MUSIC VISUALIZATION USING SOURCE SEPARATED STEREOPHONIC MUSIC

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Engineering in Electrical Engineering

by

Hannah Chookaszian

June 2022

COMMITTEE MEMBERSHIP

TITLE: Music Visualization Using Source Separated Stereophonic Music

AUTHOR: Hannah Chookaszian

DATE SUBMITTED: June 2022

COMMITTEE CHAIR: Wayne Pilkington, Ph.D.
Professor of Electrical Engineering

COMMITTEE MEMBER: Jane Zhang, Ph.D.
Professor of Electrical Engineering

COMMITTEE MEMBER: Dennis Sun, Ph.D.
Professor of Statistics

ABSTRACT

Music Visualization Using Source Separated Stereophonic Music

Hannah Chookaszian

This thesis introduces a music visualization system for stereophonic source separated music. Music visualization systems are a popular way to represent information from audio signals through computer graphics. Visualization can help people better understand music and its complex and interacting elements. This music visualization system extracts pitch, panning, and loudness features from source separated audio files to create the visual. Most state-of-the art visualization systems develop their visual representation of the music from either the fully mixed final song recording, where all of the instruments and vocals are combined into one file, or from the digital audio workstation (DAW) data containing multiple independent recordings of individual audio sources. Original source recordings are not always readily available to the public so music source separation (MSS) can be used to obtain estimated versions of the audio source files. This thesis surveys different approaches to MSS and music visualization as well as introduces a new music visualization system specifically for source separated music.

ACKNOWLEDGMENTS

Thanks to:

- Professor Pilkington, for the idea for this thesis and helping me throughout this process
- My parents, for always supporting my endeavors
- Andrew Guenther, for uploading this template

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CODE LISTINGS

Chapter 1

INTRODUCTION

1.1 Statement of Problem

Music information retrieval (MIR) and music visualization systems have been a topic of research for many years in the audio processing community [12]. Visualization approaches have improved over time by focusing on different relevant features in a musical performance and meaningful ways to represent them. Most state-of-the art visualization systems develop their visual representation of the music from either the fully mixed final song recording, where all of the instruments and vocals are combined into one file, or from the digital audio workstation (DAW) data containing multiple independent recordings of individual audio sources. While eaiser to implement, visualization systems based on DAW data are quite limiting because the original DAW files for music, especially popular music, is often unavailable to the public. This is unfortunate since independently rendering each music source and their changes in volume, pitch, and timbre can help create an effective visualization to enhance the listener's understanding of how different musical elements in a song are interating to create the full subjective experience of a song. Using music source separation (MSS), the primary sources that create a song from the more readily available mixed song files can be obtained for any piece of music. Introducing MSS into a music visualization project creates an end-to-end system that visualizes information before the mixing occurs.

1.2 List of Terms

Digital signals processing, music source separation, music information retrieval, music information visualization.

1.3 Purpose of Study

Although many different music visualization methods have been introduced in the past years, many do not offer visualization of individual sources without the original DAW information. This study looks at using music source separation and its subsequent source tracks to make a unique and customizable music visualization system.

Chapter 2

BACKGROUND

2.1 Stereophonic vs. Monophonic Audio

Monophonic (mono) audio is a type of audio recording that utilizes one waveform or channel to reproduce all the sounds in the recording. This one channel is then played across however many speakers the listener chooses to use. Mono audio signals are highly correlated because all instruments in the mix are played over the same channel. Stereophonic (stereo) audio is another type of audio which has been popular since the 1950s, and remains the dominant format for music recording and distribution. Stereo audio uses two channels played in parallel to create the illusion of spatial sound. This requires two or more speakers with each speaker playing the music information from one channel. When listening to stereo recordings, directional sounds are perceived across the horizontal plane in front of the listener with the ideal location for listening being in the middle of the two speakers.

Stereo audio can be produced by recording a source with two microphones or stereo panning a single source recording track in post-processing. Stereo panning creates the illusion of the sound coming from a certain place on the horizontal plane. The direction that the sound is perceived to arrive from is called the phantom image. Stereo panning can be created through level changes or delays in the different channels. Level based panning involves creating differences in channel amplitudes by attenuating one channel. The phantom image shifts towards the opposite channel that the attenuation is applied to. Delay based panning is created by adding a delay to one channel of the audio which will shift the image away from the side that has the delay

[11]. Stereo panning takes advantage of how our brains subconsciously determine a sound source's location based on amplitude and arrival time difference in the same sounds detected by our two ears.

2.2 Music Representation

It is important to understand the different ways to represent music in order to perform music source separation and visualization. Music can be represented in the form of a musical score on paper, an electronic instrument through Musical Instrument Digital Interface (MIDI), or as an acoustic sound wave. Performance and recording of music results in sounds which are transmitted through the air and received as pressure oscillations. The transmission, receptions, and reproduction of sounds that lie within the amplitude and frequency limits of human hearing is called audio.

Since this thesis focuses on recorded music, the main representation of music will be through acoustic sound waves or audio. Audio signals are able to completely encode all the information necessary to reproduce a piece of music, but information such as pitch and onset times are not explicitly provided. Musical audio signals are also typically a mixture of different instruments and voices which are overlaid. This mixture that is present in audio signals makes processing more difficult because the individual instruments are highly correlated; as they are following the same rhythm patterns and playing notes with the same or harmonically-related frequencies.

When evaluating an audio representation, it is very important to understand the properties associated with audio signals. The frequency, pitch, dynamics, intensity, loudness, and timbre can all be examined in an audio representation. Frequency and pitch are closely related in pure tones because the frequency can be matched exactly

to a pitch. The frequency can be measured in Hertz (Hz) and is equal to the reciprocal of the period. Figure 2.1 shows the important features of a waveform.

In complex and highly correlated mixtures, the relation between frequency and pitch is much more complicated and therefore it is useful to separate mixtures into individual sources. Even with separation, the source waveforms are still complex and the sound from a musical source will not be a pure tone. The musical tone heard in a song is a superposition of multiple pure tones which each have their own frequency. These different sinusoids which are mixed together are referred to as partials. The lowest partial present in a waveform is called the fundamental frequency and this frequency determines the musical pitch of the sound.

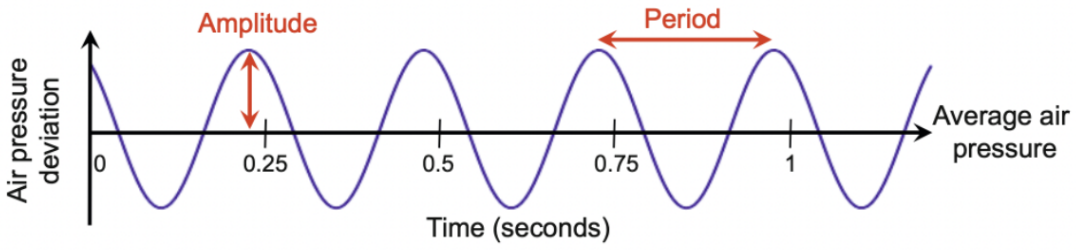

Figure 2.1: Waveform With a Frequency of 4 Hz [1]

Dynamics is a general term to describe the volume and musical symbols associated with that volume. In audio, dynamics is related to the loudness of the audio signal. The loudness is a measure of both the sound intensity and sound power. The intensity is defined in watts (W) and denotes the sound power per unit area while the sound power denotes the energy per unit time from the sound source. It is important to note that loudness can be affected by different factors such as the duration of the sound, the person listening, and the frequency of the sound. Generally, the perceived loudness of a sound is different depending on the individual and age can factor into this because of the change in human ear sensitivity over time. Additionally, longer sounds are perceived louder than sounds that last less time and higher frequencies

are perceived to be louder than lower frequencies. The perceived loudness of pure tones can be seen on the equal loudness contours in Figure 2.2. The unit *phon* is used which is a normalized unit for the intensity level, measured in dB with respect to tones at 1 kHz.

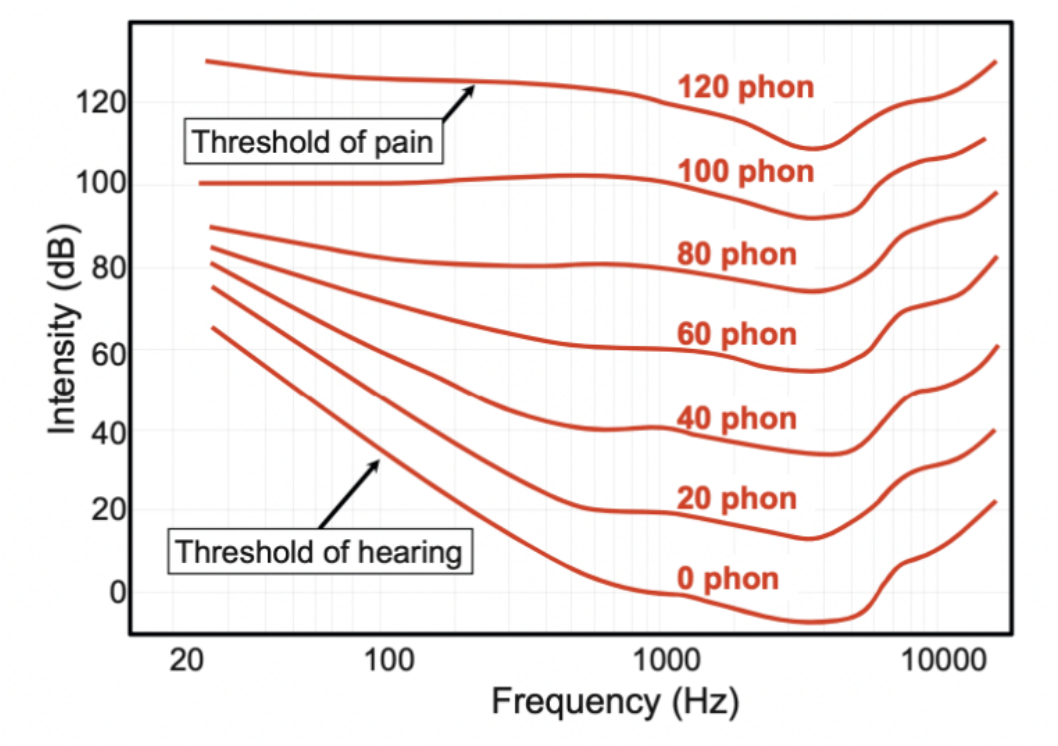

Figure 2.2: Equal Loudness Contours [1]

The *timbre* of a sound is the quality of the sound or the tone "color". Timbre is what allows humans to distinguish between different sources or different instruments playing the same musical pitch. Timbre can be difficult to derive because of its vague definition, but the partial frequencies and envelope of a musical note can be used to quantify the timbre of a sound. Figure 2.3 shows the envelope (in red) for a single musical note. The labels for attack (A), sustain (S), decay (D), and release (R) are also labeled on this diagram. These quantities measured from an envelope can be used to distinguish the timbre of a sound.

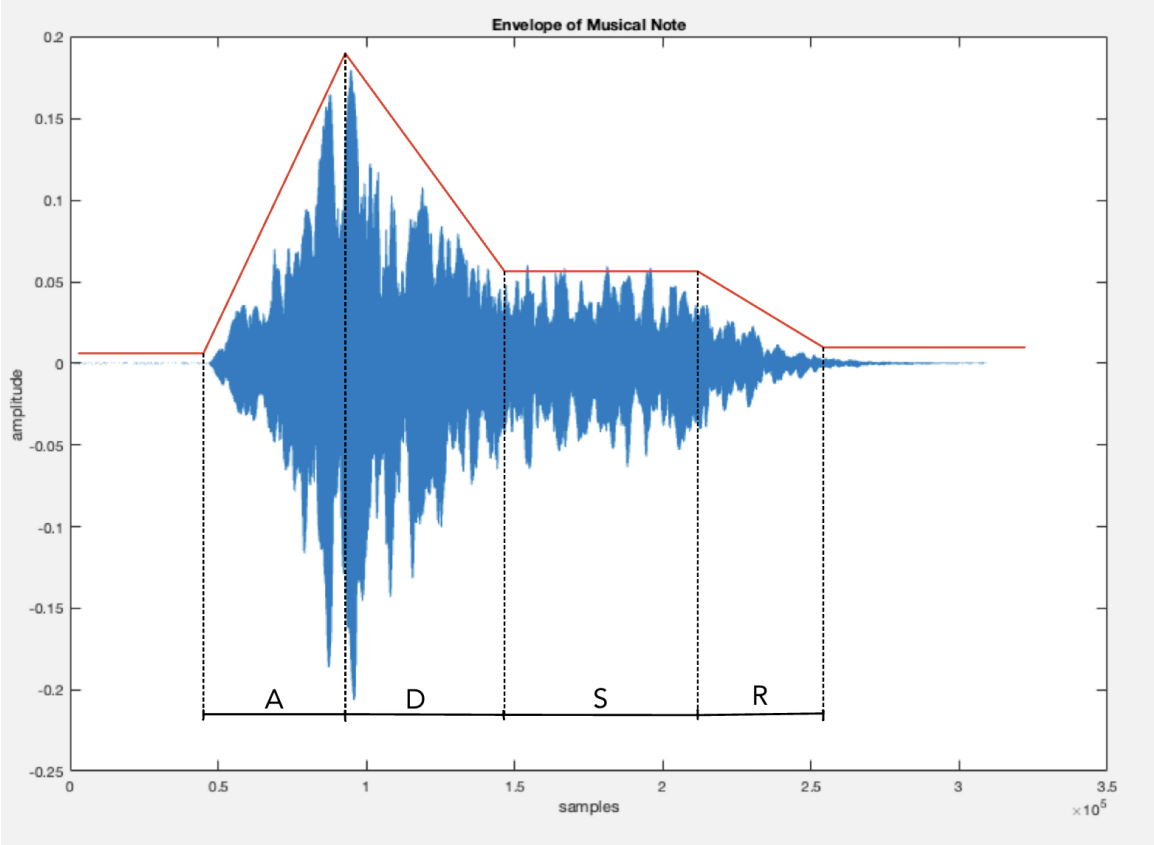

Figure 2.3: Envelope of a Sustained Musical Note

Audio representations can be shown as waveforms (pressure-time plots) in the time domain or can be shown in the frequency domain as time-frequency plots, also known as spectrograms. Figure 2.1 shows how air pressure deviations can be shown graphically as waves when related to time. At the destination, which could be a microphone or a listener, the wave is received and transformed into sound. The deviations in air pressure are created through compression and rarefaction which creates the alternating pressure that is recognized as waves. Spectrograms are a time-frequency representation commonly used in music analysis. The spectrogram is found by taking the short time Fourier transform (STFT) over multiple periods of the signal and then representing the frequency, time, and phase information in a plot. The vertical axis represents time and the horizontal axis represents frequency and is often plotted using a logarithmic representation [1]. The spectrogram for the musical note shown

in Figure 2.3 is shown in Figure 2.4. The spectrogram shows that most of the power in the signal is contained at the center samples and remains at lower frequencies. In Figure 2.4, the equally-spaced vertical bars represent the harmonic frequencies of the partials making up the sound; also giving rise to its particular timbre.

