 2015b). From this study, 71 participants had completed a mixing task, of these a final 35 were filtered based on criteria to remove likely improper mixes, where the participants had not spent enough time or performed enough actions. The study captured 3,391 total engineer interactions with over ten hours worth of mixing time. Once the mixes were created a listening test, using the Web Audio Evaluation Toolbox was used to provide a subjective score for each mix (Jillings et al., 2015). This was performed using the APE interface which allowed the listeners to rank all the stimuli on the same axis (De Man and Reiss, 2014).

7.2.1 How did engineers approach a balance mixing session?

Engineers mostly started by listening to the tracks, also known as auditioning. This crucial step would give all the engineers their first impression of the mix they are presented with. From this point, the engineers would begin making their mixing decisions. The highest recorded single action type was volume control with an average of 3 adjustments per track. With panning also being a significant track action as well since both of these provided the entire mix control surface to the engineer. Certain actions only took place when the system was auditioning, such as volume and pan movements. This shows these actions are auditory focused, compared to structural actions such as creating groups, sends or renaming tracks which generally only occurred when playback was stopped.

7.2.2 What control structures are used in the session?

Studies into busses and send structures showed that mixes which had busses tended to perform better subjectively than those which did not (Ronan et al., 2015a). This was performed against completed mixes and it was unclear if this was true at earlier stages in the mixing cycle. It was clear there was a strong correlation between the number of busses a session had and the quality of the final mix. Even in this situation, where there is no single perceptual advantage to using busses as there were no processes involved, this showed that the organisation of the session was important to the engineers' ability to handle the session. Sends were not extensively used because sends are only used for effects such as parallel compression or reverberation. This is confirmed by the few send busses created having names such as 'Verb' or 'Space'. Engineers did use the sends and mute controls extensively throughout to help control the flow of information in the mix, with the ability to focus in on certain sections. Some engineers never used this approach though, with others muting all the way down to a handful of tracks before building back up to the full mix.

7.2.3 How does the user interact with the graphical user interface?

The engineers worked left to right in the track order presented, with the most likely action to next occur on itself or the next neighbour. This strong relationship between the order of the track could be further explored in future studies to confirm if the order of the tracks is important or not. When operating by instruments, it shows that when an action occurs on a track, the next action is likely to occur on a track of the same instrument type. This is a confirmation bias as all the sessions were already laid out by instrument type, so it is unclear if this is a result of that action. The amount of actions occurring remained fairly consistent throughout the session duration.

7.2.4 Are there any similarities in the final mixes?

When exploring if the engineers had any commonality in the final mix produced, the engineers were split into two different groups. Each group was given a different initial gain and pan structure for the mix. After the analysis, it was clear that in Queens Light, which had 5 sessions in each group, that there was a strong clustering between the groups mix space. Figure **??** shows this when performing hierarchical clustering, as well as visually the mix space in Figure 4.49. The spectrum content also showed engineers would often follow a

similar trend based off the initial information presented to the engineer. Some universal concepts do come through, with vocal tracks and bass tracks being boosted significantly above the rest of the mix. The bass instrument boosting was not something previously uncovered, but due to the lack of processing available to the engineers, the bass tracks were gained stronger to ensure the bass content was present. Figure 4.43 shows the spectrograms of all the mixes and follows the curve created by Pestana et al. (2013). When producing the mix one of the most important items that was minimised was masking, with the engineer using decisions which ultimately led to mixes having lower masking properties than the original mix. These also correlated well with the scores given by the listening study.

7.3 Novel assistive technologies

From the mixing study in chapter 4 the two most significant aspects that engineers should focus on are group structuring and mask minimisation. Combined these two systems would provide an automatic balance mix generator of unknown multi-tracks, automating a 'junior engineer' role that would have been available to engineers of previous generations.

7.3.1 Automated Grouping

The grouping structure had the most significant impact on the mix, although engineers still do not use grouping structures significantly. This shows that either engineers do not believe they will need them, or they are too costly in terms of time for them to set up. Engineers which set up these structures early have a significant advantage over those who do not set them up. Therefore a system which could automate or suggest the creation of a grouping structure would be beneficial to engineers.

The developed system was built to use the instrument labels often given to a track by engineers in the session, either through the track name, labelling system or using automatic classification systems. With the instrument labels gathered, a knowledge system was needed to find a suitable set of relationships between each instrument. This again is not a well consistent study with multiple sources providing different relationships and structures, none of which provide an exhaustive list of relationships. Using Wikipedia, through its SPARQL endpoint DBPedia, the instrument information can be gathered. Since this is a public data set it is often updated with new information allowing for the knowledge tree to grow. Using graph theory to build the search graph, commonalities between the instrument labels is calculated and then grouped into clusters. These clusters are based on how similar the subjects relating to each instrument are, by measuring how many of them share common subjects using the Jaccard similarity score.

The system was evaluated by comparing against 81 mixes created by real-world engineers form the study in Chapter 4 and previous research which was made available (Jillings and Stables, 2017d; Ronan et al., 2015b). With the 81 mixes the graph edit distance, maximum common subgraph and probability density functions of graph features were used to show how similar the generated structures are to the engineer provided structures. Over the twelve songs in the data set, the generated structured scored closer similarity to the generated graphs than the failure cases of no grouping or over-grouping. The naming of the graphs is much harder to quantify.

This is because of the language used by engineers is often truncated. For example, 'BG' which is used to refer to 'Backing Vocals'. Using similar measurements such as Levenshtein distance, Longest Common Substring and a combination of the two, showed that the labels generated are not similar to those provided to engineers except in a few scenarios. Using the semantic scoring similarity through BERT the similarity is shown to be there on a semantic level (Zhang et al., 2019).

7.3.2 Automatic Balance Mix

In Chapter 6 a novel method for automatic mixing was proposed to improve the quality of the initial balance mix as made by the engineer. By restricting the problem to this more fundamental environment it should have simplified the problem to a point where a suitable mix could be established. The first impression of the mix also seems to have significance on the final outcome of the mix, with both groups A and B ending up in different final mix clusters. Therefore a system which can automatically mix the song with the goal of minimising masking would improve the presented balance mix for the engineer.

Previous automatic mixing systems have focused on either real-time approaches based on engineering perceptual tasks, such as equal loudness Mansbridge et al. (2012b) or using offline rule-based models De Man and Reiss (2013a). Since the system is primarily aimed at automating for a balance mix, an offline solution could be used which frees up the limitations that would traditionally be placed on the algorithm. The entire audio scene could be analysed from start to finish with the same information the engineer would receive.

Based on this offline approach, it was quickly established that the problem was best suited to a solver. This is because the real-time systems would use feature extraction techniques and react to changes in these features, possibly missing out on future cues. The system's job is also to set the faders, not have them react dynamically such as drawing out automation on the track. Neural networks and other unsupervised learning techniques would also be tricky to find suitable data sets, and have proved to not always be reliable in subjective listening tests with wide variances of performance (Steinmetz et al., 2021). This is most likely due to the fact that, whilst there are many track libraries available for multi-track performances, there is not a lot of data of completed mixes from start to finish.

Instead a system which could follow the mixing engineer rules as set out in texts would be a suitable starting point for the system. From Chapter 4 it was clear that a target was to minimise masking which occurs in the auditory system when the energy from one frequency is loud enough to cause the auditory system to be insensitive or filter out a nearby frequency (Fletcher, 1940). This phenomenon is called masking and was modelled into a computational system which allows machines to estimate the perceived masking levels (Moore et al., 1997). With the auditory model available it was possible to quantify the amount of masking experienced by a subject with a target and masker audio source. By applying this across a whole song, a metric called the Masked to Unmasked Ratio can be used to give an easy to interpret number of the amount of masking experienced by the listener (Aichinger et al., 2011).

The results showed for the mixed songs, the evolutionary computing output successfully minimised the masking of the tracks in question, without needing to use any panning controls. Whilst this was technically achieved it

did not score consistently well in the subjective listening test. This is because it would boost the level of any tracks which were not constrained by having a conflicting spectra in another track. This would most often be the bass track, of which there would usually be only one or two. The evolutionary computing system would therefore produce a large gain on this track. The performance of the auditory model also makes it prohibitively costly to run on current hardware and should be optimised for a real-world production cycle. Whilst evolutionary computing does reduce the number of comparisons needed compared to other solvers it still required several hours of evaluation to produce the desired mix.

7.4 Critique

From this thesis, the role of the engineer and the performance of the engineer in the mix is now more understood. It is clear that the engineer can have a wide ranging impact on the perceived performance of the mix, with poor workflow techniques resulting in under performing mixes. Two of the clearest factors which impact on the mixing process for an engineer are the use of sub-grouping to organise the mix, and mixing towards masking minimisation. Two systems were then developed aimed at solving both of these problems, with novel solutions provided for both and evaluated to assess their performance. To improve the quality of the data there are areas for improvement in each of the systems.

The primary issue with the data collection framework is the granularity of the data. The data would be improved by tracking the movements of the continuous volume and pan sliders, such that it was possible to understand the full range of motions. This would require some careful designing since the system should distinguish between a movement and a decision, but this would have provided further insight into the action being taken.

As discussed in Chapter 3, the web based nature of the system means it is an uncontrolled resource, where there is no guarantee that the participant is operating in a suitable environment. For this reason, it would be better to gather some survey entries before the study took place to ensure the system was set up correctly. This includes asking the engineer to describe their environment and equipment such that better filtering of subjects could take place.

The system was designed to have engineers progress through all five mixes. Most participants completed the first mix *Queen's Light* which ended up having a third of the total mixing submissions. The system could randomise the presented mixes to ensure a greater balance across all the mixes available.

The mask minimising balance mix is not in a performant state and would require more research to improve the efficiency of the system to make it a usable system. The system also does not yet explore panning, because of this performance limitation, since it would require double the analysis time for two ears. The cost function as well should be developed further to have more enforceable rules, to ensure that whilst the tracks are mask minimised, the mix is not unbalanced in its spectra. This is a complex set of interactions and future research would be needed to understand the balance between the two.