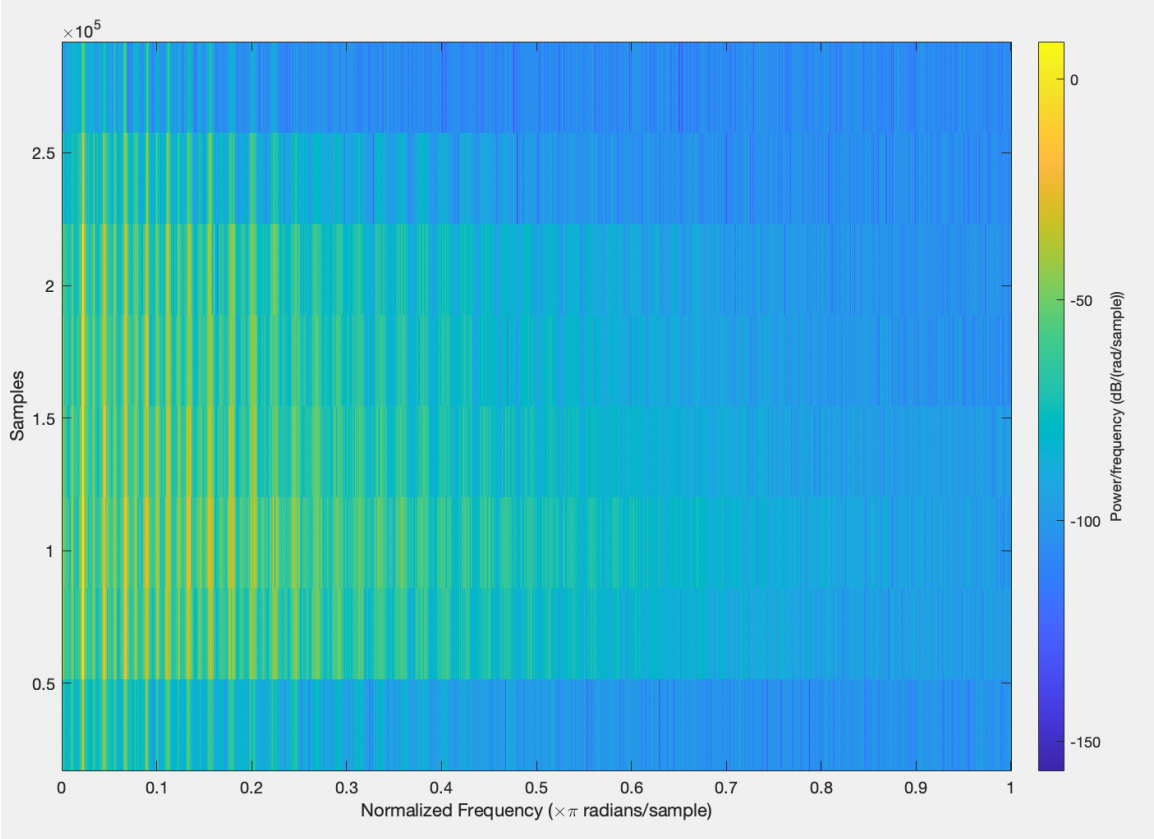

Figure 2.4: Spectrogram (time-frequency plot) of Waveform in Figure 2.3

2.3 Music Source Separation

Music source separation (MSS) is the technique for separating a piece of recorded music, which has high levels of correlation between sources, into the separate sources. Most current methods separate the music into drums, bass, vocals, and other sources. These four categories are common because most music source separation datasets use stems which are divided into these categories. The MUSDB18 dataset, for example,

has 150 songs which are provided in encoded MPEG4 files and can be separated into these four stems [13]. Approaches to MSS generally choose one type of audio representation for separation, although some state-of-the-art methods are now using both waveform and spectrogram representations of audio. The next few sections will discuss cutting edge approaches to music source separation for spectrogram domain, waveform domain, and hybrid methods. In more recent years, the highest performing MSS methods use deep learning to create a neural network which can accurately predict and separate the sources. For the four source categories, there will be four neural networks trained, one for each source.

2.3.1 Spectrogram Domain Methods

Spectrogram (time-frequency) domain methods for music source separation use the STFT to convert waveforms into the spectrogram domain for the entire mixture and try to predict the output spectrograms for each of the sources. The predictions for the output spectrograms are derived from features on the input spectrograms and masks are created to produce the output spectrograms. Spectrograms have a few characteristics that make them a good candidate for the basis of source separation. First, time-frequency representations contain multiple time-frequency bins and have a property called sparsity. Sparsity dicatates that only a small number of time-frequency bins have significant amplitude which leads to the phenomenon of disjoint orthogonality. Disjoint orthogonality defines a small probability that two independent sources have significant amplitude in the same bin. Therefore spectrograms can be very useful for source separation because each source will likely be contained within its own set of time-frequency bins. Second, the structure of spectrograms can help in source separation by looking at repetitions and harmonicity [14]. Prediction of the output spectrograms has been approached using Gaussian Modeling [15] [16], non-

negative factorization [17] [18] [19] , kernel additive modeling [20], and combinations. In recent years, deep learning and neural network approaches have shown significant improvements over these earlier approaches. The following sections outline some of the state-of-the-art deep learning approaches and highlight key structural features.

2.3.1.1 MMDenseLSTM

MMDenseLSTM is a neural network model for music source separation that retains a small model size while achieving high accuracy. Deep neural networks have been shown to improve source separation but with large numbers of time frequency (TF) bins in spectrograms, these models expand quickly and take longer to train. This particular model operates on spectrograms and uses a 4096 window size for the STFT. The model blends two existing systems: MMDenseNet and Long Short-Term Memory (LSTM); and Takahashi et al [3] demonstrates that the combination outperforms both of the individual systems.

The MMDenseNet network is a multi-scale, multi-band structure where each frequency band is linked to a single convolutional neural network (CNN) in addition to a full-band CNN. This takes advantage of the fact that spectrograms have different local structures depending on the band of frequencies considered. The basis for each single CNN is the DenseNet architecture which performs well for image classification [2]. The multiple feature re-weight (MFR) DenseNet is shown in Figure 2.5.

This network is expanded to multiple scales in order to perform well for source separation by applying a dense block at multiple scales which is done by downsampling the output progressively and then upsampling until the original resolution is recovered. MDenseNet is then expanded to multiple frequency bands to account for the fact that spectrogram local structures correspond to the frequency band. This means

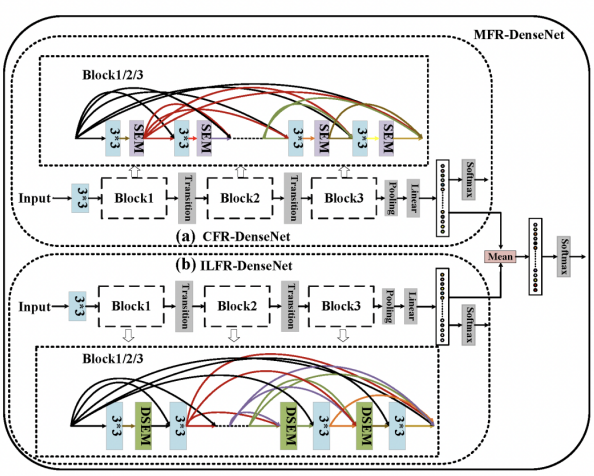

Figure 2.5: DenseNet Architecture [2]

the input is split into frequency bands and an MDenseNet structure is applied to each band.

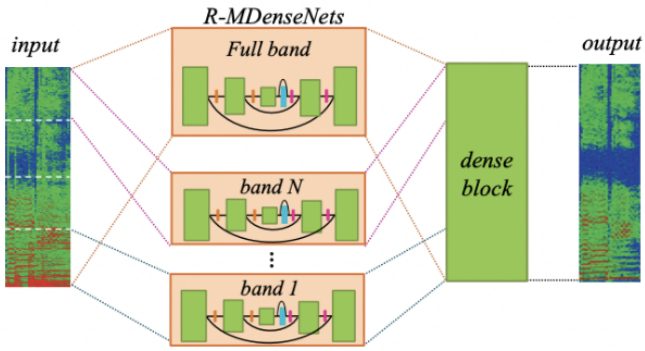

Figure 2.6: MMDenseLSTM Architecture [3]

The LSTM block is inserted in the upsampling path directly after the block with the smallest dimensionality. LSTM networks are a type of recurrent neural network that works well for temporal modeling, but has limitations related to underlying factors of variation within the input [21]. This placement of the LSTM is beneficial because the MMDense blocks are effective at modeling the local structures and the LSTM is effective at modeling the global structure. By placing the LSTM block at the point where the dimensionality is the lowest, the structure maintains a small model size while still having the fully-connected benefits provided by LSTM blocks. Figure 2.6

shows the structure for the MMDenseLSTM neural network and an example of an input and output spectrogram [3].

2.3.1.2 Parallel Stacked Hourglass Network

Another state-of-the-art neural network architecture for music source separation in the spectrogram domain is the parallel stacked hourglass network (PSHN). This specific architecture was constructed with traditional music in mind, which usually has fewer instruments. This model is a fully convolutional multi-scale network which generates time-frequency masks based on features learned from the input spectrogram. The overall architecture can be broken down into three parallel hourglass networks which are separated by frequency band. The networks are shown in Figure 2.7 and are separated into the upper band stacked hourglass network (UBSHN), full band stacked hourglass network (FBSHN), and lower band stacked hourglass network (LBSHN). The stacked hourglass networks are made up of hourglass modules which use a top-down bottom-up approach to extract multi-scale features. The modules have five downsampling operations followed by five upsampling operations which use nearest-neighbor interpolation. The modules are stacked and they create a network which also includes intermediate predictions. Intermediate predictions estimate masks between hourglass modules and therefore a separate loss function is required for each prediction. This means that each hourglass network requires four loss functions to find the final mask prediction. When tested on two popular datasets for traditional music, DSD100 and MIR-1K, the PSHN architecture performs comparably with state-of-the-art methods. This network especially performed well in singing voice separation because of its tailoring towards a low number of sources in the separation problem [4].

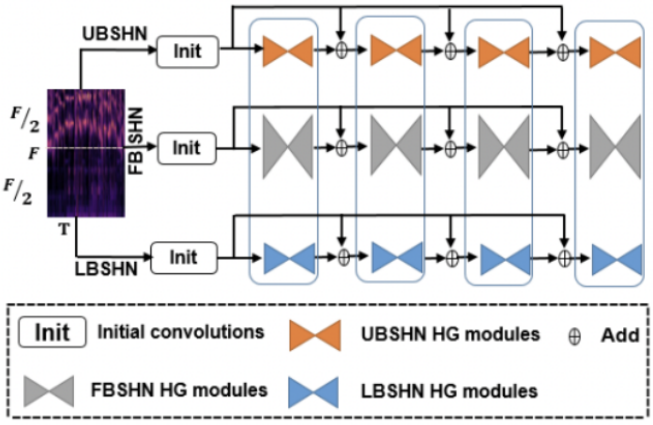

Figure 2.7: Parallel Stacked Hourglass Network Architecture [4]

2.3.1.3 Spec2Spec

The Spec2Spec neural network structure was inspired by methods that successfully used generative adversarial networks (GANs) for speech enhancement and singing voice separation. It is based on the idea that GANs are useful for domain translations. The fundamental concept behind a GAN is that it is composed of a generator which finds an estimated output and a discriminator which takes the estimated output and the ground truth data to adjust the weights. [5] considers MSS as a domain translation because it is a translation between two spectrogram domains, the total spectrogram and the separated spectrograms.

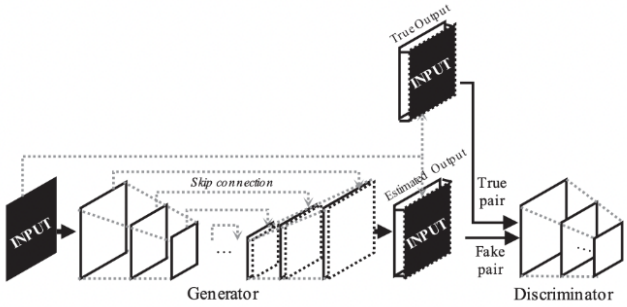

Figure 2.8: Spec2Spec Architecture [5]

In this structure, the generator is made of an encoder and decoder which are both composed of convolutional layers. There is a skip connection between the encoder and decoder stages which allows different levels of feature extraction during encoding and the reuse of these features in decoding. Each layer in the generator uses Leaky ReLU as the activation function and has a set of batch normalization. The discriminator step has convolutional layers with Leaky ReLU also, but does not have an activation on the output. Since there is no nonlinear function on the output, the output is a matrix. This network was tested on the DSD100 dataset, which is a dataset of 100 songs split into 50 for training and 50 for testing, and it performed similarly to state-of-the-art methods in terms of signal-to-distortion ratio and other common metrics [5]. The architecture for Spec2Spec is shown in Figure 2.8.

2.3.1.4 Open-Unmix

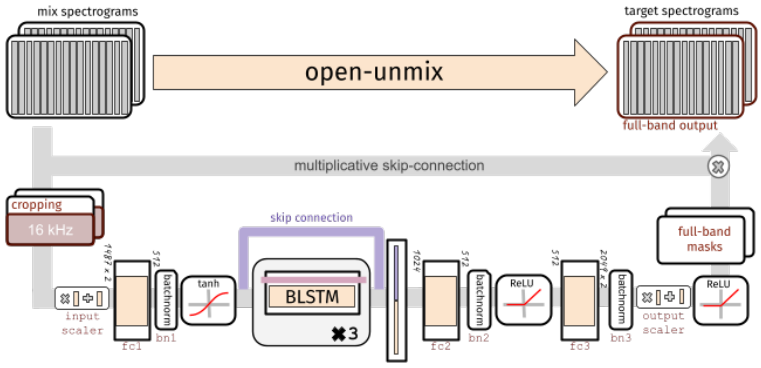

Figure 2.9: Open-Unmix Architecture [6]

Figure 2.9 shows Open-Unmix (UMX) which is an open source project for MSS that aims to make state-of-the-art MSS techniques available to all researchers. This project is an open source tool that is simple to extend, not packaged, and hackable. UMX is also reproducible through shared code, use of pre-trained models, and testing. The structure of the network is a bidirectional LSTM and when tested on the MUSDB18 dataset, the network performs at state-of-the-art levels [6]. UMX offers two different

preprocessing pipelines for the neural network. The first pipeline computes the STFT on the fly which takes more processing for each set of training iterations (epoch). The second pipeline option is to precompute the magnitude spectrograms and input them directly into the neural network. This second option is ideal because the overall training of the architecture will take less time.

2.3.2 Waveform Domain Methods

Time-frequency domain methods for source separation have been used more widely than waveform domain methods. This is because there are multiple advantages to using spectrograms for source separation. Spectrograms have properties such as the sparsity and structure that can make the challenge of source separation much easier, but at the cost of computation [14]. Converting to the spectrogram domain requires taking the STFT before separation and then performing the inverse STFT (ISTFT) after separation. Since this takes up computational power, finding a way to perform source separation directly on the waveform is a particularly interesting problem for researchers. In addition to saving on computational power, operating directly on the waveform avoids any loss of information in reconstructing the waveforms from the spectrograms of separated sources. Up until recently, waveform domain methods have not performed comparably with spectrogram domain methods, but more recent developments in using deep learning for separation in the waveform domain, have made systems that are comparable to state-of-the-art MSS methods. Waveform domain methods work by exchanging the STFT step that is found in spectrogram domain methods, with a feature extraction step. After the feature extraction, the waveform is sent into the neural network and processing is done directly on the waveform [7].

2.3.2.1 Conv-TasNet

Conv-TasNet (convolutional time-domain audio separation network) is one of the most widely recognized state-of-the-art speech separation systems and one of the first waveform domain systems to outperform spectrogram domain methods. Conv-TasNet is based on LSTM-TasNet which uses a TasNet and LSTM network in series. The structure of a TasNet network is a convolutional encoder-decoder architecture where the encoder has a non-negativity constraint on the output and the decoder is a linear decoder that inverts the encoder output to obtain a sound waveform. The overall architecture of this system uses three stages: encoder, separation, and decoder. Conv-TasNet takes a different approach than an LSTM-TasNet because it uses 1-D convolutional blocks for the separation step instead of an LSTM network. Using 1-D convolutional blocks allows for parallel processing and therefore better speed and a smaller model size. Conv-TasNet was tested and trained on a speech separation dataset and when compared with spectrogram domain methods, it often outperformed. Although this system is not a MSS specific system, it is the basis for the state-of-the-art waveform domain methods [7].

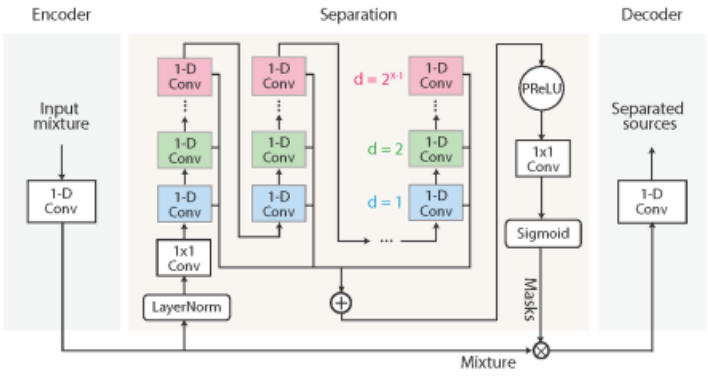

Figure 2.10: Conv-TasNet Architecture [7]

2.3.2.2 Demucs

The Demucs architecture for source separation in the waveform domain is one of the first MSS approaches in the waveform domain that performs similarly to or outperforms spectrogram domain methods. This architecture is inspired by music synthesis rather than masking methods that are employed on spectrogram domain methods. The architecture, which is shown in Figure 2.11, is based on Conv-TasNet. Demucs was developed by adapting Conv-TasNet for stereophonic MSS. This was done by feeding the music sources into the network in eight second clips because Conv-TasNet was designed for only small segments of audio. The network was also expanded by

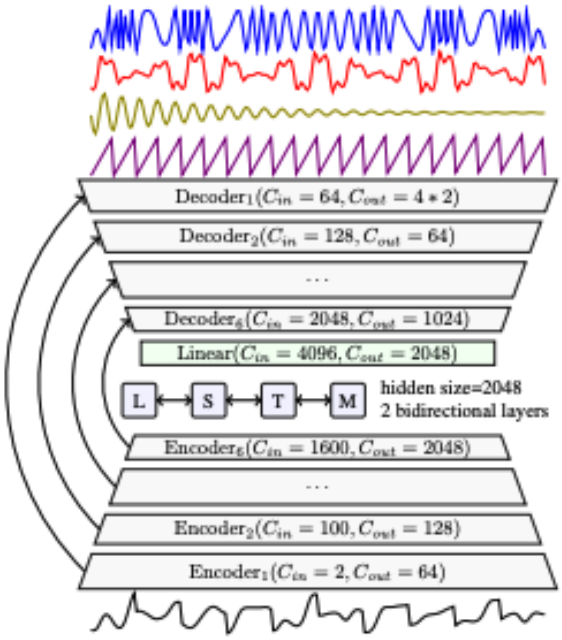

Figure 2.11: Demucs Architecture [8]

increasing the kernel size of the convolutional encoder-decoder and increasing the number of channels. Demucs also uses an LSTM network for the separation step and includes skip connections. The encoder is made of L stacked convolutional blocks which contain an input with ReLU activation on the output followed by a convolution with a gated linear unit activation. The decoder works to inverse the encoder

to bring back the sound waveforms. In reconstructing the individual waveforms, the skip connection is very helpful because it provides phase information from the original signal. The skip connections are connections from the encoder to decoder steps with the same indices. Demucs was evaluated on the MUSDB18 dataset and it was able to perform with state-of-the-art results. Demucs outperformed spectrogram domain methods in the bass and drums categories [8].

2.3.3 Hybrid Domain Methods

In addition to spectrogram domain and waveform domain methods, another major category in MSS is hybrid approaches. Hybrid methods make use of both spectrograms and the original waveforms and are often able to outperform methods in the individual domains.

2.3.3.1 Bridging Networks

Bridging networks address two major issues that appear in many MSS approaches. The first being that most methods only consider the time domain or frequency domain versions of the original waveform, but not both. Using both gives more information to the neural network and therefore results in better source estimations. The second issue is that many methods do not consider mutual influence in the output sources because the loss functions are applied independently to each source.

In [9] Sawata et al. propose two new loss functions, multi-domain loss and combination loss, in order to address common issues. The multi-domain loss appends either a STFT or ISTFT layer after the neural network during training and then calculates the loss before and after that layer. This loss is shown in Figure 2.12 below. The extra layer is only appended during training. The loss is calculated by taking the mean

squared error (MSE) loss between the ground truth spectrogram and the estimated spectrogram in the frequency domain and adding it with a scaled signal to distortion ratio in the time domain. The scaling factor used for this approach was $\alpha = 10$.

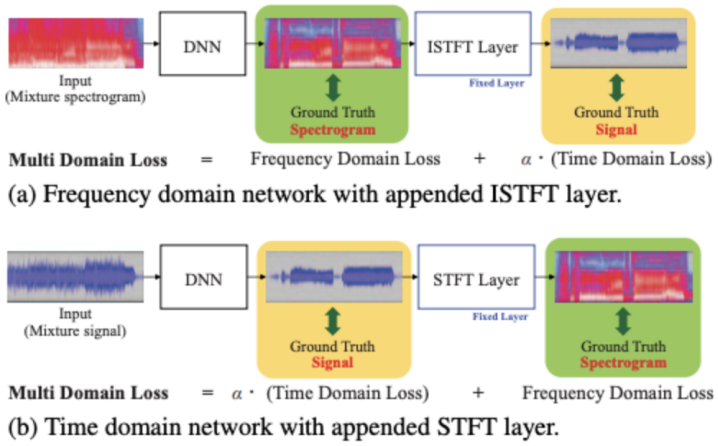

Figure 2.12: Example of appended layers [9]

Combination loss considers the relationship among the output sources by producing output spectrograms for combinations of the output sources. This means two or more estimated spectrograms can be combined to try and find correlated errors from leakage. If two sources contain leakage from the same source, the common loss will be easier to identify by combining the output masks.

These new loss functions were tested on the Open-Unmix architecture during training. Since the loss functions proposed only need to be added during training, they can be useful for any MSS deep learning architecture. In addition to adding the loss function, this approach also proposed adding bridging networks to the Open-Unmix architecture. This is done by connecting the paths to cross each source's network and applying an averaging operator. With these additions, the proposed network is called CrossNet Open-Unmix. When tested on the MusDB18 dataset, the CrossNet outperformed the results from the initial Open-Unmix network [9].

2.3.3.2 Hybrid Demucs

Hybrid Demucs is an expansion upon the original Demucs architecture which performs separation using the frequency and time domain [10]. This architecture consists of a temporal branch, spectral branch, and shared layers. The temporal branch performs waveform processing similar to the Demucs architecture. The spectral branch uses the STFT to convert to a spectrogram and then uses the same number of convolutional layers as the temporal branch to downsample the spectrogram into one frequency bin. The spectral layer does this through frequency-wise convolution and division of the number of frequency bins with every layer. After both branches have been downsampled in their encoder steps, they are summed and passed through an encoder and decoder layer which produces outputs for both branches. Finally, both outputs

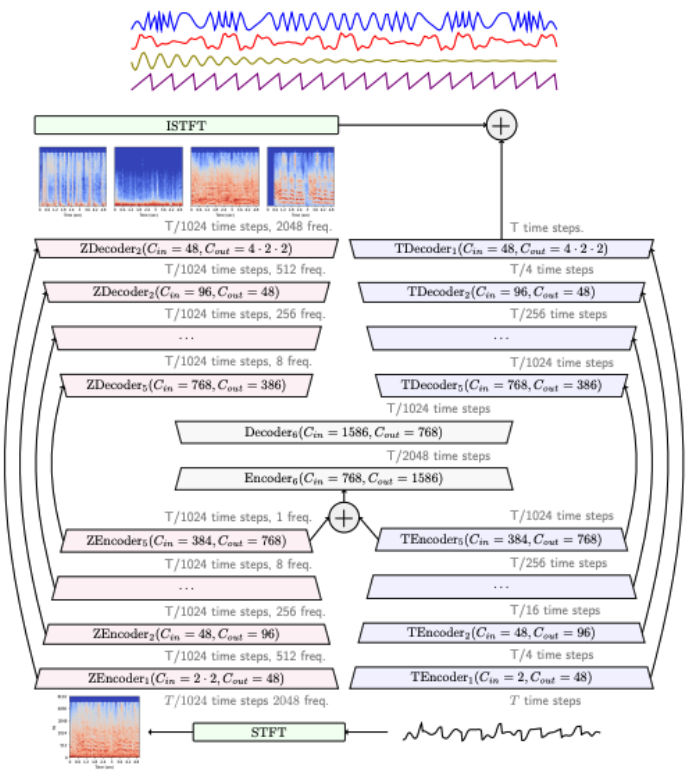

Figure 2.13: Hybrid Demucs Architecture [10]

are passed through their respective decoder layers and the ISTFT is performed on

the spectral branch to obtain the final source waveforms. The architecture for Hybrid Demucs is shown in Figure 2.13.

This network was tested against other networks and it outperformed other networks in the drums and bass sources, while still performing at state-of-the-art levels for vocals and the "other" category [10].

2.4 Source Panning Parameter Estimation

An important step in creating the visualization from the separated sources is estimating the panning parameter of the source to find its relative horizontal location. In many MSS applications that do not utilize deep learning, source panning parameter estimation can be helpful for approximating and separating the sources [22] [23]. Source panning parameter estimation can also be helpful post-separation for visualization purposes. Using source panning parameter estimation for visualizations can help to recreate the phantom image that listeners hear in stereo music recordings.

Figure 2.14 shows where the placement of the phantom image will fall if the left and right speakers are at equal volume levels and the listener is in the middle of the horizontal plane. If the listener moves from the center of the horizontal plane, the phantom image will also move in that direction. When estimating the source panning parameter, the listener is assumed to be in the center of the horizontal plane.

In many DAW applications, the panning is applied using a panning potentiometer which can adjust the panning for the source from left to right. The potentiometer has panning locations from 0 to 180 degrees which respectively aligns with hard left panning to hard right panning. When a panning potentiometer is used, the sine/cosine law is applied to the right and left channels in order to shift the phantom image of the

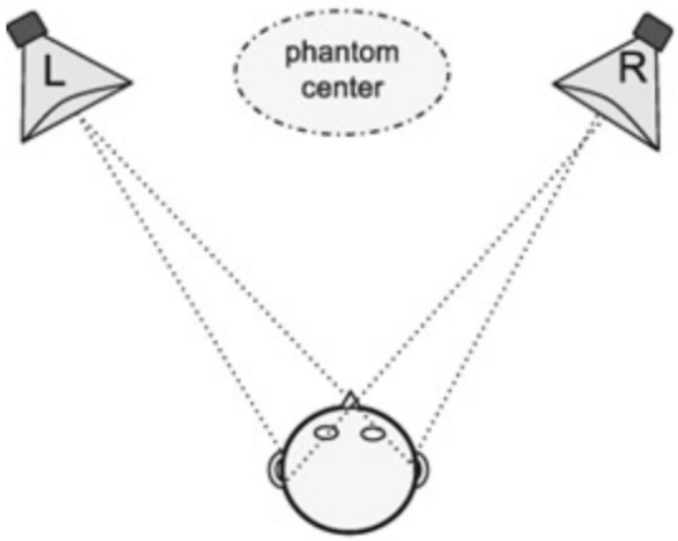

Figure 2.14: Phantom Image Illustration [11]

music. The equations for finding the right and left channel signals in the creation of stereo audio are shown in Equation 2.1 and Equation 2.2 respectively where w stands for the panning potentiometer location.

$$\text{Left Signal} = cos(w) * \text{Input Signal} \tag{2.1}$$

$$\text{Right Signal} = sin(w) * \text{Input Signal} \tag{2.2}$$

Rearranging these equations, the panning potentiometer location can be found with Equation 2.3. This equation assumes constant acoustic power across the two channels [22] [11].

$$\theta = arctan\left(\frac{\text{Right Signal}}{\text{Left Signal}}\right) \tag{2.3}$$

2.4.1 Constant Power Panning Law

In order to derive Equation 2.3, the constant power law is assumed. The constant power law says that in audio, the total power of the channels does not change as the panning parameter is varied. The constant power panning law assures that the perceived volume of the listener stays consistent regardless of panning [24].

2.5 Pitch Detection

Pitch detection is a large area of study in the field of MIR. The pitch or fundamental frequency is the rate at which an instrument or vocal chords vibrates. The pitch of an instrument or singing voice is a quality which reflects the highness or lowness of that sound's frequency; and utilizes a critical feature for creating meaningful visualizations to augment the understanding and experience of music.

2.5.1 Pitch Ranges

Typically, the fundamental frequency of male voices will range from 55 Hz to 131 Hz and female voices will range from 170 Hz to 262 Hz. Instruments have a much larger fundamental frequency range which can go from 16 Hz all the way to 4 kHz. Overall, most fundamental frequencies will fall in the range of 50 Hz to 1000 Hz so these are the frequencies that are typically focused on for musical pitch detection algorithms [25].

2.5.2 Pitch Detection Algorithms

Pitch detection is widely considered a thoroughly investigated problem in the field of MIR so there are many different solutions for pitch detection. This thesis will cover some of the most popular pitch detection algorithms and highlight their key differences.

2.5.2.1 Autocorrelation Method

The autocorrelation method for pitch detection is a very popular and accurate algorithm. The autocorrelation method has been studied for years and is included in most comparative studies such as in [26] [27] [28]. This method cross correlates the signal with itself for a frame of samples and examines the resulting peaks. Autocorrelation is a two-sided function so the result from the cross correlation will range from -W to W, with W being the window length. The global maximum will always be in the center at n = 0 (zero lag between the signal and itself) and additional local maxima will appear at multiples of the signal's period. The distance to the first maxima from the zero-lag center represents the fundamental period of the signal and the inverse of this value is the fundamental frequency that defines the pitch. Picking the correct peak from the autocorrelation results can often be difficult, so many methods use a threshold to eliminate extra peaks from noise [29]. The function for computing autocorrelation is shown in Equation 2.4. W is the window length and the autocorrelation is computed by multiplying the signal by a shifted version of itself. The results are then summed to get the autocorrelation value at that index.

$$r[n] = \sum_{n=0}^{W} x[n]x[n+W] \qquad W = 0, 1, 2, ..., W \tag{2.4}$$

2.5.2.2 Square Difference Function

The square difference function (SDF) algorithm uses the characteristic of music signals that if a signal is pseudo-periodic, any two adjacent periods of the waveform will be similar in shape so shifting the signal by one period should result in the peaks lining up. After the peaks are in line the difference between the peaks can be taken to find the fundamental periods. However, since music signals can have negative values, taking the square of the difference will ensure only non-negative values are added to the sum. For each window that the function is applied to, the first local minima is found and used as the fundamental period [28]. The equation for SDF is shown in Equation 2.5. Each version of the signal is subtracted by a circularly shifted version of itself and squared. Then each window is summed to find the SDF value.

$$r[n] = \sum_{n=0}^{W} (x[n] - x[n-W])^2 \qquad W = 0, 1, 2, ..., W \tag{2.5}$$

2.5.2.3 Average Magnitude Difference Function

The average magnitude difference function (AMDF) [30] follows the same idea that the square difference function does except instead of taking the square of the difference, this function takes the absolute value of the difference. This results in similar pitch values but has a less harsh penalty for when the peaks do not line up perfectly. This function also uses the local minima to find the fundamental period of the signal [27]. Equation 2.6 shows the AMDF. Each time the window is shifted the signal and the shifted version are subtracted and the absolute value is taken. The AMDF value for

each index is the average of the values from the subtraction.

$$r[n] = \frac{1}{W}\sum_{n=0}^{W}|x[n] - x[n-W]| \qquad W = 0, 1, 2, ..., W \tag{2.6}$$

2.5.2.4 Cepstrum Method

The Cepstrum method for pitch detection [31] is a frequency domain approach to pitch detection which is focused on detection for voiced speech. We can assume that most voiced speech sounds are convolved with a vocal tract filter so this algorithm is often used in speech processing applications [29]. This method tends to be very accurate but is more computationally expensive because it requires the Fourier transform to be taken. This algorithm works by moving the signal to the Fourier domain by taking the Fast Fourier Transform (FFT). The FFT is then converted to a logarithmic scale and converted back to the time domain using the inverse Fast Fourier Transform (iFFT). This Cepstrum in the time domain will have peaks corresponding to the fundamental period of the signal and these peaks are used to find the fundamental frequency [29]. Equations 2.7, 2.8, and 2.9 show the calculation for the Cepstrum. H is the vocal tract filter and S is the original signal, both in the frequency domain. These are multiplied together and then the log is applied to find the Cepstrum. This will produce multiple small peaks spaced along the frequency axis and they will occur at multiples of the fundamental frequency.

$$X(f) = S(f)H(f) \tag{2.7}$$

$$log(X(f)) = log(S(f)H(f)) \tag{2.8}$$

$$log(X(f)) = log(S(f)) + log(H(f)) \tag{2.9}$$

2.6 Loudness

Loudness is a perceptual concept which is influenced by frequency. Listeners tend to hear higher frequencies as louder and lower frequencies as quieter when at equal loudness [24]. This is demonstrated by the equal loudness contours shown in Figure 2.2. In applications which read in the audio, the amplitude is normalized in the range of -1 to 1. The amplitude can be used as a reference for how loud the source will sound to the listener. An example of the right and left sides of an audio sample are shown in Figure 2.15.