7.5 Future Work

With the data collection framework presented in Chapter 3 there are numerous avenues for research that could be pursued. One aspect from the study in Chapter 4 is the impact the starting position has on the outcome of the mix. A formal study based on this phenomenon, along with track positioning and ordering, would show greater insight into the importance of presentation of the work in the mix.

Intelligent production tools will continue revolutionise the music and audio production industry as new technologies evolve. The groundwork laid by the previous works shows a strong pipeline of tools to enable producers of audio context to create high quality work in a fraction of the time. One such area that has not been under focus is how intelligent tools can be used to improve the accessibility of audio assets.

Visually and auditory impaired users, both creators and consumers, usually have to use third party tools to improve the experience. For example, visually-impaired users can use screen reader technologies, where the computer will read out the visual element the mouse is currently over. This can help bridge the gap in human-computer interfacing, where the graphical user interface is the dominant form of computer response. In audio production, screen readers often do not work as they use information given by the software to respond to the users, and the highly-specialised DAW software does not provide such information. This removes the ability for even basic screen readers to feed-back what is going on. Likewise, DAW's use fader positions, visual level meters and spectrograms to provide complex information back to the engineers, which a screen-reader would not understand on its own.

One avenue of this work can be to use the given platform and data-collection methodology already developed to conduct targeted research on various levels of feedback needed to visually-impaired engineers. Not only can improved screen-readers be made, but intelligent feedback such a voicing of alerts the engineer should be made aware of. Haptic feedback and other forms of physical feedback devices can also help users navigate the session, providing improved levels of feedback of information for the engineer to process.

Controlling of the software is also a problem for visually impaired engineers. Different forms of HCI devices have been used in the past to navigate and control music production software. These devices elevate beyond the keyboard and mouse that is so common-place today to provide a greater field of control. With the rise in recent years of smart-home devices such as Amazon's 'Alexa' or Apple's 'Siri' it is not a push to provide a form of vocal control to a DAW. This system would be possible to embed into the underlying data-collection system to provide a third method of control. Integrating with AWS and Azure platforms, which expose Speech-to-text as a service and the ability to create your own dictionary of terms. There would be a significant time lag in developing such a system, but it would be revolutionary for a DAW to integrate.

Moving onto the other side of the coin, audio consumption should also be prioritised. Intelligibility of content is a known problem for audio impaired listeners. Listeners with partial deafness or audio trauma can be excluded from content which they cannot suitably experience. Likewise, as people age their hearing degrades and changes. Monitoring software can help engineers to understand and grade their content to ensure it is intelligible and accessible. Just as there are measures for visual content for people with various forms of colour-blindness,

it should be possible to provide a measurement based on several different forms of auditory deafness and impairments.

Bibliography

Ahn, C. W. and Ramakrishna, R. S. (2003), Elitism-based compact genetic algorithms, in *IEEE Transactions on Evolutionary Computation*, volume 7, no. 4, pages 367–385. [Cited on Page 28.]

Aichinger, P., Sontacchi, A. and Schneider-Stickler, B. (May 2011), Describing the Transparency of Mixdowns: The Masked-to-Unmasked-Ratio, in *Audio Engineering Society Convention 130*, URL `http://www.aes.org/e-lib/browse.cfm?elib=15811`. [Cited on Pages 22, 23, 122, 208, and 215.]

Anthony, B. (2018), Mixing as a performance: Educating tertiary students in the art of playing audio equipment whilst mixing popular music, in *Journal of Music, Technology & Education*, volume 11, no. 1, pages 103–122. [Cited on Page 8.]

AudioMulch (????), What is AudioMulch?, `http://www.audiomulch.com/info/what-is-audiomulch`, accessed: 2020-08-23. [Cited on Pages ix and 14.]

Ayodele, T. O. (2010), Types of machine learning algorithms, in *New advances in machine learning*, volume 3, pages 19–48. [Cited on Page 23.]

Back, T. (1996), *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford university press. [Cited on Page 26.]

Barbier, G., Zafarani, R., Gao, H., Fung, G. and Liu, H. (2012), Maximizing benefits from crowdsourced data, in *Computational and Mathematical Organization Theory*, volume 18, no. 3, pages 257–279. [Cited on Page 42.]

Bas, E., Tekalp, A. M. and Salman, F. S. (2007), Automatic vehicle counting from video for traffic flow analysis, in *2007 IEEE intelligent vehicles symposium*, pages 392–397, Ieee. [Cited on Page 37.]

Bech, S. and Zacharov, N. (2006), *Perceptual Audio Evaluation*, John Wiley & Sons. [Cited on Page 38.]

Bell, A. P. (2014), Trial-by-fire: A case study of the musician–engineer hybrid role in the home studio, in *Journal of Music, Technology & Education*, volume 7, no. 3, pages 295–312. [Cited on Page 13.]

Bergroth, L., Hakonen, H. and Raita, T. (2000), A survey of longest common subsequence algorithms, in *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 39–48, doi: 10.1109/SPIRE.2000.878178. [Cited on Page 165.]

Berry, R. (2020), Radio, music, podcasts-BBC Sounds: Public service radio and podcasts in a platform world, in *radio journal: international studies in broadcast & audio media*, volume 18, no. 1, pages 63–78. [Cited on Page 44.]

Bitzer, J., LeBoeuf, J. and Simmer, U. (2008), Evaluating perception of salient frequencies: Do mixing engineers hear the same thing?, in *Audio Engineering Society Convention 124*, Audio Engineering Society. [Cited on Pages 8 and 101.]

Booker, L. B., Goldberg, D. E. and Holland, J. H. (1989), Classifier systems and genetic algorithms, in *Artificial intelligence*, volume 40, no. 1-3, pages 235–282. [Cited on Page 30.]

Brandenburg, K. and Bosi, M. (1997), Overview of MPEG Audio: Current and Future Standards for Low Bit-Rate Audio Coding, in *J. Audio Eng. Soc*, volume 45, no. 1/2, pages 4–21, URL `http://www.aes.org/e-lib/browse.cfm?elib=7871`. [Cited on Page 22.]

Braun, S. (2001), WINDOWS, in Braun, S. (Editor), *Encyclopedia of Vibration*, pages 1587–1595, Elsevier, Oxford, ISBN 978-0-12-227085-7, doi: https://doi.org/10.1006/rwvb.2001.0052, URL `https://www.sciencedirect.com/science/article/pii/B0122270851000527`. [Cited on Page 109.]

Brindle, A. (1981), Genetic Algorithms for Function Optimization PhD Thesis, in *University of Alberta*. [Cited on Page 28.]

Bromham, G., Moffat, D., Barthet, M. and Fazekas, G. (2018), The impact of compressor ballistics on the perceived style of music, in *Audio Engineering Society Convention 145*, Audio Engineering Society. [Cited on Page 49.]

Bruford, F., Barthet, M., McDonald, S. and Sandler, M. (2019), Modelling musical similarity for drum patterns: A perceptual evaluation, in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, pages 131–138. [Cited on Page 49.]

Buffa, M., Lebrun, J., Kleimola, J., Larkin, O. and Letz, S. (4 2018), Towards an open Web Audio plugin standard, in *Companion of the Web Web Conference 2018*, doi: 10.1145/184558.3188737. [Cited on Page 54.]

Bullock, J. and UCEB Conservatoire (2007), Libxtract: a Lightweight Library for audio Feature Extraction., in *Proceedings of the International Computer Music Conference*, volume 43. [Cited on Page 56.]

Bunke, H. and Shearer, K. (1998), A graph distance metric based on the maximal common subgraph, in *Pattern recognition letters*, volume 19, no. 3-4, pages 255–259. [Cited on Pages 134, 150, and 151.]

Burgess, R. (2014), *The History of Music Production*, Oxford University Press. [Cited on Pages 5, 13, 43, and 173.]

Can I Use (2022), Can I Use: Web Audio API, Online. [Accessed 2nd April 2022], URL `https://caniuse.com/audio-api`. [Cited on Page 45.]

Carson, T. (2021), immaterial. cloud: Using peer-to-peer technologies for music, in *2021 Web Audio Conference*, Barcelona, Spain. [Cited on Page 44.]

Cartwright, M., Pardo, B., Mysore, G. J. and Hoffman, M. (March 2016), Fast and Easy Crowdsourced Perceptual Audio Evaluation, in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'16)*. [Cited on Pages 42, 67, and 178.]

Cartwright, M., Pardo, B. and Reiss, J. (2014), Mixploration: Rethinking the audio mixer interface, in *Proceedings of the 19th international conference on Intelligent User Interfaces*, pages 365–370. [Cited on Pages ix, 34, and 35.]

Cartwright, M., Seals, A., Salamon, J., Williams, A., Mikloska, S., MacConnell, D., Law, E., Bello, J. P. and Nov, O. (2017), Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations, in *Proceedings of the ACM on Human-Computer Interaction*, volume 1, no. CSCW, pages 1–21. [Cited on Page 42.]

Caruana, R. A., Eshelman, L. J. and Schaffer, J. D. (1989), Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover, in *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 1*, pages 750–755. [Cited on Pages ix and 30.]

Casario, M., Elst, P., Brown, C., Wormser, N. and Hanquez, C. (2011), HTML5 Forms, in *HTML5 Solutions: Essential Techniques for HTML5 Developers*, pages 63–96, Springer. [Cited on Page 61.]

Cauchy, A. et al. (1847), Méthode générale pour la résolution des systemes d'équations simultanées, in *Comp. Rend. Sci. Paris*, volume 25, no. 1847, pages 536–538. [Cited on Page 26.]

Cha, S.-H. (2007), Comprehensive survey on distance/similarity measures between probability density functions, in *City*, volume 1, no. 2, page 1. [Cited on Page 153.]

Choi, H. (2018), AudioWorklet: The future of web audio, in *International Computer and Music Conference*. [Cited on Pages 44 and 55.]

Choi, H. and Berger, J. (2013), WAAX: Web Audio API eXtension., in *New Interfaces for Musical Expression*, pages 499–502, Seoul, South Korea. [Cited on Page 54.]