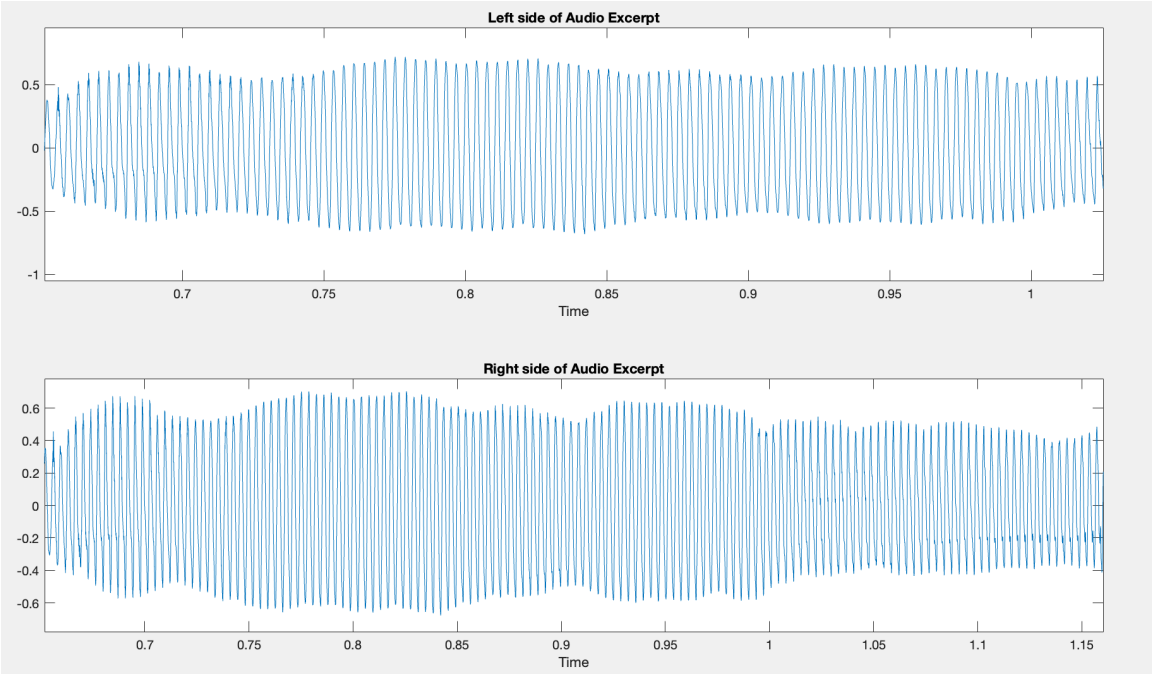

Figure 2.15: Example of Two Waveforms of a Stero Audio Sample

2.7 Music Visualization

Music visualization uses computer graphics to represent a musical performance. Many approaches for music visualization use the original score or performance information to

create a visualization [32]. Visualization allows users to understand patterns within music and extract further understanding of a musical piece. Music visualization techniques can be broken up by the type of input data that is used for the system. Techniques will either take the MIDI information, audio signal, or other inputs [12]. Since this thesis is concerned with recorded audio, the input that this research is focused on is stereo audio signal input. The MIDI input requires information from the digital audio workstation which is not always available to users. In addition to the type of input, music visualizations also choose some features to focus on for the visualization. These features can include but are not limited to: pitch, timbre, structure, harmony, panning, melody, and mood. These features can be translated into visualizations through different techniques that use color, shapes, line graphs, and other visual aids to deepen the understanding of the music [12]. Pitch detection, panning parameter estimation, and loudness were outlined in sections above and can be used for the basis of a visualization. The section below will explore the relationships between music with color.

2.7.1 Music and Color

The relationship between color and music has been theorized about since early human history. In ancient Greece the colors of the rainbow were associated with different musical notes [12]. One of the most commonly referenced phenomenon that explores the relationship between music and color is music-color synesthesia. Music-color synesthesia is a condition where the listener hears a color along with hearing the sound [33]. However, people with or without synesthesia still tend to associate certain parts of music with visual or spatial experiences. One connection between music and color is linked with emotion. Emotion can be connected to both music and color and studies have shown that these close connections can cause associations between music

and color in non-synesthetes [33]. In general, higher pitches tend to be associated lighter colors while lower pitches tend to be associated with darker colors. In an experiment on music-color associations for nonsynesthetes, participants were asked to choose colors based on a musical piece and the results show a correlation between music, color, and emotion [34]. Some approaches to colored musical visualizations associate specific notes with different colors, while others use color to represent a combination of tones [12].

Chapter 3

INVESTIGATION

Many different types of music visualization systems have been proposed with different input types, processing of the signals, and visualization styles. This project aims to incorporate state-of-the-art music source separation techniques into a music visualization system in order to produce visualizations that can summarize how key musical elements combine into a holistic experience of a song. Most music visualization systems that decompose the different music sources of a song into parts have access to the digital audio workstation files. However, this project aims for an end-to-end visualization that allows users to work with any song and visualize it.

This chapter will describe the goals, constraints, and assumptions of the project. The chapter will also go into the investigation that was conducted into the different signals processing techniques that were considered for the project.

3.1 Goals

The goals of this project were defined and used to guide the creation of the music visualization system. The goals for this project are as follows:

1. The system must be end-to-end so that a user can input a song and a visualization will be produced.

2. The source separation must be accurate in order to inform the pitch detection properly.

3. The pitch detection algorithm must be robust and able to accurately predict the pitch even in the presence of noise.

4. The source localization technique must align with the correct panning parameters.

5. The visualization should help increase user understanding of music, particularly how pitch, panning, and loudness affect a song.

3.2 Constraints and Assumptions

A few constraints and assumptions were made to create the overall visualization system. The main constraint was maintaining a reasonable processing time for rendering a video for a full-length song. Since music source separation and video rendering can each require large amounts of time to complete, the total processing time should be kept at a minimum to make the system more useable. There were also a few assumptions made to inform the pitch detection and localization techniques. The localization technique assumes that the audio was panned between channels using a panning potentiometer, so that the horizontal source location is determined by amplitude difference between channels rather than arrival time differences. The pitch detection algorithm does not account for any frequencies above 1000 Hz, since note pitch fundamental frequencies for instruments and singing voices fall below this threshold.

3.3 Investigation of Music Source Separation Techniques

3.3.1 Signal-to-Distortion Ratio

Signal to distortion ratio (SDR) is the most commonly used metric to evaluate music source separation networks. The equation for signal distortion ratio is shown in

Equation 3.1. Here, s(n) denotes the original signal and $\hat{s}$(n) denotes the separated signal. Where s(n)-$\hat{s}$(n) is the error from all types of distortion including interference, noise, and artifacts. A small constant ϵ is also added to avoid zero divisions.

$$SDR = 10 \log_{10} \frac{\sum_n ||s(n)||^2 + \epsilon}{\sum_n ||s(n) - \hat{s}(n)||^2 + \epsilon} \tag{3.1}$$

The signal to distortion ratio can be calculated using test data for each source that the source separation network separates.

3.3.2 MUSDB18 Dataset

The MUSDB18 [13] dataset consists of 150 full length tracks along with their isolated sources. The isolated sources are separated into four groups: vocals, bass, drums, and other. All of the signals from this dataset are stereophonic mixtures and have a sampling rate of 44.1 kHz which is typical for audio signals. The MUSDB18 dataset is made of compressed stems which need to be decoded for use. There are multiple parsers provided on the MUSDB18 website to decode and iterate over the dataset.

3.3.3 Music Source Separation Preprocessing and Training

For spectrogram and hybrid domain MSS methods, the main preprocessing step for the neural network is computing the STFT spectrograms. For stereophonic MSS methods, this means computing the individual spectrograms for both channels of the audio. It is helpful to precompute all the spectrograms before beginning training in order to reduce the required computing power during training. Figure 3.1 and Figure 3.2 show the spectrograms for each channel of the same example song from the MUSDB18 dataset.

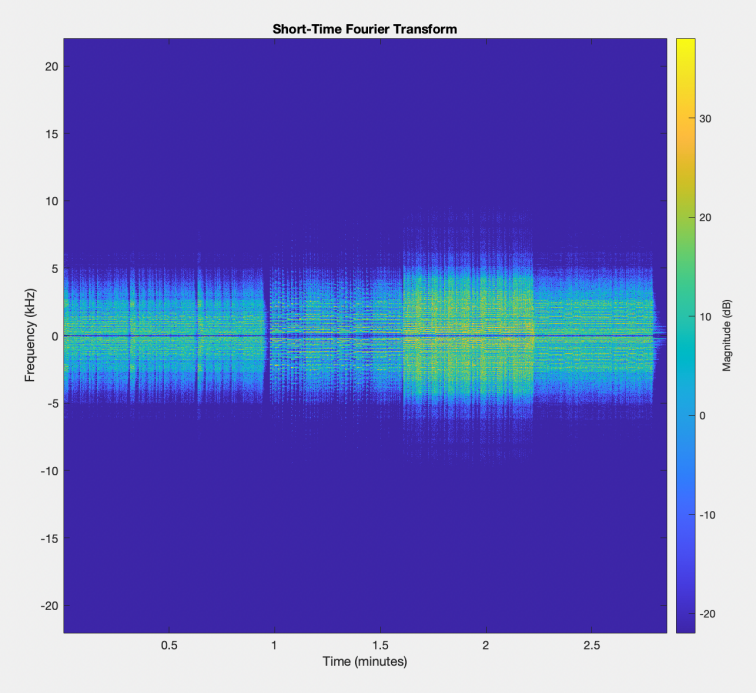

Figure 3.1: Channel 1 Spectrogram

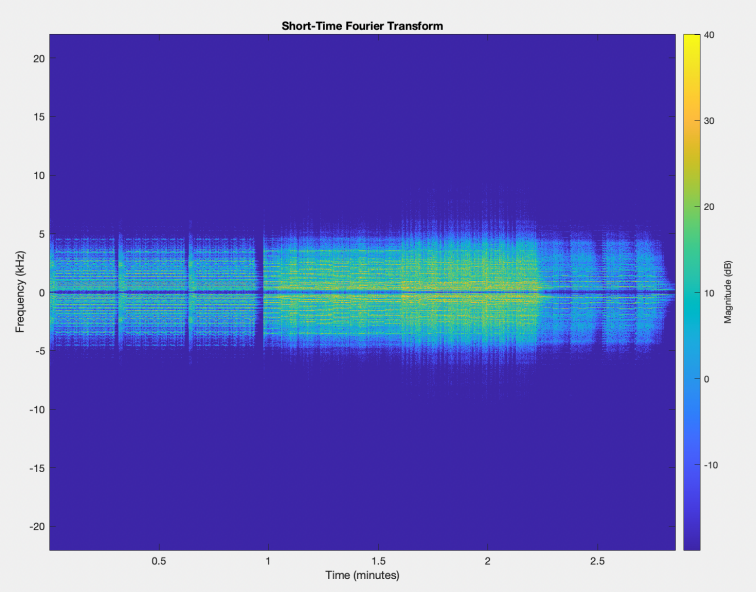

Figure 3.2: Channel 2 Spectrogram

For spectrogram domain MSS methods, the input into the network is typically both spectrograms of the two audio channels. The network produces a mask that can be applied to the magnitude spectrum to isolate each source's contribution to the

spectrum. Each mask is first initialized and the network is then trained to try and create a mask that will isolate the correct spectrogram for all examples from one source class. The spectrograms that are output from the network are applied to the spectrogram magnitudes, and converted back to the time domain using inverse STFTs computed from the masked magnitude spectrograms and the phase information from the original (non-separated) spectrogram.

3.3.4 Music Source Separation Comparison

The Signal Separation Evaluation Campaign (SiSEC) has been comparing state-of-the-art MSS systems since 2008 [35]. SiSEC regularly reports on the progress in the source separation community and last published its findings in 2018. The Music Demixing (MDX) Challenge 2021 [36] is the follow up to the SiSEC in the professionally produced music (MUS) separation task. SiSEC 2018 and the MDX challenge use the MUSDB18 dataset for comparison although they vary in their selection of metrics for evaluation. The SiSEC uses signal-to-distortion ratio (SDR), signal-to-artifact ratio (SAR), signal-to-interference ratio (SIR), and image-to-spatial distortion ratio (ISR) from the BSS Eval toolbox [37]. The MDX challenge only uses SDR for evaluation of the state-of-the-art methods. In the MDX challenge, the individual source's SDRs are averaged to find the overall song SDR. Since SDR includes all types of distortion, it is the most general metric for comparing MSS approaches.

Table 3.1 above summarizes the SDR values for the techniques discussed in the literature review section of this thesis. These values were obtained from reports in the academic papers describing each of these architectures. The SDR_{song} values were calculated by averaging the four sources values. When comparing different state-of-the-art architectures for MSS, the Hybrid Demucs approach shows the best SDR in

Table 3.1: Summary of Music Source Separation Technique Metrics

Method	Training Set	SDR_{song}	SDR_{bass}	SDR_{drums}	SDR_{other}	SDR_{vocals}
MMDLSTM	MUSDB18	5.42	5.19	6.62	4.93	4.94
PSHN	DSD100	3.78	2.35	4.52	2.55	5.70
Open-Unmix	MUSDB18	5.325	5.23	5.73	4.02	6.32
Demucs	MUSDB18	6.7575	7.01	6.86	5.19	7.97
H-Demucs	MUSDB18	**7.33**	**8.12**	**8.04**	**5.19**	**7.97**

every category. The SDR is an important consideration for this project because too much distortion can cause the pitch detection algorithm to produce erratic results.

In addition to examining SDR, some other important metrics to consider are model size and training time. Source separation models in the spectrogram domain require more preprocessing because the STFT needs to be performed on the inputs. Hybrid domain methods also have very large model sizes because they utilize multiple branches (both spectrogram and waveform processing).

Without a powerful Graphics Processing Unit (GPU) available, training times can be quite long, so using a pre-trained neural network is ideal. Additionally, models for MSS need to be trained for each source, which means training the model and saving the final weights four times. In [8], the training times for a few of the state-of-the-art MSS neural networks are compared. This study used a GPU for training and found that open-unmix takes about 0.2 seconds per training batch and demucs takes 1.4 seconds per training batch. In a comparative study on neural networks trained on standard CPUs and GPUs [38], it was found that GPUs consistently complete training faster than CPUs. In some cases from this study, the GPU performed training four to five times faster than the CPU.

3.4 Investigation of Pitch Detection Techniques

3.4.1 Averaging Channels

Since the input to this project is a stereophonic music file and the algorithms that were implemented require one channel of information, the channels of the stereophonic recordings are averaged at each sample. These single-channeled versions of the recordings are used for all the pitch detection in this thesis.

3.4.2 MDB-melody-synth Dataset

The MDB-melody-synth dataset [39] consists of 65 full length songs in which the melody track has be re-synthesized to contain perfect fundamental frequency (f0) annotation. The dataset consists of three folders containing the original mix, resynthesized mix, and the annotations for the resynthesized mix. This dataset is ideal for testing automatic pitch detection methods because the accuracy can be checked with the annotations.

3.4.3 Applying a Window Function

For each pitch detection method, the audio is split up into segments the size of the window defined by the user. Since these audio segments contain a large set of samples, the hop size between one segment and the next is less than the window size in order to have some overlap and avoid missing information. The most important part of this window is the center, so applying a windowing function can help eliminate some of the less important information from the window and reduce edge effects and spectral leakage. Figure 3.3 shows a Hamming window of length 2048.

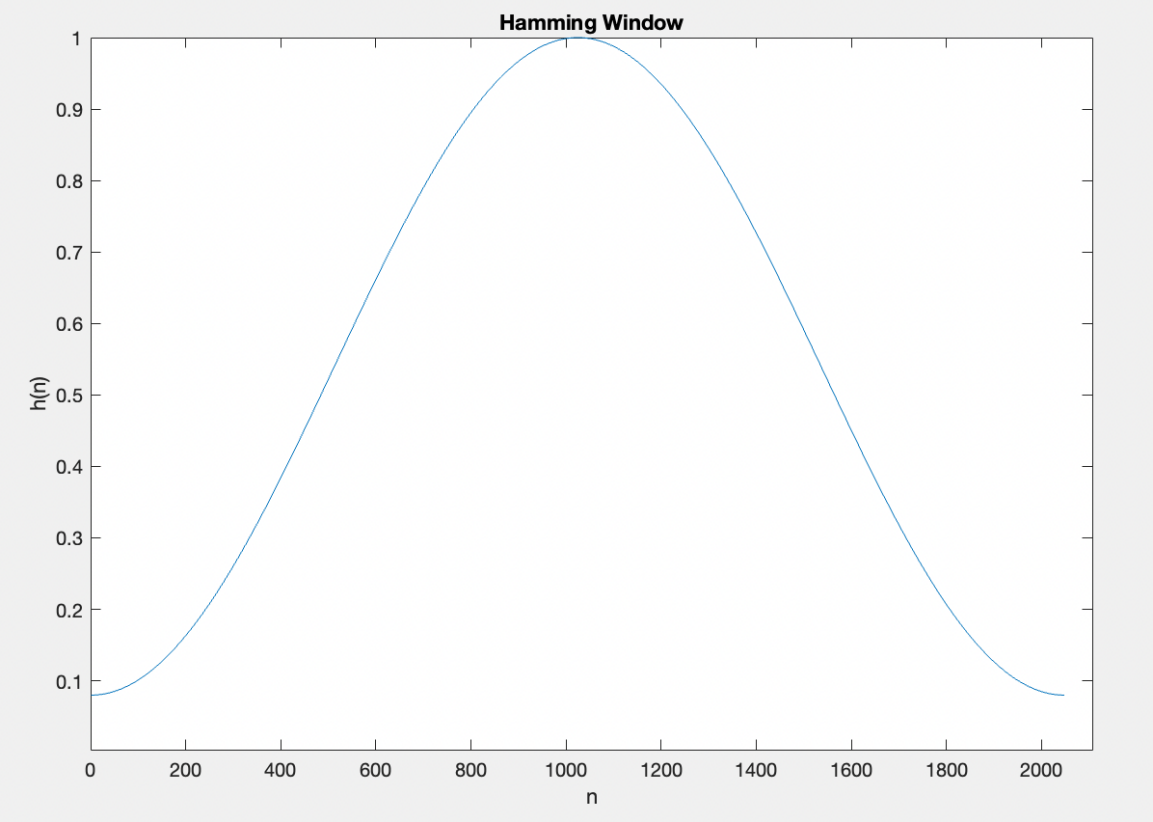

Figure 3.3: Hamming Window With W=2048

Applying the Hamming window function to the audio segment results in tapering of the audio. An example of the audio with and without the window function is shown in Figure 3.4. This demonstrates how the window function emphasizes the center part of the sampled audio. The original audio waveform is shown in orange and the windowed waveform is shown in blue.

3.4.4 Autocorrelation

The autocorrelation method for pitch detection was implemented in MATLAB. The general flow diagram for this algorithm is shown in Figure 3.5. First, the song is segmented into sections with the length equal to the chosen window size. Each segment has a window function applied to it and then the autocorrelation is performed. After

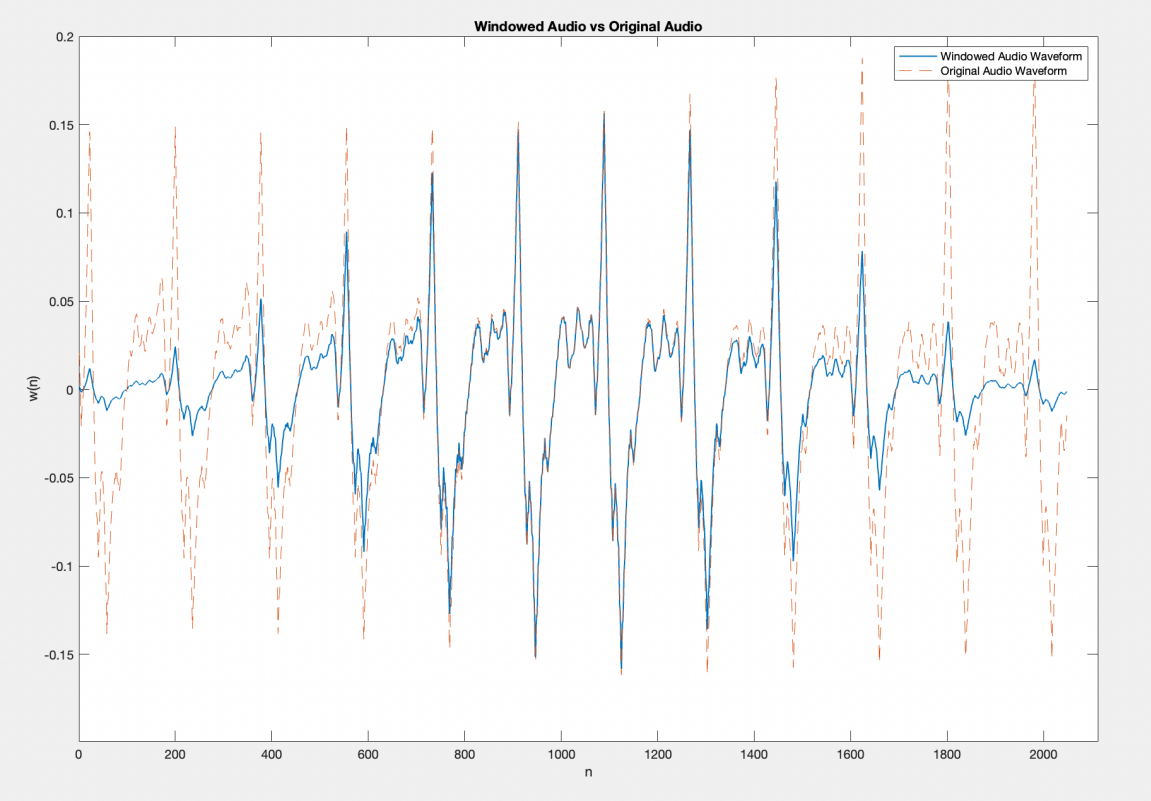

Figure 3.4: Audio With and Without Hamming Window Applied

the autocorrelation is performed, the index of the first peak is found and the fundamental frequency is calculated with this from the index of this first peak relative to the zero-lag position. The equations for finding the fundamental frequency are shown in Equation 3.2 and Equation 3.3.

$$\text{Fundamental Period } T = \frac{\text{Autocorrelation Peak Index (samples)}}{\text{Sampling Rate (samples/sec)}} \tag{3.2}$$

$$\text{Fundamental Frequency } f_0 = \frac{1}{T} \tag{3.3}$$

Figure 3.6 shows the result of the autocorrelation function for one window of the audio. The autocorrelation function is two-sided, so the first peak is picked from those occuring after zero lag and the center peak is ignored. In this case, the first peak is at index 178 so the fundamental frequency can be found as shown in Equation

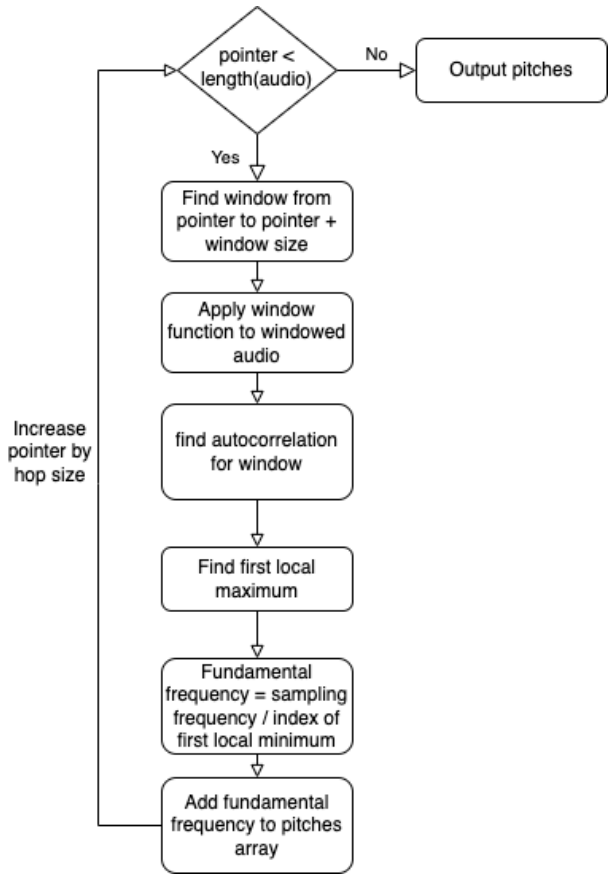

Figure 3.5: Autocorrelation Code Flowchart

3.4. The pointer is then increased by the hop size which is usually half of the window size in order to have 50 percent overlap. This process is continued until the end of the audio has been reached. It is important to note that the window size affects the minimum fundamental frequency that can be found. Since the window size determines the largest possible index in the autocorrelation, the smallest fundamental frequency possible is the sampling rate divided by the window size. If the window size is 2048, the lowest fundamental frequency that can be found is $\frac{44100}{2048} = 21.5332$ Hz. This is a sufficiently low frequency for music pitch detection because humans can only detect frequencies from about 20 Hz - 20 kHz [40].

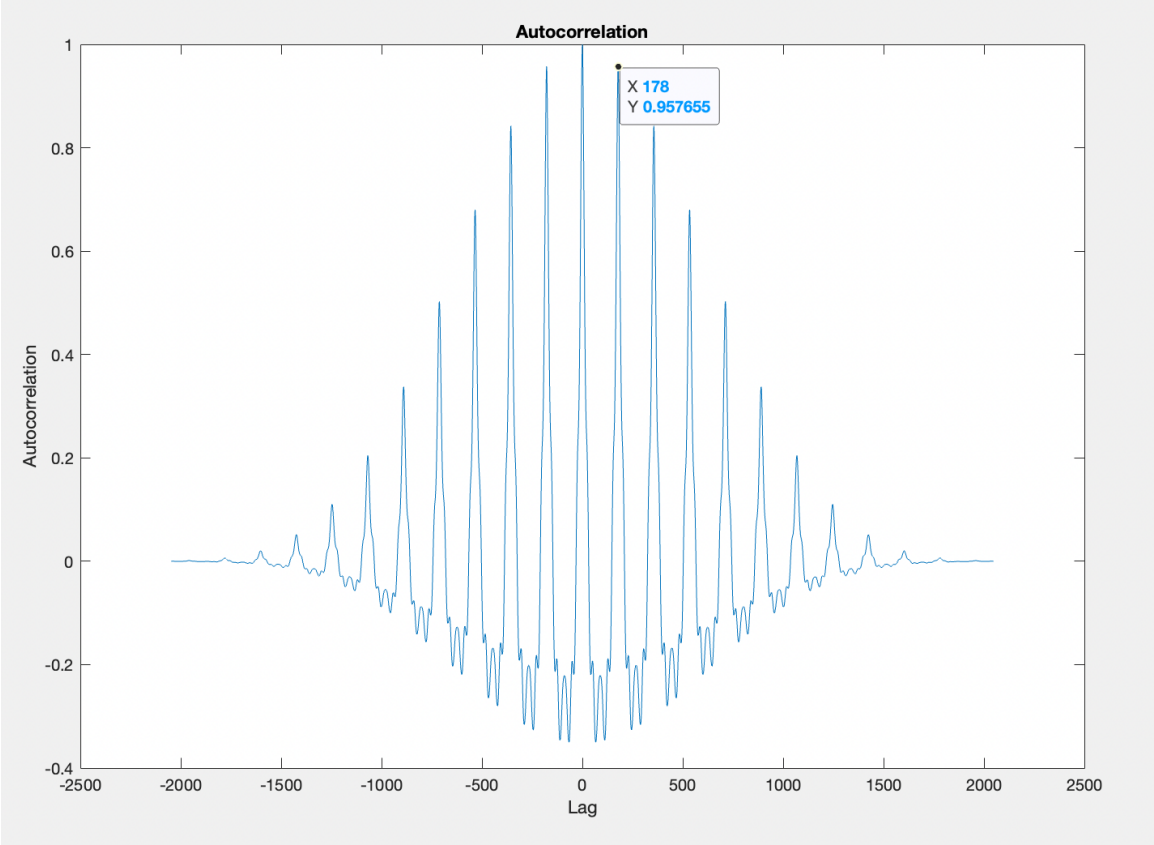

Figure 3.6: Autocorrelation of Audio Window

For this audio sample the sampling frequency is 44100 Hz, which is a typical sampling frequency for music.

$$\text{Fundamental Frequency} = \frac{\text{Sampling Frequency}}{\text{Index of First Peak}} = \frac{44100Hz}{178} = \mathbf{247.7528\ Hz} \tag{3.4}$$

3.4.5 Average Magnitude Difference Function

The average magnitude difference function (AMDF) was also implemented in MATLAB and tested on the same audio segment as the autocorrelation function. For the implementation, the difference is calculated between the original windowed segment

and the circular shifted version of that windowed segment. The flowchart for the AMDF is shown in Figure 3.7.

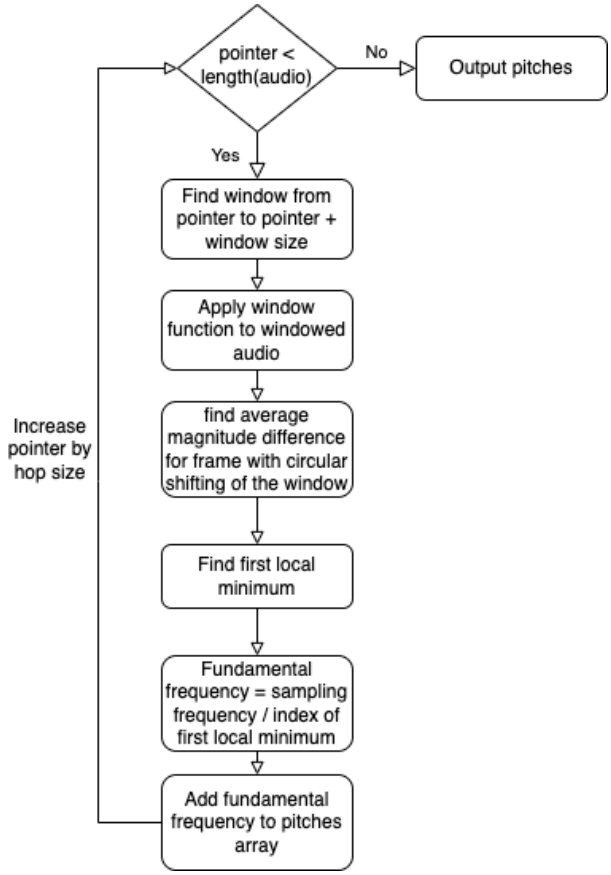

Figure 3.7: AMDF Code Flowchart

For the AMDF and SDF approaches the first local minimum is chosen for the index because these algorithms are checking for when the error between the periodic signals is the lowest. When tested on the same audio segment, the same index for the first local minimum was found for the autocorrelation, AMDF, and SDF approaches; and these matches the first peak index when the autocorrelation method was used. Figure 3.8 shows the result of applying the AMDF to one window. Equation 3.5 shows the calculation of the fundamental frequency for the AMDF result shown in Figure 3.8.

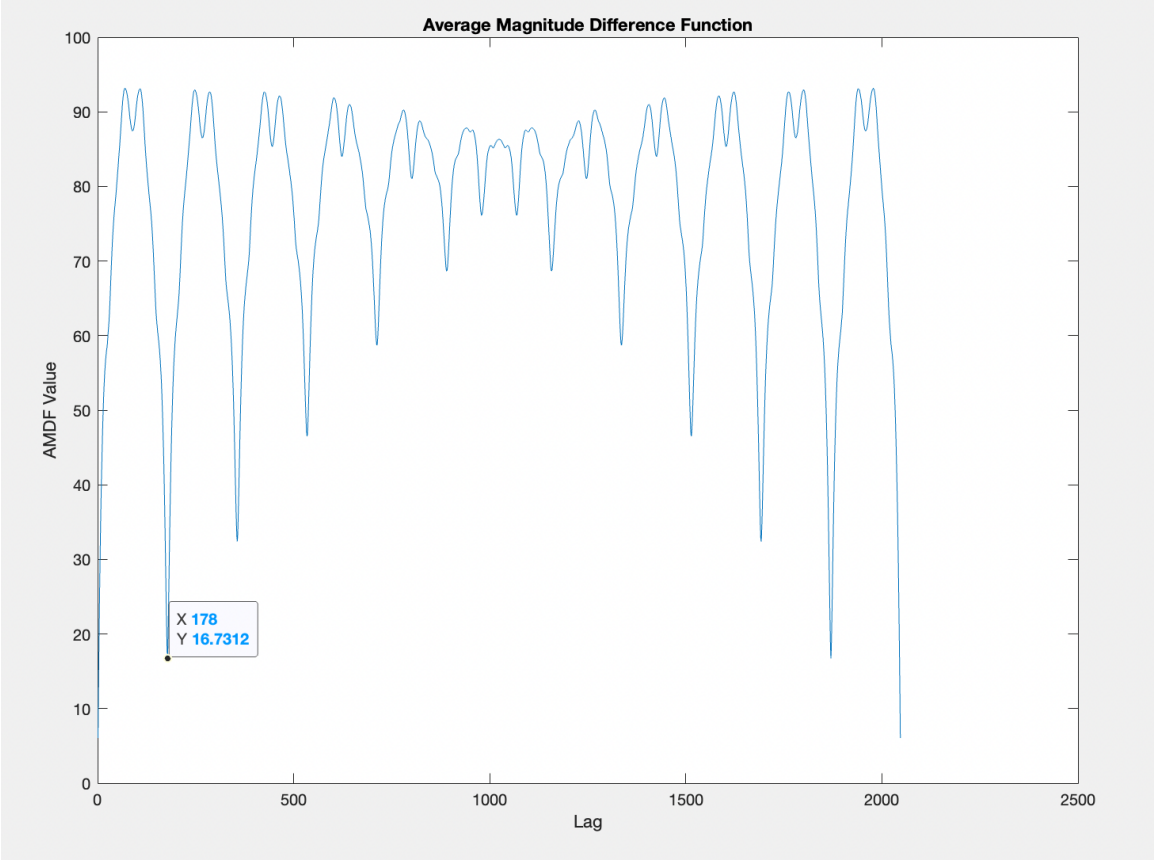

Figure 3.8: AMDF of Audio Window

$$\text{Fundamental Frequency} = \frac{\text{Sampling Frequency}}{\text{Index of First Peak}} = \frac{44100Hz}{178} = \mathbf{247.7528\ Hz} \tag{3.5}$$

3.4.6 Square Difference Function

The square difference function (SDF) was implemented in MATLAB on the same audio segment as the other pitch detection algorithms shown above. This function is very similar to AMDF, but differences in the audio and its shifted version are more penalized. This can be seen in Figure 3.10 because the local minimums reach much lower values than in the AMDF implementation. This can be good for more periodic signals, but can have negative effects on less periodic signals because the local minima for less periodic signals will be less distinct. The flowchart for the SDF algorithm is

shown in Figure 3.9 and is very similar to the AMDF implementation with the main difference being taking the square of the difference.