Clark, D. (1982), High-Resolution Subjective Testing Using a Double-Blind Comparator, in *Journal of the Audio Engineering Society*, volume 30, no. 5, pages 330–338. [Cited on Pages 39 and 48.]

Constantinou, S. (2019), Working With Sound in the DAW: Towards a New Materiality of the Audio-Object, in *Producing Music*, pages 229–245, Routledge. [Cited on Pages 53 and 59.]

Damskägg, E.-P. and Välimäki, V. (2017), Audio time stretching using fuzzy classification of spectral bins, in *Applied Sciences*, volume 7, no. 12, page 1293. [Cited on Page 49.]

David, H. A. (1963), *The method of paired comparisons*, volume 12, London. [Cited on Page 48.]

Dayhoff, J. E. (1990), *Neural network architectures: an introduction*, Van Nostrand Reinhold Co. [Cited on Pages 24 and 25.]

De Man, B., Boerum, M., Leonard, B., King, R., Massenburg, G. and Reiss, J. D. (May 2015), Perceptual evaluation of music mixing practices, in *138th Convention of the Audio Engineering Society*, Audio Engineering Society. [Cited on Pages 7 and 125.]

De Man, B., Jillings, N. and Stables, R. (2018), Comparing stage metaphor interfaces as a controller for stereo position and level, in *4th Workshop on Intelligent Music Production*. [Cited on Page 35.]

De Man, B., Leonard, B., King, R. and Reiss, J. D. (October 2014a), An Analysis and Evaluation of Audio Features for Multitrack Music Mixtures, in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*. [Cited on Page 101.]

De Man, B., Mora-Mcginity, M., Fazekas, G. and Reiss, J. D. (Oct 2014b), The Open Multitrack Testbed, in *Audio Engineering Society Convention 137*, online. `http://www.aes.org/e-lib/browse.cfm?elib=17400`. [Cited on Pages 65 and 176.]

De Man, B., Reiss, J. and Stables, R. (September 2017), Ten years of automatic mixing, in *Proceedings of the 3rd Workshop on Intelligent Music Production*. [Cited on Page 15.]

De Man, B. and Reiss, J. D. (Oct 2013a), A Knowledge-Engineered Autonomous Mixing System, in *Audio Engineering Society Convention 135*, URL `http://www.aes.org/e-lib/browse.cfm?elib=17011`. [Cited on Pages 15, 23, 38, 193, and 215.]

De Man, B. and Reiss, J. D. (May 2013b), A Pairwise and Multiple Stimuli Approach to Perceptual Evaluation of Microphone Types, in *Audio Engineering Society Convention 134*. [Cited on Page 242.]

De Man, B. and Reiss, J. D. (April 2014), APE: Audio Perceptual Evaluation toolbox for MATLAB, in *136th Convention of the Audio Engineering Society*. [Cited on Pages x, xv, 47, 48, 67, and 212.]

De Man, B. and Reiss, J. D. (September 2017), The Mix Evaluation Dataset, in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*. [Cited on Pages 5, 88, 102, and 132.]

Doktorski, H. (n.d), *Taxonomy of Musical Instruments*, `http://free-reed.net/description/taxonomy`. [Cited on Page 133.]

Driver, H. E. and Kroeber, A. L. (1932), *Quantitative expression of cultural relationships*, volume 31, Berkeley: University of California Press. [Cited on Page 25.]

Dugan, D. (1975), Automatic microphone mixing, in *Journal of the Audio Engineering Society*, volume 23, no. 6, pages 442–449. [Cited on Page 17.]

Eargle, J. (2002), *Handbook of Recording Engineering*, Springer, 4th edition. [Cited on Pages 10, 11, 12, 13, 38, 129, 130, and 211.]

Eberlein, E., Popp, H., Grill, B. and Herre, J. (1993), Layer-III-A Flexible Coding Standard, in *Audio Engineering Society Convention 94*, Audio Engineering Society. [Cited on Page 35.]

Eichas, F., Möller, S. and Zölzer, U. (9 2017), Block-Oriented grey box modelling of guitar amplifiers, in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, Edinburgh, UK. [Cited on Page 23.]

European Broadcast Union (6 2014), R128: Audio loudness normalisation and permitted maximum level, uRL `https://tech.ebu.ch/docs/r/r128.pdf`. [Cited on Page 178.]

Fenton, S. (october 2018), automatic mixing of multitrack material using modified loudness models, in *journal of the audio engineering society*. [Cited on Page 17.]

Févotte, C., Torrésani, B., Daudet, L. and Godsill, S. J. (2007), Sparse linear regression with structured priors and application to denoising of musical audio, in *IEEE Transactions on Audio, Speech, and Language Processing*, volume 16, no. 1, pages 174–185. [Cited on Page 24.]

Fletcher, H. (1940), Auditory patterns, in *Reviews of modern physics*, volume 12, no. 1, page 47. [Cited on Pages 19 and 215.]

Fletcher, H. and Munson, W. A. (1933), Loudness, its definition, measurement and calculation, in *Bell System Technical Journal*, volume 12, no. 4, pages 377–430. [Cited on Pages 19, 20, and 175.]

Fraunhofer IIS (2021), The MP3 History, URL `https://www.mp3-history.com/en/timeline.html`, accessed 27th July 2021. [Cited on Page 43.]

Gelineck, S., Büchert, M. and Andersen, J. (2013), Towards a more flexible and creative music mixing interface, in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 733–738. [Cited on Page 35.]

Giannoulis, D., Massberg, M. and Reiss, J. D. (2012), Digital dynamic range compressor design—A tutorial and analysis, in *Journal of the Audio Engineering Society*, volume 60, no. 6, pages 399–408. [Cited on Page 55.]

Gilkey, R. H. and Robinson, D. E. (1986), Models of auditory masking: A molecular psychophysical approach, in *The Journal of the Acoustical Society of America*, volume 79, no. 5, pages 1499–1510, doi: $10.1121/1.393676$. [Cited on Page 19.]

Gill, P., Stewart, K., Treasure, E. and Chadwick, B. (2008), Methods of data collection in qualitative research: interviews and focus groups, in *British dental journal*, volume 204, no. 6, pages 291–295. [Cited on Pages 35, 36, and 37.]

Goldberg, D. E. and Lingle, R. (2014), Alleles, loci, and the traveling salesman problem, in *Proceedings of the first international conference on genetic algorithms and their applications*, pages 154–159, Psychology Press. [Cited on Page 28.]

Gómez, J. S., Abeßer, J. and Cano, E. (2018), Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning., in *ISMIR*, pages 577–584. [Cited on Page 25.]

Goodwin, S. N. (Feb 2009), How Players Listen, in *Audio Engineering Society Conference: 35th International Conference: Audio for Games*, URL `http://www.aes.org/e-lib/browse.cfm?elib=15172`. [Cited on Page 36.]

Gover, M., Sarasúa, Á., Parra, H., Janer, J., Mayor, O., Cuesta, H., Pascual, M. P., Gkiokas, A. and Gómez, E. (2021), Choir Singers Pilot–An online platform for choir singers practice, in . [Cited on Page 44.]

Granello, D. H. and Wheaton, J. E. (2004), Online data collection: Strategies for research, in *Journal of Counseling & Development*, volume 82, no. 4, pages 387–393. [Cited on Pages 36, 37, and 43.]

Green, B. and Seshadri, S. (2013), *AngularJS*, " O'Reilly Media, Inc.". [Cited on Page 59.]

Greenwood, D. D. (1961), Auditory Masking and the Critical Band, in *The Journal of the Acoustical Society of America*, volume 33, no. 4, pages 484–502, doi: $10.1121/1.1908699$. [Cited on Page 18.]

Grey, J. M. (1977), Multidimensional perceptual scaling of musical timbres, in *Journal of the Acoustical Society of America*, volume 61, no. 5. [Cited on Pages 39 and 41.]

Grey, J. M. and Gordon, J. W. (1978), Perceptual effects of spectral modifications on musical timbres, in *The Journal of the Acoustical Society of America*, volume 63, no. 5, pages 1493–1500. [Cited on Page 193.]

Gribben, C. and Lee, H. (2015), Toward the development of a universal listening test interface generator in Max, in *Audio Engineering Society Convention 138*, Audio Engineering Society. [Cited on Pages xv and 47.]

Haas, A., Rossberg, A., Schuff, D. L., Titzer, B. L., Holman, M., Gohman, D., Wagner, L., Zakai, A. and Bastien, J. (2017), Bringing the web up to speed with WebAssembly, in *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 185–200. [Cited on Page 54.]

Harris, F. J. (1978), On the use of windows for harmonic analysis with the discrete Fourier transform, in *Proceedings of the IEEE*, volume 66, no. 1, pages 51–83. [Cited on Page 109.]

Hart, S. G. and Staveland, L. E. (1988), Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research, in *Advances in psychology*, volume 52, pages 139–183, Elsevier. [Cited on Page 36.]

Haupt, R. L. and Haupt, S. E. (2004), *Practical genetic algorithms*, John Wiley & Sons. [Cited on Page 29.]

Hepworth-Sawyer, R. (Editor) (2009), *From Demo to Delivery: The Process of Production*, Focal Press. [Cited on Page 11.]

Hepworth-Sawyer, R. and Golding, C. (2011), *What is Music Production? A producer's guide: the role, the people, the process*, Focal Press. [Cited on Pages 10, 12, and 13.]

Hines, A., Gillen, E., Kelly, D., Skoglund, J., Kokaram, A. and Harte, N. (2014), Perceived audio quality for streaming stereo music, in *Proceedings of the 22nd ACM international conference on Multimedia*. [Cited on Page 38.]

Hockman, J. A., Bello, J. P., Davies, M. E. and Plumbley, M. D. (September 2008), Automated rhythmic transformation of musical audio, in *Proceedings of the 11th International Conference on Digital Audio Effects*, pages 177–180, Citeseer. [Cited on Pages 5 and 16.]

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H. and Van de Weijer, J. (2011), *Eye tracking: A comprehensive guide to methods and measures*, OUP Oxford. [Cited on Page 37.]

Hracs, B. J. (2012), A creative industry in transition: the rise of digitally driven independent music production, in *Growth and Change*, volume 43, no. 3, pages 442–461. [Cited on Pages ix, 11, and 12.]

Huber, D. M. and Runstein, R. E. (2005), *Modern Recording Techniques*, Focal Press, 5th edition. [Cited on Pages 10, 11, 60, 88, and 173.]

International Telecommunication Union (8 1996), ITU-R Rec. P.800: Methods for subjective determination of transmission quality, uRL `https://www.itu.int/rec/dologin_pub.asp?lang=s&id=T-REC-P.800-199608-I!!PDF-E&type=items`. [Cited on Pages 38, 50, and 178.]

International Telecommunication Union (10 1997), Recommendation BS.1286: Methods for the subjective assessment of audio systems with accompanying picture, uRL `https://www.itu.int/rec/R-REC-BS.1286-0-199710-W/en`. [Cited on Page 38.]

International Telecommunication Union (3 2011), ITU-R BS.1770-2: Algorithms to measure audio programme loudness and true-peak audio level, uRL `https://www.itu.int/rec/R-REC-BS.1770-2-201103-S/en`. [Cited on Pages 17, 41, 68, 101, 109, 120, 177, and 178.]

ITU-R (1990), Recommendation BS.562-3: Subjective assessment of sound quality. [Cited on Page 39.]

ITU-R (2015), Recommendation BS.1116-3: Methods for the subjective assessment of small impairments in audio systems. [Cited on Pages x, 39, 40, and 48.]

ITU-R (2015), Recommendation BS.1534-3. Method for the subjective assessment of intermediate quality levels of coding systems, in . [Cited on Pages x, 40, 48, and 64.]

Izhaki, R. (2012), *Mixing Audio: Concepts, Practices and Tools*, Elsevier Ltd., 2nd edition, ISBN 978-0-240-52222-7. [Cited on Pages 5, 6, 10, 15, 50, 60, 61, 79, 84, 88, 95, 101, 105, 116, 126, 130, 144, 179, and 211.]

Jadhav, M. A., Sawant, B. R. and Deshmukh, A. (2015), Single page application using angularjs, in *International Journal of Computer Science and Information Technologies*, volume 6, no. 3, pages 2876–2879. [Cited on Page 59.]

Jillings, N., and Stables, R. (2016a), JSAP: Intelligent audio plugin format for the Web Audio API, in *Proceedings of the 2nd AES Workshop on Intelligent Music Production*. [Cited on Pages 2 and 4.]

Jillings, N., Bullock, J. and Stables, R. (2016b), Js-xtract: A realtime audio feature extraction library for the web, in . [Cited on Pages 2, 4, and 56.]

Jillings, N., De Man, B., Stables, R. and Reiss, J. D. (2018), Investigation into the effects of subjective test interface choice on the validity of results, in *Audio Engineering Society Convention 145*, Audio Engineering Society. [Cited on Pages 4, 37, and 40.]

Jillings, N., Moffat, D., De Man, B. and Reiss, J. D. (July 2015), Web Audio Evaluation Tool: A browser-based listening test environment, in *12th Sound and Music Computing Conference*. [Cited on Pages xiv, xv, 48, 64, 66, 178, 179, 208, 211, and 212.]

Jillings, N., Moffat, D., De Man, B., Reiss, J. D. and Stables, R. (April 2016c), Web Audio Evaluation Tool: A browser-based listening test environment, in *2nd Web Audio Conference*. [Cited on Pages 2 and 45.]

Jillings, N. and Stables, R. (2017a), Automatic channel routing using musical instrument linked data, in *Proceedings of the 3rd Workshop on Intelligent Music Production*. [Cited on Pages 3 and 4.]

Jillings, N. and Stables, R. (2017b), Automatic masking reduction in balance mixes using evolutionary computing, in *Proceedings of the 143rd Convention of the Audio Engineering Society*. [Cited on Page 3.]

Jillings, N. and Stables, R. (2017c), An Intelligent audio workstation in the browser, in *Proceedings of the 3rd Web Audio Conference*. [Cited on Pages 2 and 4.]

Jillings, N. and Stables, R. (Jun 2017d), Investigating Music Production Using a Semantically Powered Digital Audio Workstation in the Browser, in *2017 Audio Engineering Society International Conference on Semantic Audio*, URL `http://www.aes.org/e-lib/browse.cfm?elib=18770`. [Cited on Pages 2, 4, 147, 170, 171, 208, and 214.]

Jillings, N., Stables, R., Sean, E. and De Man, B. (2019), Digital Audio Workstation, Patent, URL `https://patents.google.com/patent/WO2021069630A1/en`, pCT/EP2020/078346. [Cited on Page 4.]

Jillings, N., Stables, R., Sean, E. and De Man, B. (2020), Collaboration system, Patent, URL `https://patents.google.com/patent/GB2595456A/en`, pCT/GB2021/051260. [Cited on Page 4.]

Jillings, N., Wang, Y., Reiss, J. D. and Stables, R. (September 2016d), JSAP: A Plugin Standard for the Web Audio API with Intelligent Functionality, in *141st Audio Engineering Society Convention*, Los Angeles, CA, USA. [Cited on Pages 2, 4, and 212.]

Jillings, N., Wang, Y., Stables, R., and Reiss, J. D. (2017), Intelligent audio plugin framework for the Web Audio API, in *Proceedings of the 3rd Web Audio Conference*. [Cited on Page 4.]

Julstrom, S. and Tichy, T. (Mar 1983), Direction-Sensitive Gating: A New Approach to Automatic mixing, in *Audio Engineering Society Convention 73*, URL `http://www.aes.org/e-lib/browse.cfm?elib=11785`. [Cited on Page 17.]

Katoch, S., Chauhan, S. S. and Kumar, V. (2021), A review on genetic algorithm: past, present, and future, in *Multimedia Tools and Applications*, volume 80, pages 8091–8126. [Cited on Pages 27, 30, and 176.]

King, R., Leonard, B. and Sikora, G. (2010), Variance in level preference of balance engineers: A study of mixing preference and variance over time, in *Audio Engineering Society Convention 129*, Audio Engineering Society. [Cited on Pages 9, 50, and 211.]

Kleimola, J. and Larkin, O. (2015), Web audio modules, in *Proceedings of the Sound and Music Computing 2015*, Maynooth, Ireland. [Cited on Page 54.]

Koljonen, J. and Alander, J. T. (2006), Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms, in *Proceedings of the ninth Scandinavian conference on artificial intelligence (SCAI 2006)*, pages 54–60. [Cited on Page 28.]

Kolozali, S., Barthet, M., Fazekas, G. and Sandler, M. B. (2011), Knowledge Representation Issues in Musical Instrument Ontology Design., in *ISMIR*, pages 465–470. [Cited on Page 133.]

Kraft, S. and Zölzer, U. (2014), BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality, in *Linux Audio Conference, Karlsruhe, DE*. [Cited on Pages xv, 45, and 47.]

Kreitz, G. and Niemela, F. (2010), Spotify–large scale, low latency, P2P music-on-demand streaming, in *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, IEEE. [Cited on Page 44.]

Labairu-Trenchs, A., Alsina-Pagès, R. M., Orga, F. and Foraster, M. (2018), Noise annoyance in urban life: the citizen as a key point of the directives, in *Multidisciplinary digital publishing institute proceedings*, volume 6, no. 1, page 1. [Cited on Pages 49 and 64.]

Leemann, A., Jeszenszky, P., Steiner, C., Studerus, M. and Messerli, J. (2020), Linguistic fieldwork in a pandemic: Supervised data collection combining smartphone recordings and videoconferencing, in *Linguistics Vanguard*, volume 6, no. s3. [Cited on Page 37.]

Leonard, B. (October 2020), A survey of current music technology & recording arts curriculum order, in *Journal of the Audio Engineering Society*. [Cited on Page 36.]

Levenshtein, V. I. (February 1966), Binary Codes Capable of Correcting Deletions, Insertions and Reversals, in *Soviet Physics Doklady*, volume 10, page 707. [Cited on Page 165.]

Lind, F. and MacPherson, A. (2017), Soundtrap: A collaborative music studio with Web Audio, in . [Cited on Page 15.]

Lipowski, A. and Lipowska, D. (2012), Roulette-wheel selection via stochastic acceptance, in *Physica A: Statistical Mechanics and its Applications*, volume 391, no. 6, pages 2193–2196. [Cited on Page 27.]

Lipshitz, S. P. and Vanderkooy, J. (1981), The Great Debate: Subjective Evaluation, in *Journal of the Audio Engineering Society*, volume 29, no. 7/8, pages 482–491. [Cited on Pages 39, 41, and 48.]

Lloyd, S. (1982), Least squares quantization in PCM, in *IEEE transactions on information theory*, volume 28, no. 2, pages 129–137. [Cited on Page 25.]

Lohn, J. D., Linden, D. S., Hornby, G. S. and Kraus, W. F. (June 2004), Evolutionary design of an X-band antenna for NASA's Space Technology 5 mission, in *IEEE Antennas and Propagation Society Symposium, 2004.*, volume 3, pages 2313–2316 Vol.3, doi: $10.1109/\mathrm{APS}.2004.1331834$. [Cited on Page 26.]

Lund, T. (2005), Realtime loudness control for broadcast, in *Broadcast Asia 2005*. [Cited on Pages 17 and 22.]

Ma, Z., De Man, B., Pestana, P. D., Black, D. A. and Reiss, J. D. (2015), Intelligent multitrack dynamic range compression, in *Journal of the Audio Engineering Society*, volume 63, no. 6, pages 412–426. [Cited on Page 5.]

Ma, Z., Reiss, J. D. and Black, D. A. (2013), Implementation of an intelligent equalization tool using Yule-Walker for music mixing and mastering, in *Audio Engineering Society Convention 134*, Audio Engineering Society. [Cited on Page 5.]

Maimon, O. and Rokach, L. (2005), *Data mining and knowledge discovery handbook*, Springer. [Cited on Page 114.]