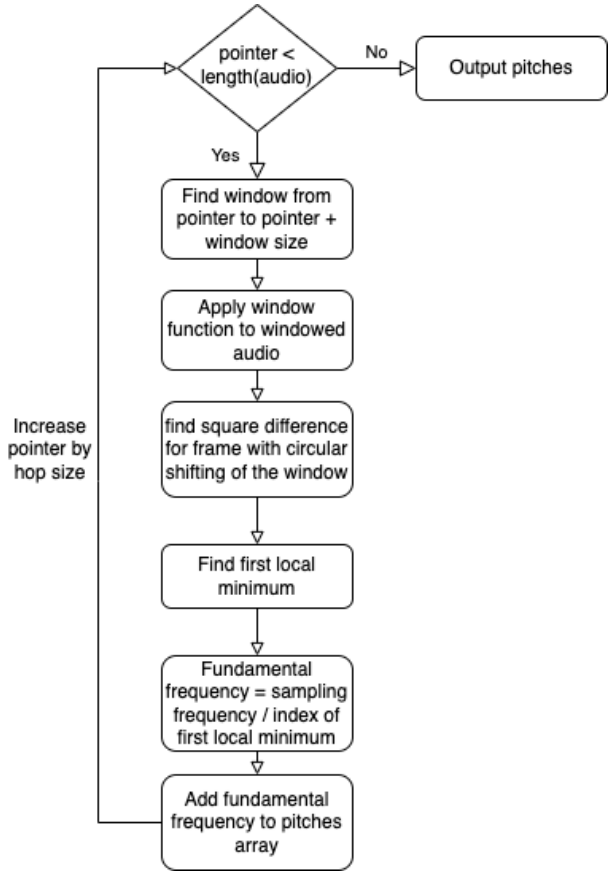

Figure 3.9: SDF Code Flowchart

Figure 3.10 shows the result of the square difference function on the same audio segment. The same first local minimum index is found in this implementation as in the the previous ones and the calculation for the fundamental frequency is shown in Equation 3.6.

$$\text{Fundamental Frequency} = \frac{\text{Sampling Frequency}}{\text{Index of First Peak}} = \frac{44100Hz}{178} = \mathbf{247.7528\ Hz} \tag{3.6}$$

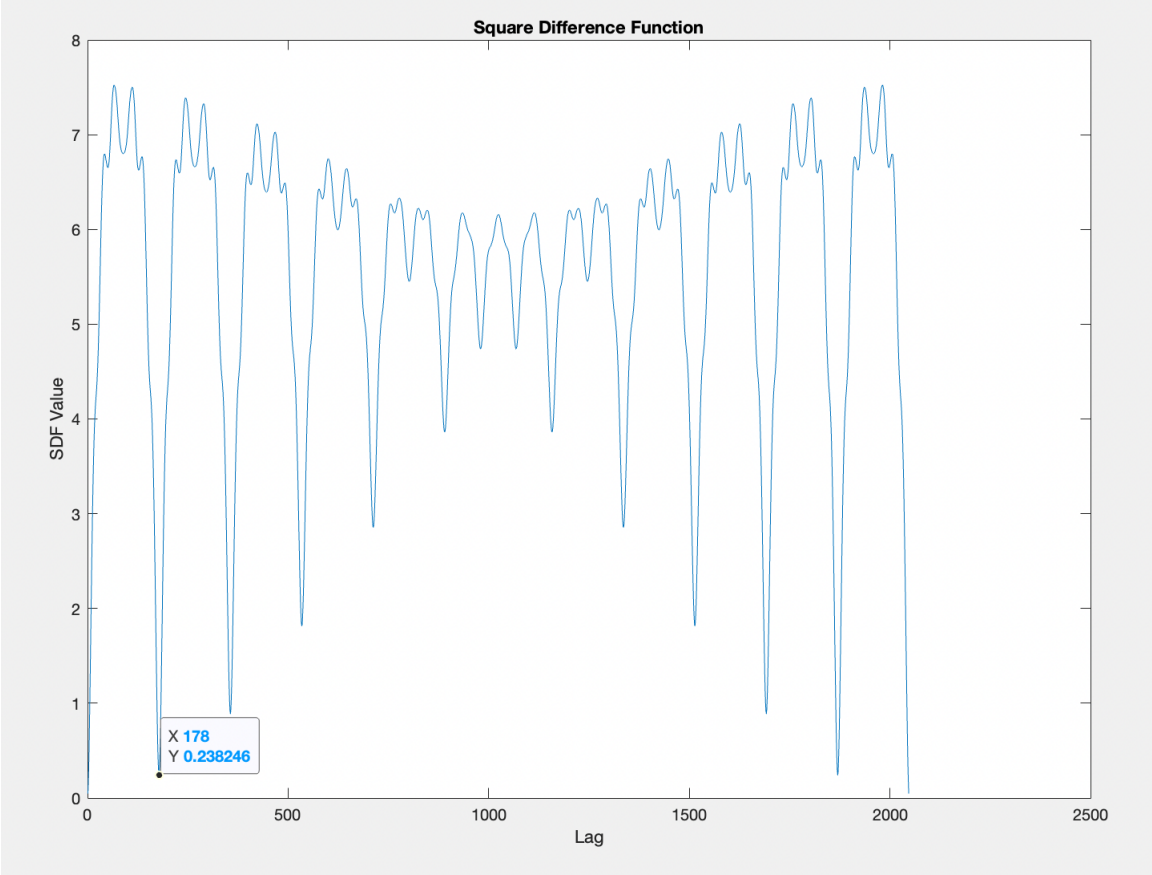

Figure 3.10: SDF of Audio Window

3.4.7 Pitch Detection Comparison

All three pitch detection algorithms that were implemented were tested on the MDB Melody Synth Dataset and the results are shown in the Table 3.2. The metrics were found by testing each algorithm on every song in the resynthesized part of the Melody Synth Dataset and then taking the average value. The algorithms were also tested for the amount of time to process one three minute duration song. The results show that the autocorrelation approach takes the least amount of time and achieves the highest accuracy.

Table 3.2: Summary of Pitch Detection Technique Metrics

Metric	Autocorrelation	AMDF	SDF
Average Percent Error	19.197%	27.3162%	30.8701%
Average Mean Absolute Error	31.7619	50.5618	58.7203
Average RMS Error	327.1112	265.4312	289.1156
Average MS Error	1.8264e+05	9.6700e+04	1.1102e+05
Run time (1 song)	2.315 s	14.791 s	15.251 s

Chapter 4

APPROACH

This chapter discusses the specific approach to creating the proposed music visualization system. The following sections will also discuss all considerations made and limitations encountered when deciding on methods for each element of the project.

4.1 System Overview

The overall music visualization system is shown in Figure 4.1. The input to the system is a music file and the output is a visual of that music file. In between the input and output, MSS and visualization creation occur.

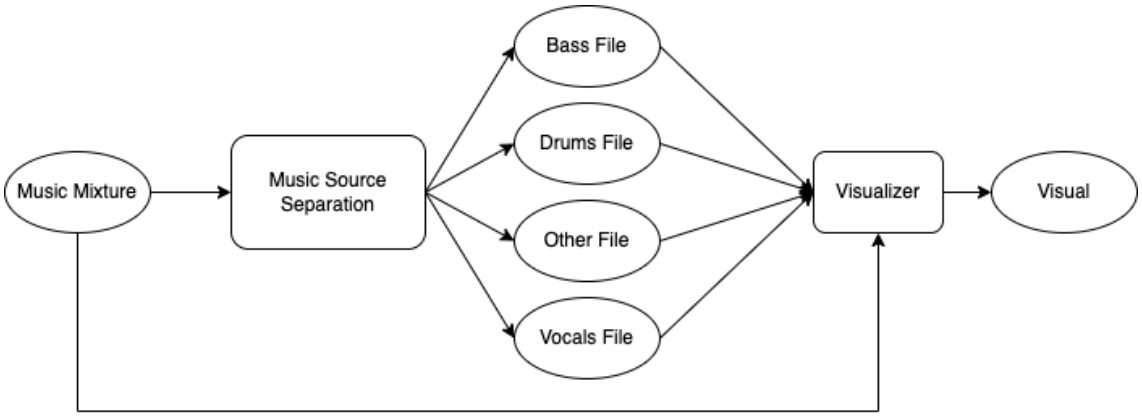

Figure 4.1: System Overview

The visualizer step for one file is also broken down further in Figure 4.2 which shows the processes performed on each separated file. The visualizer takes in all four separated source music files and processes them at the same time. Each file is broken up into segments and the segments are iterated through. For each segment, the pitch,

panning parameter, and average loudness is found and these metrics are used to create a visualization frame. The frame found from the segment of music is then added to the video file and the visualizer checks if the end of the audio has been reached. The output of the visualizer is the 1000x1000 pixels visualization video file.

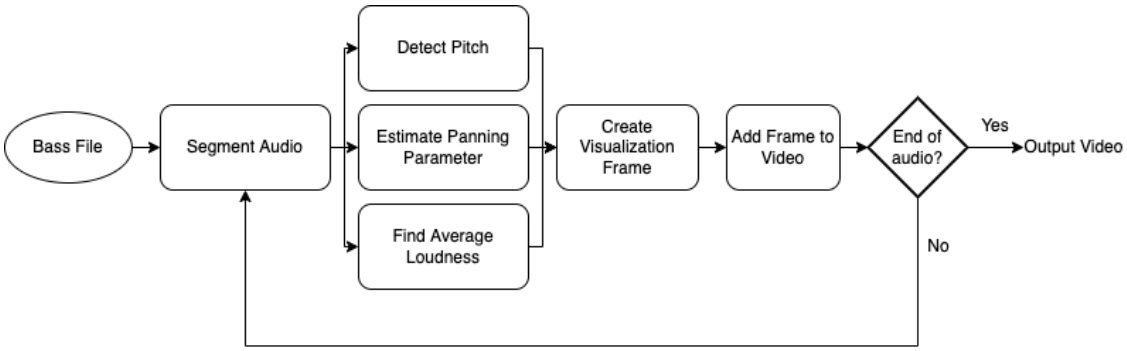

Figure 4.2: Visualizer Overview for One File

4.2 Music File Formats

Music file formats can be broken up into lossless and lossy file types. The most popular lossless audio formats are AIF and WAV files. The most common lossy audio format is MP3 because it maintains a good quality of sound while having a small file size [41]. For most open source MSS applications, the input accepts any type of music file format and outputs WAV files.

4.2.1 Sampling Rates

Every digital audio file has a sampling rate associated with the recording. Files produced with DAWs are often sampled at 44.1 kHz or 48 kHz. This project assumes a sampling rate of 44.1 kHz because this sampling rate is used for the output of the MSS network.

4.3 Music Source Separation

For the MSS part of the project, multiple methods for obtaining separated source files were attempted. The following sections outline the two main approaches for the MSS problem and highlights some of the pros and cons of each method. Based on the comparisons done in Section 3 of this thesis, the Hybrid Demucs method was chosen for the MSS tool because of its high SDR values across the different categories.

4.3.1 Pretrained Neural Networks

Many of the state-of-the-art approaches to MSS are open source projects which provide researchers with pretrained deep neural networks. Having access to the weights from a pretrained neural network is advantageous for multiple reasons. The main advantage is that state-of-the-art results can be achieved without the large amounts of processing power necessary to train a large and accurate neural network. Open source solutions also come with wrappers which will define and perform the preprocessing steps to the neural network. Each open source approach requires users to download the wrappers and dependencies, but these can be avoided by using cloud computing solutions such as Google Colab.

4.3.2 Retraining Networks

Another option that was considered for the MSS portion was retraining a state-of-the-art neural network with new parameters. The main problem posed by this option is the need for considerable computing power. Training neural networks on GPUs can decrease the training time significantly; but without those resources, the revised neural network will take much longer to train and may not be able to achieve the same

results. Using open source projects provides access to a neural network trained by high powered computers and without those training resources, retraining the network will likely not yield state-of-the-art results.

4.4 Pitch Detection

For the pitch detection of each separated source, the stereophonic channels were averaged into a monophonic waveform. The pitch detection algorithm chosen for the final implementation was the autocorrelation algorithm with a window size of 2048. This was chosen for its high accuracy and low processing time. Some modifications were made to the algorithm to improve the accuracy. For the pitch detection algorithm to be accurate, some checks needed to be added to throw out inaccurate results from bad segments of audio. The first such issue arises when the audio segment is silent for more than half of the window time. This caused the autocorrelation result to be very noisy and the peaks selected were often incorrect, resulting in overly high pitch frequency estimates. Figure 4.3 shows the result of performing autocorrelation on a signal that is silent for more than half of the samples. The autocorrelation signal has a large amount of noise on it and this results in the incorrect peak being chosen.

Figure 4.4 shows a zoomed in view of the waveform shown in Figure 4.3. The labels demonstrate the actual index that should be picked for the pitch detection (index = 128) and the first peak caused by noise (index = 5). In order to avoid incorrect peak picking from silent segments, the pitches resulting from these segments are set to zero.

Removing audio segments with more than 50% silence helped to eliminate some major outliers in the pitch detection algorithm, but some other outliers were still present. Some of the autocorrelation results were very far from the annotated value because of

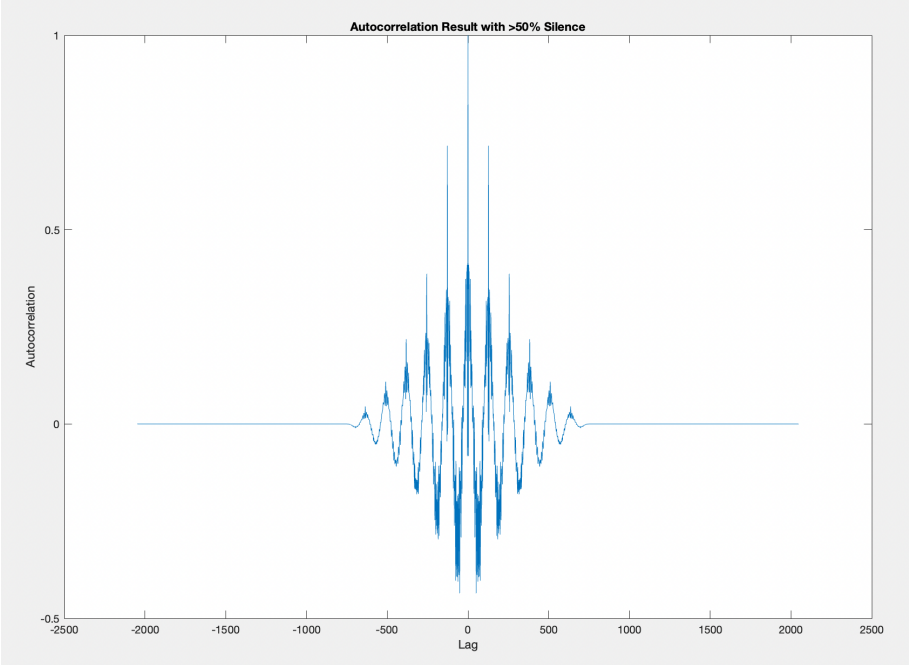

Figure 4.3: Autocorrelation Result With $> 50\%$ Silence

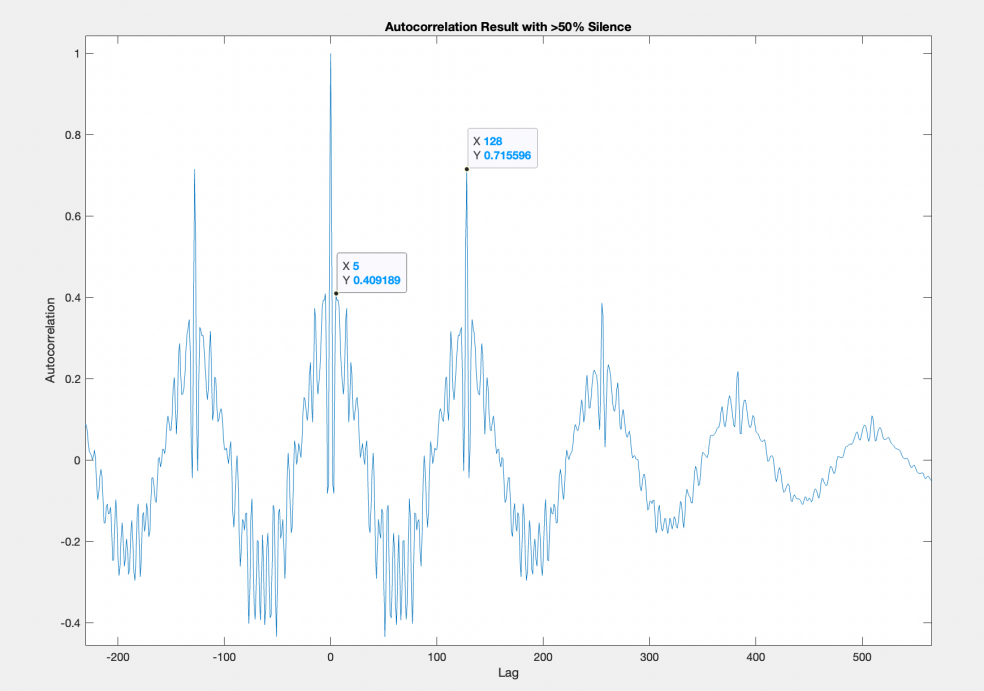

Figure 4.4: Zoomed Version of Figure 4.3

noise not caused by silence. These outliers were handled by adding a one-dimensional 20th order median filter to the autocorrelation results. The order of the filter was

chosen empirically through testing erroneous and normal autocorrelations with different median filter lengths. An example of a waveform that would produce an incorrect pitch estimation with a median filter applied is shown in Figure 4.5.