Maniezzo, V. (Jan 1994), Genetic evolution of the topology and weight distribution of neural networks, in *IEEE Transactions on Neural Networks*, volume 5, no. 1, pages 39–53, ISSN 1045-9227, doi: $10.1109/72.265959$. [Cited on Page 27.]

Mann, Y. (2015), Interactive music with tone. js, in *Proceedings of the 1st annual Web Audio Conference*. [Cited on Page 54.]

Mansbridge, S., Finn, S. and Reiss, J. D. (October 2012a), An Autonomous System for Multitrack Stereo Pan Positioning, in *133rd Convention of the Audio Engineering Society*, URL `http://www.aes.org/e-lib/browse.cfm?elib=16478`. [Cited on Pages 10, 15, 17, 207, and 209.]

Mansbridge, S., Finn, S. and Reiss, J. D. (2012b), Implementation and evaluation of autonomous multi-track fader control, in *Audio Engineering Society Convention 132*, Audio Engineering Society. [Cited on Pages 5, 15, 17, 23, 176, 177, 178, 181, 183, 188, 189, 190, 191, 192, 193, 194, 195, 196, 201, 207, 208, and 215.]

Marrington, M. et al. (2017), Composing with the digital audio workstation, in *The Singer-Songwriter Handbook*, pages 77–89. [Cited on Pages 14, 58, and 59.]

Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A. and Bengio, Y. (2016), SampleRNN: An unconditional end-to-end neural audio generation model, in *arXiv preprint arXiv:1612.07837*. [Cited on Page 49.]

Mengual, L., Moffat, D. and Reiss, J. D. (2016), Modal synthesis of weapon sounds, in *Audio Engineering Society Conference: 61st International Conference: Audio for Games*, Audio Engineering Society. [Cited on Page 49.]

Miller, G. A. (1995), WordNet: a lexical database for English, in *Communications of the ACM*, volume 38, no. 11, pages 39–41. [Cited on Pages 165 and 170.]

Moffat, D. and Reiss, J. D. (2018), Perceptual evaluation of synthesized sound effects, in *ACM Transactions on Applied Perception (TAP)*, volume 15, no. 2, pages 1–19. [Cited on Pages 35 and 49.]

Moffat, D., Ronan, D. and Reiss, J. D. (11 2015), An evaluation of audio feature extraction toolboxes, in *18th International Conference on Digital Audio Effects (DAFx-15)*, Trondheim, Norway. [Cited on Page 56.]

Moffat, D. and Sandler, M. (Oct 2019), Machine Learning Multitrack Gain Mixing of Drums, in *Audio Engineering Society Convention 147*, URL `http://www.aes.org/e-lib/browse.cfm?elib=20550`. [Cited on Page 49.]

Moffat, D., Selfridge, R. and Reiss, J. D. (2019), Sound effect synthesis, in *Foundations in Sound Design for Interactive Media*, pages 274–299, Routledge. [Cited on Pages 38 and 58.]

Moll, P., Frick, V., Rauscher, N. and Lux, M. (2020), How players play games: observing the influences of game mechanics, in *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems*, pages 7–12. [Cited on Page 37.]

Montecchio, N. and Cont, A. (October 2011), Accelerating the Mixing Phase in Studio Recording Productions by Automatic Audio Alignment, in *International Symposium on Music Information Retrieval (ISMIR)*, pages 627–632, Miami, United States, URL `https://hal.archives-ouvertes.fr/hal-01161008`, cote interne IRCAM: Montecchio11b. [Cited on Page 15.]

Moore, A. and Wakefield, J. (2017), An Investigation into the Relationship Between the Subjective Descriptor Aggressive and the Universal Audio 1176 FET Compressor, in . [Cited on Page 49.]

Moore, B. C. J., Glasberg, B. R. and Baer, T. (1997), A Model for the Prediction of Thresholds, Loudness, and Partial Loudness, in *J. Audio Eng. Soc*, volume 45, no. 4, pages 224–240, URL `http://www.aes.org/e-lib/browse.cfm?elib=10272`. [Cited on Pages 19, 20, 21, 22, 38, 122, 174, and 215.]

Müller, M. and Ewert, S. (2011), Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features, in *Proceedings of the 12th International Conference on Music Information Retrieval*. [Cited on Page 56.]

Murphy, K. (2012), *Machine Learning: A Probabilistic Perspective*, The MIT Press. [Cited on Page 23.]

Murtagh, F. and Conreras, P. (2012), Algorithms for hierarchical clustering: an overview, in *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, volume 2, no. 1, pages 86–97. [Cited on Pages 115 and 143.]

Mynett, M., Wakefield, J. and Till, R. (2010), Intelligent equalisation principles and techniques for minimising masking when mixing the extreme modern metal genre, in *Heavy Fundametalisms: Music, metal and Politics*, pages 141–146, Brill. [Cited on Page 10.]

Naderi, B., Möller, S. and Cutler, R. (2021), Speech Quality Assessment in Crowdsourcing: Comparison Category Rating Method, in *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 31–36, IEEE. [Cited on Page 178.]

Nagy, H., Csapo, A. B. and Wersényi, G. (2016), Contrasting results and effectiveness of controlled experiments with crowdsourced data in the evaluation of auditory reaction times, in *2016 7th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 000,421–000,426, doi: 10.1109/CogInfoCom.2016.7804586. [Cited on Page 42.]

Newman, M. E. J. (2018), *Networks*, Oxford University Press. [Cited on Pages 138, 145, 152, and 153.]

Newman, M. E. J., Strogatz, S. H. and Watts, D. J. (Jul 2001), Random graphs with arbitrary degree distributions and their applications, in *Phys. Rev. E*, volume 64, page 026,118, doi: 10.1103/PhysRevE.64.026118, URL https://link.aps.org/doi/10.1103/PhysRevE.64.026118. [Cited on Pages 152 and 164.]

Newton, M. J. and Smith, L. S. (2012), A neurally inspired musical instrument classification system based upon the sound onset, in *The Journal of the Acoustical Society of America*, volume 131, no. 6, pages 4785–4798. [Cited on Page 25.]

Office for National Statistics (May 2015), Internet Users: 2015, Online, URL https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/bulletins/internetusers/2015. [Cited on Page 43.]

Office for National Statistics (May 2020), Internet Users: 2020, Online, URL https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/bulletins/internetusers/2020. [Cited on Page 43.]

Ojanen, M. et al. (2015), Mastering Kurenniemi's Rules (2012): the role of the audio engineer in the mastering process, in *Journal on the Art of Record Production*. [Cited on Page 11.]

Orga, F., Mitchell, A., Freixes, M., Aletta, F., Alsina-Pagès, R. M. and Foraster, M. (2021), Multilevel annoyance modelling of short environmental sound recordings, in *Sustainability*, volume 13, no. 11, page 5779. [Cited on Page 49.]

Owsinski, B. (2017), *The Mixing Engineer's Handbook*, Bobby Owsinski Media Group, 4th edition. [Cited on Pages 6, 7, and 53.]

Parizet, E. and Nosulenko, V. (1999), Multi-dimensional listening test: Selection of sound descriptors and design of the experiment, in *Noise Control Engineering*, volume 47. [Cited on Page 40.]

Patterson, R. D., Nimmo-Smith, I., Weber, D. L. and Milroy, R. (1982), The deterioration of hearing with age: Frequency selectivity, the critical ratio, the audiogram, and speech threshold, in *The Journal of the Acoustical Society of America*, volume 72, no. 6, pages 1788–1803. [Cited on Page 21.]

Peeters, G. (4 2004), A large set of audio features for sound description (similarity and classification) in the CUIDADO project. [Cited on Page 194.]

Perez-Gonzalez, E. and Reiss, J. (Oct 2009), Automatic Equalization of Multichannel Audio Using Cross-Adaptive Methods, in *Audio Engineering Society Convention 127*, URL http://www.aes.org/e-lib/browse.cfm?elib=15026. [Cited on Pages 5, 15, 17, 18, 23, and 106.]

Perez-Gonzalez, E. and Reiss, J. D. (10 2009), Automatic Gain and Fader Control For Live Mixing, in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA. [Cited on Page 207.]

Pestana, P. D., Ma, Z., Reiss, J. D., Barbosa, A. and Black, D. A. (2013), Spectral characteristics of popular commercial recordings 1950-2010, in *Audio Engineering Society Convention 135*, Audio Engineering Society. [Cited on Pages 7, 50, 109, 127, and 214.]

Pestana, P. D., Reiss, J. D. et al. (Jan 2014), Intelligent audio production strategies informed by best practices, in *53rd International Conference of the AES*. [Cited on Pages 8, 36, 58, and 107.]

Pras, A., De Man, B. and Reiss, J. D. (2018), A case study of cultural influences on mixing practices, in *Audio Engineering Society Convention 144*, Audio Engineering Society. [Cited on Page 8.]

Pruuvsa, Z. and Holighaus, N. (2017), Phase vocoder done right, in *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 976–980, IEEE. [Cited on Page 49.]

Purshouse, R. C. and Fleming, P. J. (2002), Why use elitism and sharing in a multi-objective genetic algorithm?, in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary computation*, pages 520–527. [Cited on Page 28.]

Putnam, M. T. (5 1980), A Thirty-five Year History And Evolution of the Recording Studio, in *66th Audio Engineering Society Convention*, Los Angeles, CA, USA. [Cited on Page 53.]

Ramírez, M. M., Benetos, E. and Reiss, J. D. (2020), Deep learning for black-box modeling of audio effects, in *Applied Sciences*, volume 10, no. 2, page 638. [Cited on Page 49.]

Razali, N. M., Geraghty, J. et al. (2011), Genetic algorithm performance with different selection strategies in solving TSP, in *Proceedings of the world congress on engineering*, volume 2, pages 1–6, International Association of Engineers Hong Kong, China. [Cited on Page 28.]

Reid, J. (2015), HTML5 API Reference, in *HTML5 Programmer's Reference*, pages 285–307, Springer. [Cited on Page 56.]

Reips, U.-D. (2002), Standards for Internet-based experimenting, in *Experimental psychology*, volume 49, no. 4, pages 243–256. [Cited on Page 42.]

Reiss, J. D. (7 2011), Intelligent systems for mixing multichannel audio, in *Digital Signal Processing (DSP), 2011 17th International Conference on*, IEEE, Corfu, Greece. [Cited on Pages ix, 5, 15, 16, and 55.]

Reiss, J. D. (2016), A meta-analysis of high resolution audio perceptual evaluation, in *Journal of the Audio Engineering Society*, volume 64, no. 6, pages 364–379. [Cited on Pages 41 and 237.]

RIAA (2020), U.S. Sales Database: Recorded Music Revenues by Format, URL `https://www.riaa.com/u-s-sales-database/`, accessed 27th July 2021. [Cited on Pages ix, 43, and 44.]

Ronan, D., Moffat, D., Gunes, H. and Reiss, J. D. (Dec 2015a), Automatic subgrouping of multitrack audio, in *18th International Conference on Digital Audio Effects (DAFx-15)*, Trondheim, Norway. [Cited on Pages xii, xiii, 125, 132, 146, 147, 148, 149, 154, 155, 170, and 213.]

Ronan, D. M., De Man, B., Gunes, H. and Reiss, J. D. (10 2015b), The impact of subgrouping practices on the perception of multitrack mixes, in *139th Convention of the Audio Engineering Society*, New York, NY, USA. [Cited on Pages 5, 38, 88, 89, 126, 129, 131, 132, 161, 170, 171, and 214.]

Rosenbrock, H. H. (01 1960), An Automatic Method for Finding the Greatest or Least Value of a Function, in *The Computer Journal*, volume 3, no. 3, pages 175–184, ISSN 0010-4620, doi: $10.1093/\mathrm{comjnl}/3.3.175$, URL `https://doi.org/10.1093/comjnl/3.3.175`. [Cited on Pages 31 and 202.]

Rouse Ball, W. (1960), The Eight Queens Problem, in *Mathemtical Recreations and Essays*. [Cited on Page 26.]

Sanfeliu, A. and Fu, K.-S. (1983), A distance measure between attributed relational graphs for pattern recognition, in *IEEE Transactions on Systems, Man, and Cybernetics*, volume SMC-13, no. 3, pages 353–362, doi: $10.1109/\mathrm{TSMC}.1983.6313167$. [Cited on Page 147.]

Sauer, C., Roth-Berghofer, T., Auricchio, N. and Proctor, S. (2013), Recommending audio mixing workflows, in *International Conference on Case-Based Reasoning*, pages 299–313, Springer. [Cited on Pages 9 and 50.]

Schaeffer, S. E. (2007), Graph clustering, in *Computer science review*, volume 1, no. 1, pages 27–64. [Cited on Page 139.]

Schmitt, L. M. (2001), Theory of genetic algorithms, in *Theoretical Computer Science*, volume 259, no. 1-2, pages 1–61. [Cited on Page 27.]

Schoeffler, M., Stöter, F.-R., Bayerlein, H., Edler, B. and Herre, J. (November 2013), An Experiment about Estimating the Number of Instruments in Polyphonic Music: A Comparison Between Internet and Laboratory Results, in *14th International Society for Music Information Retrieval Conference (ISMIR 2013)*. [Cited on Pages 42 and 178.]

Schoeffler, M., Stöter, F.-R., Edler, B. and Herre, J. (2015), Towards the Next Generation of Web-based Experiments: A Case Study Assessing Basic Audio Quality Following the ITU-R Recommendation BS.1534 (MUSHRA), in *1st Web Audio Conference*, Paris, France. [Cited on Page 45.]

Selfridge, R., Moffat, D., Avital, E. J. and Reiss, J. D. (2018), Creating real-time aeroacoustic sound effects using physically informed models, in *Journal of the Audio Engineering Society*. [Cited on Page 49.]

Selfridge, R., Moffat, D. and Reiss, J. D. (2017a), Physically derived sound synthesis model of a propeller, in *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*, pages 1–8. [Cited on Pages 49 and 64.]

Selfridge, R., Moffat, D. and Reiss, J. D. (2017b), Sound synthesis of objects swinging through air using physical models, in *Applied Sciences*, volume 7, no. 11, page 1177. [Cited on Pages 49 and 237.]

Senior, M. (2019), *Mixing Secrets for the Small Studio*, Focal Press, 2nd edition. [Cited on Pages xv, 3, 6, 66, 116, and 211.]

Simpson, A. J., Terrell, M. J. and Reiss, J. D. (2013), A practical step-by-step guide to the time-varying loudness model of Moore, Glasberg, and Baer (1997; 2002), in *Audio Engineering Society Convention 134*, Audio Engineering Society. [Cited on Pages 20 and 21.]

Soulodre, G. A. and Lavoie, M. C. (Sep 1999), Subjective Evaluation of Large and Small Impairments in Audio Codecs, in *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. [Cited on Page 40.]

Spratley, S., Beck, D. and Cohn, T. (2019), A unified neural architecture for instrumental audio tasks, in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 461–465, IEEE. [Cited on Page 49.]

Stables, R., Enderby, S., De Man, B., Fazekas, G. and Reiss, J. D. (2014), SAFE: A system for the extraction and retrieval of semantic audio descriptors, in *15th Intl. Soc. Music Inf. Retr. (ISMIR)*. [Cited on Pages x, 9, 51, 53, 57, and 212.]

Stables, R., Hockman, J. and Southall, C. (2016), Automatic Drum Transcription using Bi-directional Recurrent Neural Networks., in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, ISMIR. [Cited on Page 25.]

Stasis, S., Jillings, N., Enderby, S. and Stables, R. (2017a), Audio processing chain recommendation, in *Proceedings of the 20th International Conference on Digital Audio Effects*. [Cited on Pages 4, 57, and 212.]

Stasis, S., Jillings, N., Enderby, S. and Stables, R. (2017b), Audio processing chain recommendation using semantic cues, in *Proceedings of the 3rd Workshop on Intelligent Music Production*. [Cited on Pages 4 and 57.]

Stasis, S., Stables, R. and Hockman, J. (2016), Semantically Controlled Adaptive Equalisation in Reduced Dimensionality Parameter Space, in *Applied Sciences*, volume 6, no. 4, ISSN 2076-3417, doi: $10.3390/\mathrm{app6040116}$, URL `http://www.mdpi.com/2076-3417/6/4/116`. [Cited on Pages 5, 16, and 53.]

Steinmetz, C. J., Pons, J., Pascual, S. and Serrà, J. (2021), Automatic multitrack mixing with a differentiable mixing console of neural audio effects, in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 71–75, IEEE. [Cited on Pages 25 and 215.]

Stephenson, B., Besacier, L., Girin, L. and Hueber, T. (2020), What the future brings: Investigating the impact of lookahead for incremental neural TTS, in *arXiv preprint arXiv:2009.02035*. [Cited on Page 49.]

Stickland, S., Athauda, R. and Scott, N. (2019), Design of a real-time multiparty DAW collaboration application using Web MIDI and WebRTC APIs, in *Proceedings of the International Web Audio Conference*, pages 59–64. [Cited on Page 15.]

Stickland, S., Scott, N. and Athauda, R. (2018), A Framework for Real-Time Online Collaboration in Music Production, in *Proceedings of the ACMC2018: Conference of the Australasian Computer Music Association*, pages 79–86. [Cited on Page 15.]

Sutton, J. and Austin, Z. (2015), Qualitative research: Data collection, analysis, and management, in *The Canadian journal of hospital pharmacy*, volume 68, no. 3, page 226. [Cited on Pages 37 and 64.]

Tabak, J. (2004), *Geometry: The Language of Space and Form*, Facts on File, Inc. [Cited on Page 115.]

Terrell, M., Simpson, A. and Sandler, M. (2014), The mathematics of mixing, in *Journal of the audio engineering society*, volume 62, no. 1/2, pages 4–13. [Cited on Pages 7, 15, and 129.]

Tervo, S., Laukkanen, P., Pätynen, J. and Lokki, T. (2014), Preferences of critical listening environments among sound engineers, in *Journal of the Audio Engineering Society*, volume 62, no. 5, pages 300–314. [Cited on Page 8.]

Tom, A., Reiss, J. and Depalle, P. (March 2019), An automatic mixing system for multitrack spatialization for stereo based on unmasking and best panning practices, in *Audio Engineering Society Convention 146*, URL `https://www.aes.org/e-lib/browse.cfm?elib=20311`. [Cited on Page 18.]

Tomczak, M., Southall, C. and Hockman, J. (2018), Audio style transfer with rhythmic constraints, in *Proceedings of the 21st Digital Audio Effects (DAFx)*. [Cited on Page 25.]

Toole, F. E. (1982), Listening Tests—Turning Opinion into Fact, in *Journal of the Audio Engineering Society*, volume 30, no. 6, pages 431–445. [Cited on Page 39.]

Torija, A. J. and Nicholls, R. K. (2022), Investigation of metrics for assessing human response to drone noise, in *International Journal of Environmental Research and Public Health*, volume 19, no. 6, page 3152. [Cited on Page 49.]

VanDam, M., Warlaumont, A. S., Bergelson, E., Cristia, A., Soderstrom, M., De Palma, P. and MacWhinney, B. (2016), HomeBank: An online repository of daylong child-centered audio recordings, in *Seminars in speech and language*, volume 37, pages 128–142, Thieme Medical Publishers. [Cited on Page 37.]

Verfaille, V., Zölzer, U. and Arfib, D. (10 2006), Adaptive Digital Audio Effects (A-DAFx): A new class of sound transformations, in *Audio, Speech, and Language Processing, IEEE Transactions on*, volume 14, pages 1817 – 1831, doi: 10.1109/TSA.2005.858531. [Cited on Pages 16 and 53.]

Vincent, E., Jafari, M. and Plumbley, M. (2006), Preliminary guidelines for subjective evalutation of audio source separation algorithms, in *UK ICA Research Network Workshop*. [Cited on Pages xv and 47.]