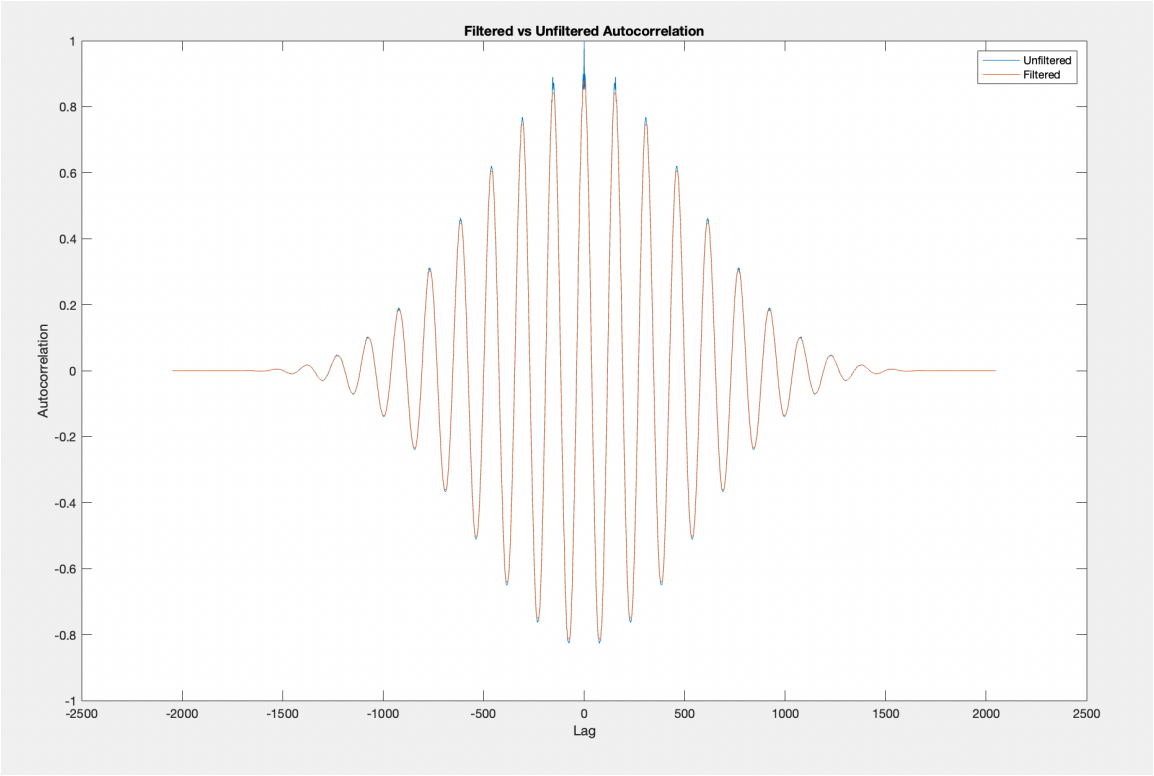

Figure 4.5: Filtered vs Unfiltered Autocorrelation Results

Figure 4.6 shows the center and first peaks zoomed in. This demonstrates how having noise in the autocorrelation can result in incorrect peaks being picked for the pitch estimate. In this case, without the median filter the first peak will be at index 5 which would result in a pitch of $\frac{44100}{5} = 8820Hz$. 8820 Hz is well out of the range of instruments that would be in a typical mix for a song. In fact, fundamental frequencies above 8 kHz are often considered extended high frequencies and are used primarily to test for hearing loss [42]. With the median filter applied, the first peak will be at index 153 so the pitch estimation will be $\frac{44100}{153} = 288.2353Hz$. This pitch is much more reasonable so the median filter was an effective solution for eliminating outliers.

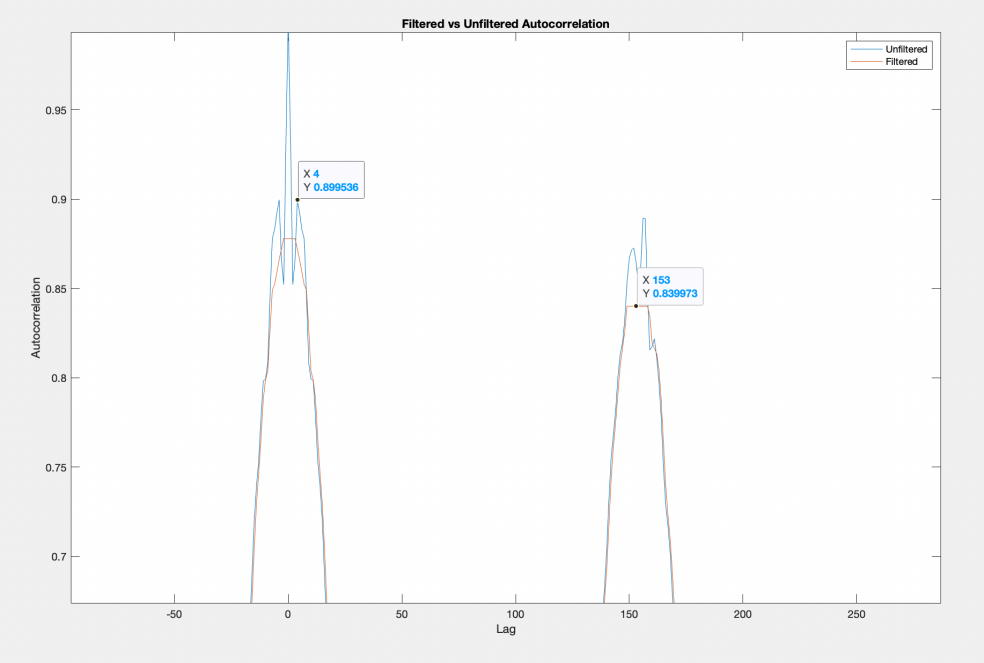

Figure 4.6: Zoomed Version of Figure 4.5

4.5 Localization

The localization algorithm discussed in Section 2.4 was implemented to predict the panning parameters for the different audio files. In order avoid the location estimation from moving erratically, the following adjustments were made. The first adjustment assumes that the panning parameter does not change as often as other characteristics such as pitch. The window size of the panning parameter estimation was set to 10240 samples which checks the location every five pitch windows. The second adjustment was made to fit the localization values to the visualization. Since the output visualization video is 1000x1000 pixels and the localization parameter ranges from 0-180 degrees, the localization angle values were re-mapped to image pixel positions in the range 0 to 1000.

4.6 Loudness Approximation

Approximating the loudness for each track in the song consisted of summing the absolute values of the channels. This assumes that the constant power panning law is applied to the stereophonic mixture. The absolute value of the left and right channels will be relative to the total power [24]. This provides the visualization with a good quantitative representation of the loudness.

4.7 Visualization

The visualization was implemented using the results from the pitch detection, localization, and loudness approximation for each of the tracks resulting from the MSS. The original sketch for the visualization idea is shown in Figure 4.7. The placement of each dot on the x-axis is defined by its panning parameter estimation and the y-axis location is defined by the pitch. The diameter of the circles are adjusted based on the loudness during that frame.

The visualization was compiled in MATLAB [43] using a blank black 1000x1000 image and the InsertShape function. Each frame is made by taking the current pitches, location, and loudness and translating these parameters to shapes. The audio is read in using the audioRead function in MATLAB and it is important to note that this function normalizes the audio between -1 and 1. As a result of this, when the average loudness for each frame is computed, the values are very small compared to the scale of the video. To remedy this, the average loudness in each frame is multiplied by 500. The frame rate chosen for the project is 30 frames per second (fps).

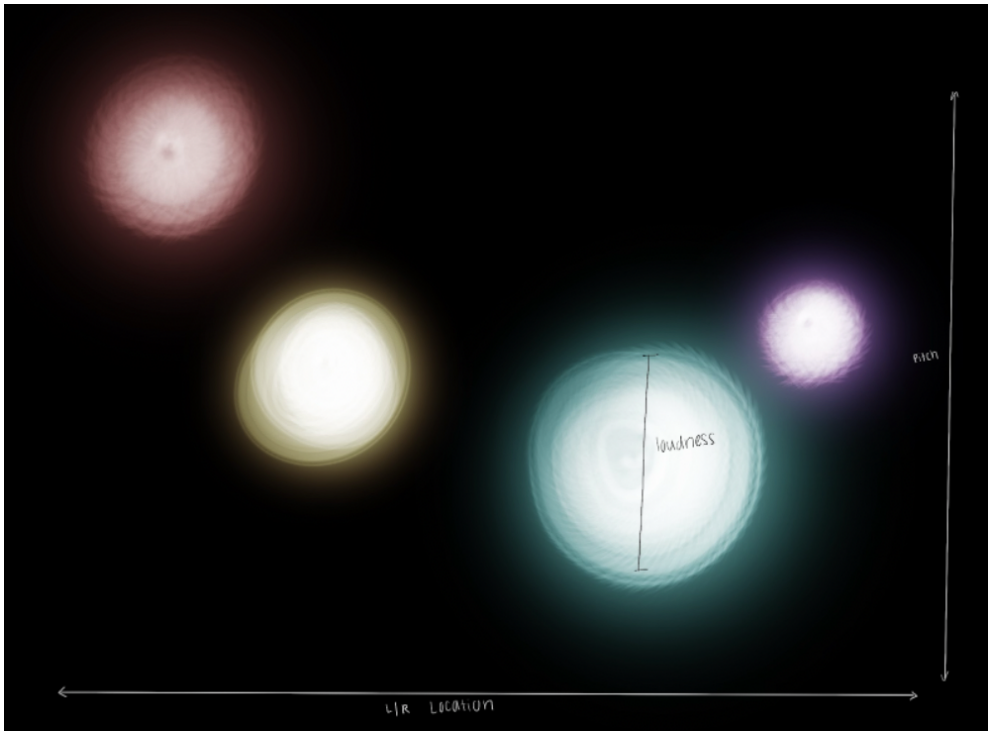

Figure 4.7: Original Visualization Sketch

4.7.1 Visualization Colors

In order to incorporate some of the research on music and color, the visualization allows users to pick the color of each of the sources. This allows users to have their visualization tailored to how they interpret music and color. The visualization also changes the shades of the colors to be slightly darker as the pitches get lower.

4.7.2 Video File Types

Using MATLAB also imposed some limitations on the file types. In order to create a video from scratch on MATLAB, you have to use the VideoWriter function which allows you to create a video by defining each frame in the video. The biggest issue with this function is that it does not allow the user to add audio into the file. The workaround in MATLAB for this issue is to use the VideoFileWriter function on the

already created video to add in the audio for each frame. This incurs unnecessary additional processing time and power but is the only workaround available at this time. The project was implemented on an Apple Macbook computer which meant there were some limitations for file types that could be used. The VideoFileWriter function allows users to output videos in AVI, MPEG4, or MJ2 format. On Apple computers, MPEG4 is the ideal output type because it does not require any additional downloads to play the video files. Unfortunately, the only way to write audio to a file on MATLAB is to use an AVI file. The system outputs an AVI file, which requires additional software downloads for Apple users to view the final product. The workaround for this is using Handbrake [44] which is an open source tool for video transcoding. This can be used to convert from AVI to MPEG4 files.

Chapter 5

RESULTS

This chapter will discuss the results from implementing the approach from Chapter 4 and outlines how the system can be used.

5.1 Music Source Separation

The MSS problem was approached two different ways: using a pretrained network and training a predefined network. Using the pretrained neural network gave state-of-the-art results in just a few minutes. When running the neural network on a CPU, the source separation can execute in the same time as the length of the song for each source.

Retraining a predefined neural network was also attempted, but could not be completed due to long training times. The training for a vocals neural network was initiated with the intention of running for 300 epochs or until a low error was found. The program was predicting around 300 hours of training time after just two epochs. In an attempt to reduce the training time, the learning rate was increased from 0.001 to 0.1 and the number of epochs was decreased to 150. Even with these changes, the training time on a CPU was still hundreds of hours and was not realistic for the scope of this project.

5.1.1 Performing Source Separation Using a Pretrained Neural Network

In order to perform source separation using a pretrained neural network, the user needs to have the dependencies for the specific framework which are usually outlined in the code repository. Once this is downloaded, the system can be called from a terminal window. With Hybrid Demucs and all the dependencies installed, the user can simply call Demucs from terminal followed by the mixture to be separated as shown in Listing 5.1.

```
demucs "/Users/hannahchookaszian/MATLAB/Thesis/MUS/train/A Classic
   Education - NightOwl/mixture.wav"
```

Listing 5.1: Calling Demucs in Terminal Example

5.2 Visualization

The visualization was put together in MATLAB and the results are best viewed in video format. For this results section, some individual frames will be presented below. Each visualization can be customized by changing the colors used to represent the different source classes. Figure 5.1 shows the window that will appear to allow the user to select the colors for the visualization.

The colors also get darker as the fundamental frequency decreases. Note that the drums remain centered for every visualization because the pitches change very drastically from frame to frame.