Wakefield, J. and Dewey, C. (2015), An investigation into the efficacy of methods commonly employed by mix engineers to reduce frequency masking in the mixing of multitrack musical recordings, in *Audio Engineering Society Convention 138*, Audio Engineering Society. [Cited on Pages 10, 18, 50, 53, and 211.]

Walton, T., Evans, M., Kirk, D. and Melchior, F. (2018), Exploring object-based content adaptation for mobile audio, in *Personal and Ubiquitous Computing*, volume 22, no. 4, pages 707–720. [Cited on Page 49.]

Wang, R. Y. and Popović, J. (2009), Real-time hand-tracking with a color glove, in *ACM transactions on graphics (TOG)*, volume 28, no. 3, pages 1–8. [Cited on Page 37.]

Ward, D., Athwal, C. and Kököer, M. (Oct 2013), An efficient time-varying loudness model, in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, ISSN 1931-1168, doi: 10.1109/WASPAA.2013.6701884. [Cited on Page 19.]

Ward, D., Reiss, J. D. and Athwal, C. (10 2012), Multi-track mixing using a model of loudness and partial loudness, in *133rd Convention of the Audio Engineering Society*, San Francisco, CA, USA. [Cited on Pages ix, 17, 22, 122, 174, 199, 201, and 207.]

Watanabe, T., Hashimoto, Y., Nishikawa, I. and Tokumaru, H. (1995), Line balancing using a genetic evolution model, in *Control Engineering Practice*, volume 3, no. 1, pages 69 – 76, ISSN 0967-0661, doi: http://dx.doi.org/10.1016/0967-0661(94)00066-P, URL http://www.sciencedirect.com/science/article/pii/096706619400066P. [Cited on Page 27.]

Wegel, R. L. and Lane, C. E. (Feb 1924), The Auditory Masking of One Pure Tone by Another and its Probable Relation to the Dynamics of the Inner Ear, in *Phys. Rev.*, volume 23, pages 266–285, doi: 10.1103/PhysRev.23.266, URL https://link.aps.org/doi/10.1103/PhysRev.23.266. [Cited on Page 18.]

White, P. (11 2012), Line 6 StageScape M20D, in *Sound on Sound*, URL https://www.soundonsound.com/reviews/line-6-stagescape-m20d. [Cited on Page 35.]

Wichern, G., Wishnick, A., Lukin, A. and Robertson, H. (2015), Comparison of loudness features for automatic level adjustment in mixing, in *Audio Engineering Society Convention 139*, Audio Engineering Society. [Cited on Page 17.]

Wickelmaier, F., Umbach, N., Sering, K. and Choisel, S. (May 2009), Comparing Three Methods for Sound Quality Evaluation with Respect to Speed and Accuracy, in *Audio Engineering Society Convention 126*. [Cited on Page 242.]

Wierstorf, H., Ward, D., Mason, R., Grais, E. M., Hummersone, C. and Plumbley, M. D. (2017), Perceptual evaluation of source separation for remixing music, in *Audio Engineering Society Convention 143*, Audio Engineering Society. [Cited on Page 49.]

Wilcoxon, F. (1945), Individual Comparisons by Ranking Methods, in *Biometrics Bulletin*, volume 1, no. 6, pages 80–83, ISSN 00994987, URL http://www.jstor.org/stable/3001968. [Cited on Pages 74 and 186.]

Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E. and Adami, C. (July 2001), Evolution of digital organisms at high mutation rates leads to survival of the fittest, in , volume 412, no. 1, pages 331 – 333. [Cited on Page 28.]

Wilson, A. and Fazenda, B. (Oct 2015a), 101 Mixes: A Statistical Analysis of Mix-Variation in a Dataset of Multi-Track Music Mixes, in *Audio Engineering Society Convention 139*, URL http://www.aes.org/e-lib/browse.cfm?elib=17955. [Cited on Pages 53 and 68.]

Wilson, A. and Fazenda, B. (July 2015b), Navigating The Mix-space: Theoretical And Practical Level-balancing Technique In Multitrack Music Mixtures, in *Proceedings of the 12th Sound and Music Computing Conference*, Sound and Music Computing. [Cited on Pages 7, 66, 101, 113, 125, 193, 208, and 212.]

Wilson, A. and Fazenda, B. M. (12 2017), Populating the Mix Space: Parametric Methods for Generating Multitrack Audio Mixtures, in *Applied Sciences*, volume 17, doi: doi:10.3390/app7121329. [Cited on Pages 23 and 200.]

Worcester, R. (1996), Political polling: 95 expertise and 5 luck, in *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, volume 159, no. 1, pages 5–20. [Cited on Page 36.]

World Wide Web Consortium (September 2018), *Web Audio API, W3C Candidate Recommendation*, uRL `https://www.w3.org/TR/webaudio/`. [Cited on Pages 44, 45, 59, and 64.]

World Wide Web Consortium (November 2019), *High Resolution Time Level 2, W3C Candidate Recommendation*, URL `https://www.w3.org/TR/hr-time-2/`. [Cited on Page 57.]

Wu, J. C., Das, O. and DiPasquale, V. (2019), A Comparative Pilot Study and Analysis of Audio Mixing Using Logic Pro X and GarageBand for iOS, in *Audio Engineering Society Convention 147*, Audio Engineering Society. [Cited on Pages 9 and 58.]

Yan, X., Yang, L., Lan, S. and Tong, X. (2012), Application of HTML5 multimedia, in *2012 International Conference on Computer Science and Information Processing (CSIP)*, pages 871–874, IEEE. [Cited on Page 44.]

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q. and Artzi, Y. (2019), BERTScore: Evaluating Text Generation with BERT, doi: 10.48550/ARXIV.1904.09675, URL `https://arxiv.org/abs/1904.09675`. [Cited on Pages 170, 171, and 215.]

Zwicker, E. (1965), Temporal Effects in Simultaneous Masking by White-Noise Bursts, in *Journal of the Acoustical Society of America*, volume 37, no. 4. [Cited on Page 19.]

Zwicker, E. and Scharf, B. (1965), A model of loudness summation, in *Psuchological Review*, volume 72, no. 1, pages 3–26. [Cited on Page 19.]

Appendix A

Listening Test Design

Due to the variation of listening test design on the impact of results, a study was performed to compare two common research questions with two interface tests. The variances between the studies would be minimised to only pertain to the variations of the interface types. Each of the MUSHRA and Pairwise (AB) tests were performed on two common test questions: quality evaluation and realism. The quality trial focuses on the subject being able to provide an accurate description or evaluation of a set of processed audio files. Commonly these quality trials are used to verify the performance of the encoding technique with various content types (Reiss, 2016). For this trial, a single castanet recording was processed through four filters to provide 5 total files:

- The unprocessed reference
- Low passed to 14kHz
- Low passed to 7kHz
- Low passed to 3.5kHz
- Low passed to 1.75kHz

This would provide 4 different audio files with a predictable outcome. The reference should score the highest, with each of the low-passed filtered sources scoring lower as the bandwidth narrows. The question presented to the user was "Which of these has the highest quality?".

The realism trial focuses on the subjective evaluation of the synthesised golf swing, generated from Selfridge et al. (2017b). The synthesis system provided a version generated by a physical model (PM) and spectral modelling synthesis (SMS). The anchor was a badly synthesised "swoosh" sound and a reference of a close-mic pick-up of a real golf swing. All these stimuli were taken from the study performed by Selfridge et al. (2017b). This again should provide a predictable measurement, not least because the published paper showed a given result, but because the reference should score the highest, the incorrect anchor the lowest and the two models in the middle. The question presented to the subject was "Which of these is the most realistic golf club swing?".

	AB	MUSHRA
Realism	57 (8)	55 (6)
Quality	66 (21)	48 (0)

Table A.1: The total number of submissions of the four tests with the number of total abandoned tests in brackets.

A	B	#A	#B	%A	%B
1.75 kHz	3.5 kHz	8	46	14.81%	85.18%
1.75 kHz	7 kHz	2	48	4.00%	96.00%
1.75 kHz	14 kHz	0	50	0.00%	100.00%
1.75 kHz	Ref.	2	48	4.00%	96.00%
3.5 kHz	7 kHz	2	49	3.92%	96.08%
3.5 kHz	14 kHz	1	48	2.04%	97.96%
3.5 kHz	Ref.	1	47	2.08%	97.92%
7 kHz	14 kHz	3	45	6.25%	93.75%
7 kHz	Ref.	2	46	4.17%	95.83%
14 kHz	Ref.	26	24	52.00%	48.00%

Table A.2: All submissions for the *Quality* trial using the AB method.

Both the AB pairwise and MUSHRA vertical slider interfaces are supported, and it has support for references, randomisation and the data collected. Each participant would be presented with one of the tasks, either AB or MUSHRA. Once they finished the first task, they would be given the second task in the alternative interface. This is because the order of the realism and quality study must be randomised too to reduce bias. For example, if a subject was presented the Quality test with MUSHRA first, they would then complete the Realism test using AB.

A total of 231 tests were collected when the study was online, of which 85 were linked tests with both a primary and secondary test completed by the same user. This meant 61 subjects only did one of the pages to completion. A total of 35 tests were abandoned before they were completed. A breakdown of the tests can be seen in Table A.1.

In the case of the AB interface, Table A.2 shows all *Quality* trials. The A and B were randomised, but the expected result of the table is that the 'B' stimuli would be the preferred option. This held true except for the 14kHz band limited versus the full band reference, where only 48.00% of trial respondents successfully identified it as the superior quality. This is not wholly a bad result, since the quality difference between the two is very limited. Factors such as the listening environment, audio equipment used and any hearing impairments of the subject could all lead to a situation where the higher frequencies are not correctly identified, making the distinction between the two difficult. The 48.00% score shows that this is in the range of random probability and that there was no significant correct identification of the higher quality source.