Figure 5.2 shows one frame from a visualization. From this frame, the user can see that the vocals and bass are panned to the left while the "other" source is panned

Figure 5.1: Color Selection For Visualization

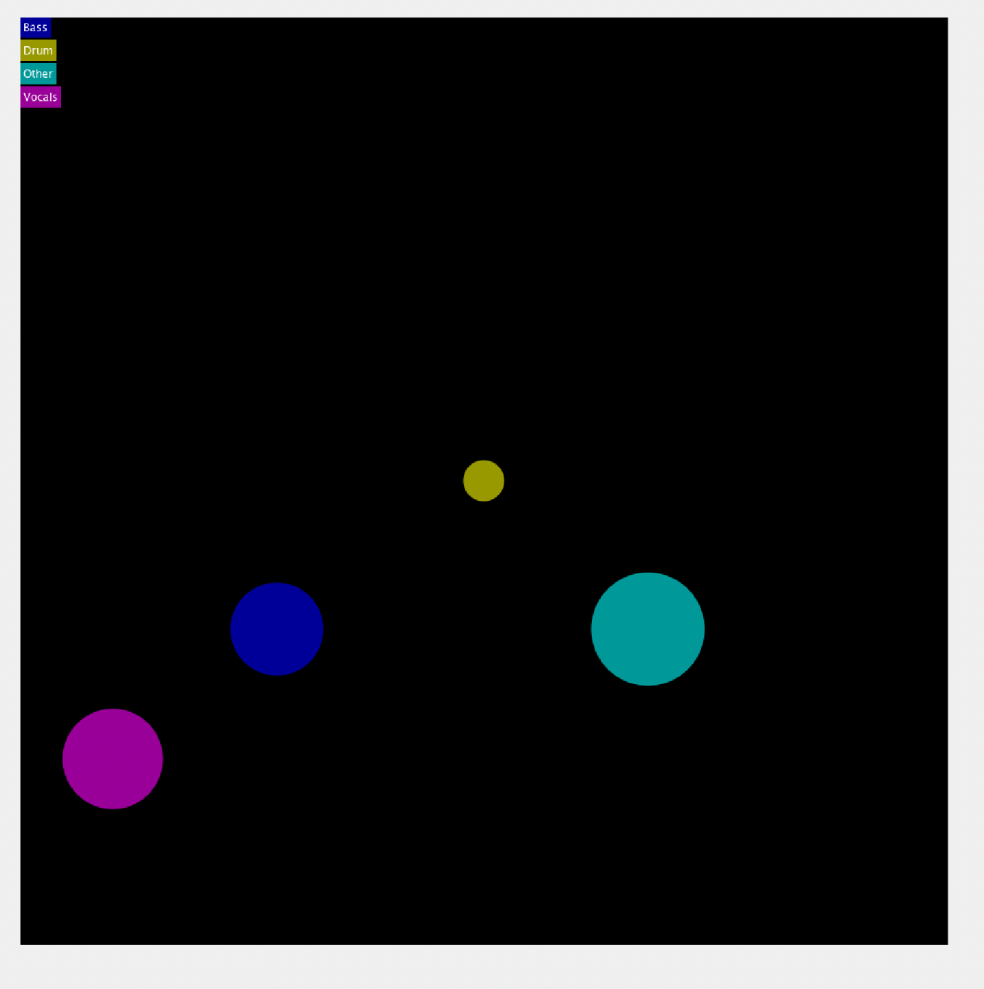

Figure 5.2: Frame from Visualization

to the right. The pitches are also relatively low at this point in the song because the colors are a slightly darker shade and the y-axis locations are low.

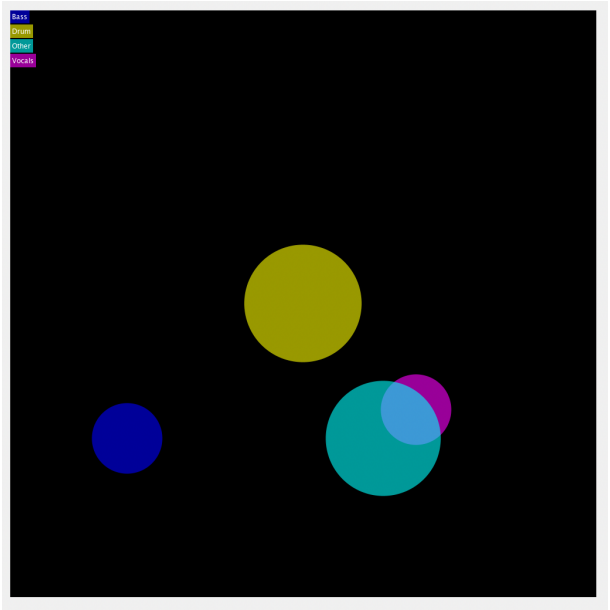

Figure 5.3: Frame from Visualization

Figure 5.3 shows another frame from the visualization of the same song. In this frame, the drums and other sources are louder because their diameters are larger. The vocals are also panned further to the right on the horizontal plane. Figure 5.4 shows a frame from the visualization of the same song where the vocals are higher. At this point in the song the fundamental frequency of the vocals is higher, so the circle representing the vocals is moved up higher on the vertical axis.

It should also be mentioned that if a source is not present in the mix at an interval of time it will not be in the visualization at that time. For example, Figure 5.5 shows a frame from a visualization where only the "other" source is contributing to the overall mix. In this case the bass, drums, and vocals sources are not in the song at this time and therefore do not appear in the visualization at this time.

Figure 5.6 shows a frame from a visualization where the song does not have any vocals at that time.

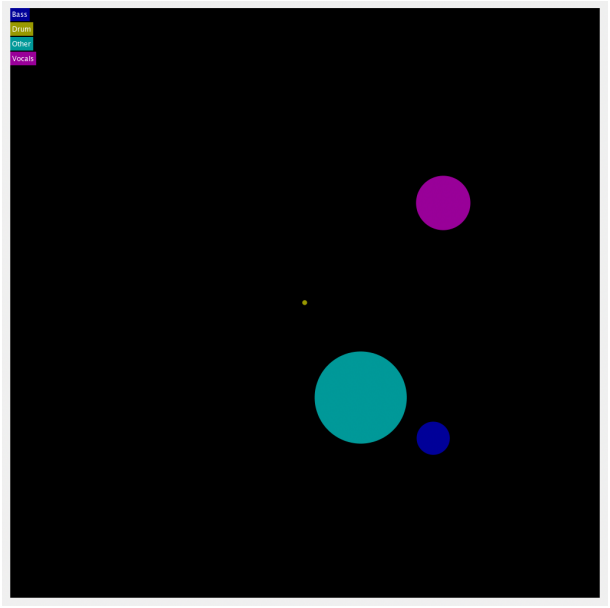

Figure 5.4: Frame from Visualization With High Vocals

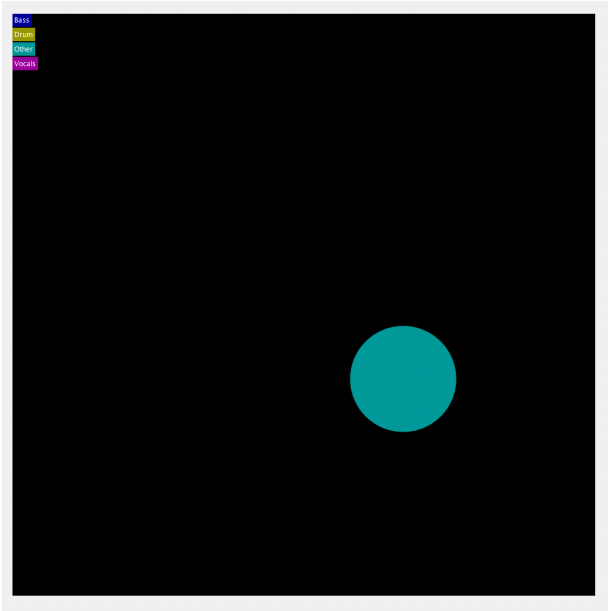

Figure 5.5: Frame from Visualization With Only "other" Source Playing

Each visualization takes slightly longer than the length of the song to compile. For instance, for a 5 minute 10 seconds song, the visualization took 390.768 seconds (6 minutes 30 seconds) to compile. The first attempt at the visualization required over

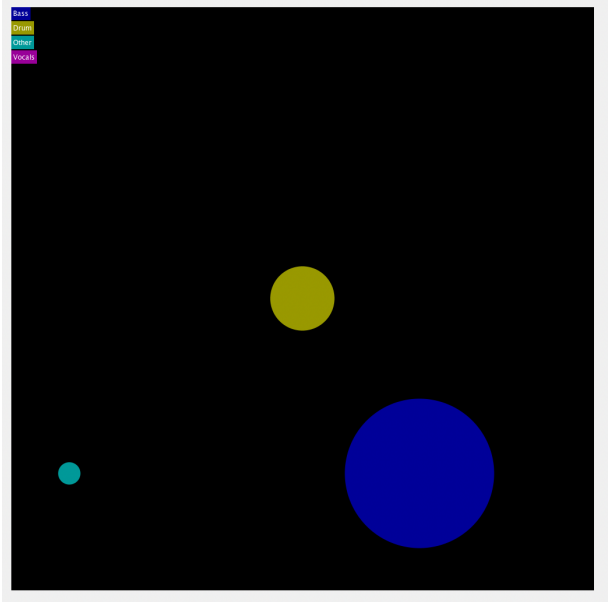

Figure 5.6: Frame from Visualization With "vocals" Source Not Playing

30 hours to compile so this was an improvement over that, but this time can still be improved further.

5.2.1 Creating a Visualization

Note that creating the visualization requires MATLAB to be downloaded. The function to run the visualization is called "new_visualization.m" and it requires the file path for the mixture file (the same file that was separated) and the name of the output file. The function assumes that the sources were separated using the Hybrid Demucs source separation and will get the source files from the output of Demucs. The function shown in Listing 5.2 should be run in MATLAB.

```
1 new_visualization("/Users/hannahchookaszian/MATLAB/Thesis/MUS/train/
    A Classic Education - NightOwl/mixture.wav", "output")
```

Listing 5.2: Creating a Visualization

Chapter 6

CONCLUSIONS

The goal of this thesis was to create a music visualization system which could take in any song file and create a representation of the different sources within it. Overall, the project was successful in creating a framework that can be used for visualizing any desired stereophonic recording. This thesis focused on using MSS and its outputs to create a unique visualization.

This thesis also reviewed different MSS methods and provides useful information on how MSS neural networks are architected. This thesis gave an example of how MSS can be used and explained in more detail what the inputs and outputs of MSS networks are. Complicated neural networks can often be black boxes and this thesis attempted to explain their structures and how they are trained.

One of the lessons learned from this thesis is the importance of comparing different methods and approaches for both computing power and time involved in the processing. One of the main issues that arose during this project was a lack of sufficient computing power. Using remote computing power can be extremely helpful in these situations and could have helped speed up training the MSS. Trying different pitch detection methods also proved to be very helpful because the time differences in the algorithms were quite substantial.

Overall, this thesis took a unique approach to music visualization and surveyed MSS techniques. It works to help users understand how different sources interact in a song together to create a musical piece.

6.1 Future Work

Future work on this project could take many different directions because this thesis covers many different topics. The first of which is creating a custom MSS method for this visualization system and attempting to extract more sources from each song. This project could also be expanded by changing the visualization and incorporating more music information into the overall visual. Employing a more sophisticated computer animation system than Matlab's simple shape insertion function could also produce a more elaborate and possibly more engaging visualization. The algorithm to create the visualization could also be optimized to reduce the processing time and the computing power needed. Since the project currently uses both Python and MATLAB, converting the code to use only one language would be helpful to create a single package for the system.

BIBLIOGRAPHY

[1] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, 2015.

[2] K. Zhang, Y. Guo, X. Wang, J. Yuan, and Q. Ding, "Multiple feature reweight densenet for image classification," *IEEE Access*, vol. 7, pp. 9872–9880, 2019.

[3] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," 2018.

[4] B. Bhattarai, Y. R. Pandeya, and J. Lee, "Parallel stacked hourglass network for music source separation," *IEEE Access*, vol. 8, pp. 206016–206027, 2020.

[5] H.-S. Choi, J. Lee, and K. Lee, "Spec2spec: Towards the general framework of music processing using generative adversarial networks," *Acoustical Science and Technology*, vol. 41, no. 1, pp. 160–165, 2020.

[6] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, 2019.

[7] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, pp. 1256–1266, aug 2019.

[8] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," 2019.

[9] R. Sawata, S. Uhlich, S. Takahashi, and Y. Mitsufuji, "All for one and one for all: Improving music separation by bridging networks," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 51–55, 2021.

[10] A. Défossez, "Hybrid spectrogram and waveform source separation," 2021.

[11] A. Roginska and P. Geluso, *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio*. Audio Engineering Society Presents, Taylor & Francis, 2017.

[12] H. B. Lima, C. G. R. D. Santos, and B. S. Meiguins, "A survey of music visualization techniques," *ACM Comput. Surv.*, vol. 54, jul 2021.

[13] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[14] E. Vincent, T. Virtanen, and S. Gannot, *Audio Source Separation and Speech Enhancement*. Wiley, 2018.

[15] S. Mirzaei, H. Van Hamme, and Y. Norouzi, "Under-determined reverberant audio source separation using bayesian non-negative matrix factorization," *Speech Communication*, vol. 81, pp. 129–137, 2016. Phase-Aware Signal Processing in Speech Communication.

[16] D. FitzGerald, A. Liutkus, and R. Badeau, "Projet — spatial audio separation using projections," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 36–40, 2016.

[17] A. Liutkus, D. Fitzgerald, and R. Badeau, "Cauchy nonnegative matrix factorization," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–5, IEEE, 2015.

[18] J. Le Roux, J. R. Hershey, and F. Weninger, "Deep nmf for speech separation," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 66–70, 2015.

[19] Y. Mitsufuji, S. Koyama, and H. Saruwatari, "Multichannel blind source separation based on non-negative tensor factorization in wavenumber domain," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 56–60, 2016.

[20] A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, and L. Daudet, "Kernel additive models for source separation," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4298–4310, 2014.

[21] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4580–4584, 2015.

[22] A. S. Master, *Stereo music source separation via Bayesian modeling*. PhD thesis, 2006. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-09-28.

[23] J. M. Hjerrild and M. G. Christensen, "Estimation of source panning parameters and segmentation of stereophonic mixtures," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 426–430, 2018.

[24] "Loudness concepts and panning laws."

[25] "Tech stuff - frequency ranges."

[26] L. Rabiner, M. Cheng, A. Rosenberg, and C. McGonegal, "A comparative performance study of several pitch detection algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 5, pp. 399–418, 1976.

[27] P. S. Rao, S. Khoushikh, S. Ravishankar, R. A. Ananthkrishnan, and K. Balachandra, "A comparative study of various pitch detection algorithms," in *2020 5th International Conference on Computing, Communication and Security (ICCCS)*, pp. 1–6, 2020.

[28] C. P. Singh and T. K. Kumar, "Efficient pitch detection algorithms for pitched musical instrument sounds: A comparative performance evaluation," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1876–1880, 2014.

[29] P. McLeod, "Fast, accurate pitch detection tools for music analysis," 2008.

[30] M. Ross, H. Shaffer, A. Cohen, R. Freudberg, and H. Manley, "Average magnitude difference function pitch extractor," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 5, pp. 353–362, 1974.

[31] A. M. Noll, "Cepstrum pitch determination," *The Journal of the Acoustical Society of America*, vol. 41, no. 2, pp. 293–309, 1967.

[32] R. Hiraga, F. Watanabe, and I. Fujishiro, "Music learning through visualization," pp. 101– 108, 02 2002.

[33] C. Curwen, "Music-colour synaesthesia: Concept, context and qualia," *Consciousness and Cognition*, vol. 61, pp. 94–106, 2018.

[34] S. E. Palmer, K. B. Schloss, Z. Xu, and L. R. Prado-León, "Music–color associations are mediated by emotion," *Proceedings of the National Academy of Sciences*, vol. 110, no. 22, pp. 8836–8841, 2013.

[35] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*, pp. 293–305, 2018.

[36] Y. Mitsufuji, G. Fabbro, S. Uhlich, F.-R. Stöter, A. Défossez, M. Kim, W. Choi, C.-Y. Yu, and K.-W. Cheuk, "Music demixing challenge 2021," *Frontiers in Signal Processing*, vol. 1, jan 2022.

[37] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[38] E. Buber and B. Diri, "Performance analysis and cpu vs gpu comparison for deep learning," pp. 1–6, 10 2018.

[39] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez Gutiérrez, and J. P. Bello, "An analysis/synthesis framework for automatic f0 annotation of multitrack datasets," in *Hu X, Cunningham SJ, Turnbull D, Duan Z. ISMIR 2017 Proceedings of the 18th International Society for Music Information Retrieval Conference; 2017 Oct 23-27; Suzhou, China.[Suzhou]: ISMIR; 2017.*, International Society for Music Information Retrieval (ISMIR), 2017.

[40] D. Purves, G. J. Augustine, D. Fitzpatrick, L. C. Katz, A.-S. LaMantia, J. O. McNamara, and S. Mark Williams, *The Audible Spectrum*. Sunderland, MA: Sinauer Associates, 2001.

[41] J. Hass, "Chapter five: Digital audio."

[42] L. L. Hunter, B. B. Monson, D. R. Moore, S. Dhar, B. A. Wright, K. J. Munro, L. M. Zadeh, C. M. Blankenship, S. M. Stiepan, and J. H. Siegel, "Extended high frequency hearing and speech perception implications in adults and children," *Hearing research*, vol. 397, pp. 107922–107922, Nov 2020. 32111404[pmid].

[43] MATLAB, *9.7.0.1190202 (R2022a)*. Natick, Massachusetts: The MathWorks Inc., 2022.

[44] HandBrake, "Handbrake/handbrake: Handbrake's main development repository."

[45] "Cal Poly Github." http://www.github.com/CalPoly.

APPENDICES

Appendix A

CODE APPENDIX

The code for the visualization can be found in this GitHub Repository:

Link to Github Repository

Appendix B

VISUALS APPENDIX

A link to some of the visuals that have been created thus far can be found below:

Link to Visuals