A	B	# A	# B	% A	% B
Anch.	PM	3	50	5.66%	94.34%
Anch.	SMS	9	42	17.65%	82.35%
Anch.	Wood	2	50	3.85%	96.15%
Anch.	Ref.	3	50	5.66%	94.34%
PM	SMS	43	9	82.69%	17.31%
PM	Wood	24	25	48.98%	51.02%
PM	Ref.	21	28	42.86%	57.14%
SMS	Wood	7	43	14.00%	86.00%
SMS	Ref.	9	41	18.00%	82.00%
Wood	Ref.	18	33	35.29%	64.71%

Table A.3: All submissions for the *Realism* trial using the AB method.

Sample	25th perc.	50th perc.	75th perc.
Anchor	0.00%	8.37%	8.50%
SMS	4.00%	23.20%	31.50%
PM	28.00%	50.20%	76.00%
Wood	52.00%	66.74%	78.00%
Ref.	85.50%	90.88%	100.00%

Table A.4: Results for the *Realism* trial using the MUSHRA method

For the *Realism* trial, TableA.3 shows fewer trials were significantly different, meaning it would require more trials to obtain the differences, or a better question to be asked. For instance, whilst everyone correctly identified that the anchor was not realistic, the reference was not always as easy to identify as the most realistic. This indicates the AB test cannot be used for relatively subjective, small difference testing, but is suitable for other, more binary based analysis questions.

Tables A.5 and A.4 show the results for the MUSHRA tests of the *Quality* and *Realism* studies respectively. The Realism study shows far greater confidence in the scores, with minimal statistical overlap between the levels. The anchor is correctly identified as being at the bottom, along with Spectral Model Synthesis (SMS) being a very poor performer. The range of the Physical Model is high showing there is still disagreement on exactly how well it performs, with the PM and Wood overlapping quite significantly. But the AB test also shows that these are virtually impossible to separate anyway, with clearly very similar performance. The AB gave 48.98% to PM and 51.02% to wood, whilst the MUSHRA test gives an average of 50.20% to PM and 66.74% to wood. Whilst not enough to be significant, this is a better separation. For the *Quality* trials, there is a lacking of clarity between the 1.75 kHz and 3.50 kHz bands, and the 14 kHz and Reference bands. The only subject which was significantly different is the 7 kHz band, sitting around the 44.67% mark. The 1.75kHz and 3.50kHz have similarly overlapping samples because of the use of the sample scales. None of the participants

Sample	25th perc.	50th perc.	75th perc.
1.75 kHz	0.00%	11.10%	19.75%
3.5 kHz	7.25%	21.02%	29.75%
7 kHz	32.50%	44.67%	52.00%
14 kHz	72.00%	83.44%	100.00%
Ref.	76.25%	88.40%	100.00%

Table A.5: Results for the *Quality* trial using the MUSHRA method

Sample	AB	MUSHRA
1.7 kHz	1.407 (3.47s)	2.688 (7.72s)
3.5 kHz	1.496 (4.12s)	2.812 (7.52s)
7 kHz	1.470 (3.83s)	3.062 (9.74s)
14 kHz	1.977 (5.75s)	5.417 (15.05s)
Ref.	1.949 (5.51s)	5.458 (14.53s)

Table A.6: Fragment listens per page for the *Quality* trial using the MUSHRA and AB methods

actually did place the 3.50kHz sample below the 1.75kHz sample. But some listeners used more of the scale than others, so when placed into a distribution the ranges can overlap.

Not sure this part of the paper actually helps

The Web Audio Evaluation Tool collects the timing of audition, click, and drag events by default, as well as the total duration of each page and complete test. The test duration is an indicator of the effort required for each test to complete. A shorter test requires less effort from the user and therefore can indicate reduced loading. Conversely, a long test, acoompanied with large numbers of playback counts and movements, indicates a higher loading. Figure A.1 shows the Histograms for the 4 tests used. The *Quality* tests generally took longer than the *Realism* tests, which is of interest due to the fact the tests themselves did not have longer audio files or different numbers. So this difference is due just to the question being posed. The AB tests were completed, on average, after 100 seconds in total compared to 60 seconds for MUSHRA. This indicates the MUSHRA tests were faster to complete overall, however the AB required the user to switch pages to navigate through the

Sample	AB	MUSHRA
Anchor	1.469 (0.85s)	2.714 (1.57s)
PM	1.792 (0.83s)	3.959 (1.84s)
SMS	1.693 (0.51s)	3.408 (1.03s)
Wood	1.792 (0.55s)	4.204 (1.29s)
Ref.	1.763 (0.79s)	3.735 (1.68s)

Table A.7: Fragment listens per page for the *Realism* trial using the MUSHRA and AB methods

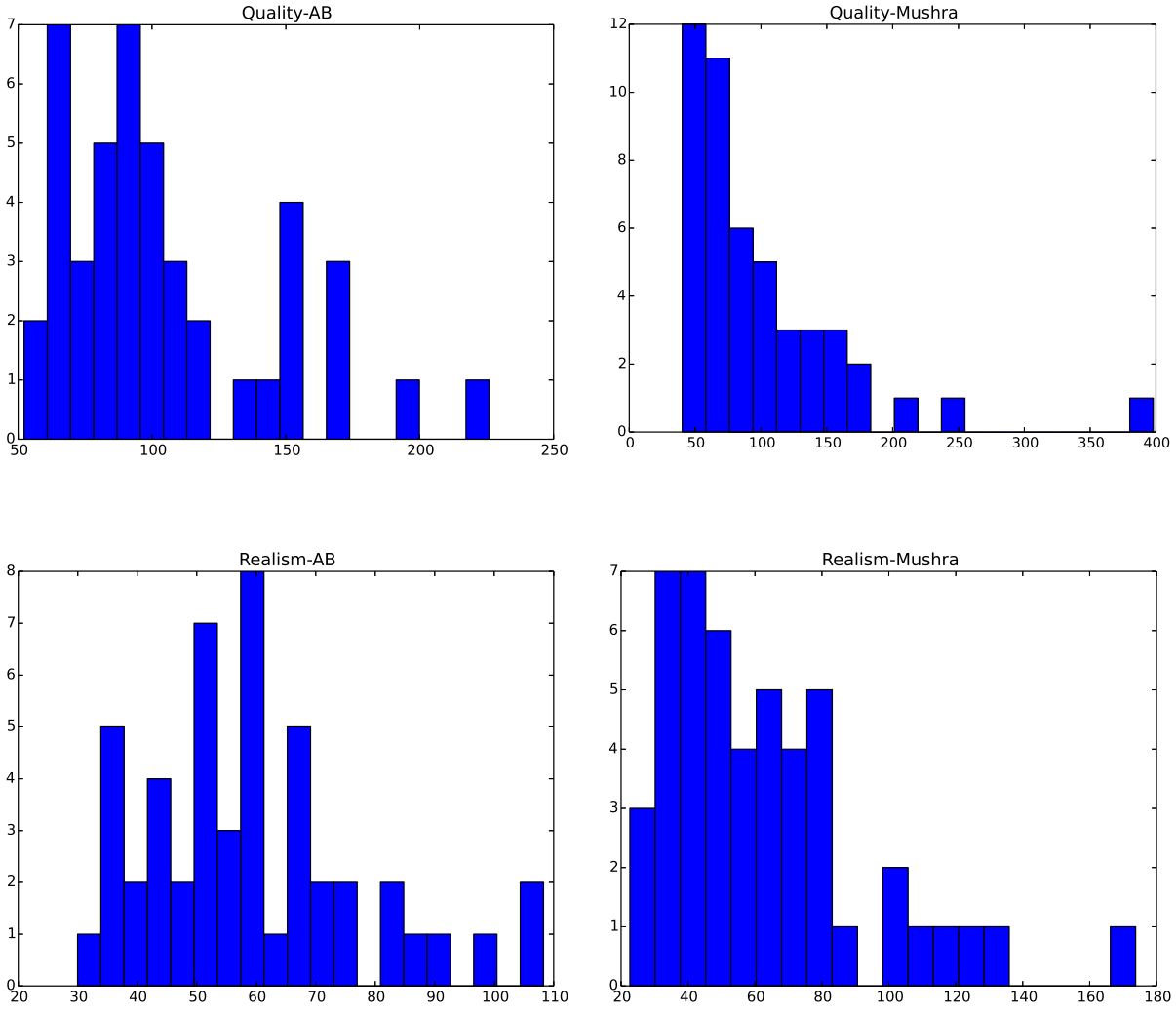


Figure A.1: Histogram of test durations for the different tests

tests. This would show that the individual comparison time may be comparable but the actual wall clock test time is longer due to the added steps involved.

The MUSHRA format results in more listens per page as the subjects can compare freely any stimuli they wish to in order to drive their decision. The subjects can learn which sources are in which position and then divide the test up to minimise the workload, indicating the importance to randomise the order to remove this potential training bias. The AB format results in more evaluations and listens per tests, as subjects have to constantly evaluate new pairs without prior knowledge of what the pair is before the page is shown. Table A.6 gives the trial average number of auditions, and the average duration, for the *Quality* trial. The MUSHRA test shows the increase in effort taken by users to compare the 14 kHz and Reference samples with both of these requring, on average, 2 more listens each time, and for up to double the amount of time. Compared to the AB which is far more uniform, although these do both still show higher audition and listen time counts. This is most likely due to the combination when the two are directly compared. Likewise in the *Realism* trial in Table A.7 the AB and MUSHRA examples are fairly even throughout each sample, showing the subjects could identify the solution quickly with minimal effort in both.

The results above show that, given the same question and samples, the conclusions taken from a listening study can be influenced by the test interface type. MUSHRA places more effort per page shown to complete, with generally more reliable results. The results clearly demonstrate the subject behaviour is markedly different in both tests, with the MUSHRA test being completed at a faster rate than the AB study, and with less effort per comparison as confirmed by previous studies (De Man and Reiss, 2013b; Wickelmaier et al., 2009). The AB test is able to quickly discern larger variances in test material, but at the detriment to small difference comparisons. MUSHRA can also be influenced heavily by the continuous scale, where users will adjust and drift across the scale, whilst the binary nature of the AB forces a selection in favour of one or the other.