**William Victor Simmons**

# Techniques for Harmonic Sound Separation

Computer Science Tripos – Part II

St Catharine's College

May 5, 2017

# Proforma

| | |
|---|---|
| Name: | **William Victor Simmons** |
| College: | **St Catharine's College** |
| Project Title: | **Techniques for Harmonic Sound Separation** |
| Examination: | **Computer Science Tripos – Part II, June 2017** |
| Word Count: | **11855 words** |
| Project Originator: | William Victor Simmons |
| Supervisor: | Dr David Greaves |

## Original Aims of the Project

Within this project, I aimed to produce a software solution which, when provided with a (monophonic or stereophonic) audio file containing a superposition of $k$ harmonic sounds and the number $k$, produces $k$ audio files describing estimates of the superposed sounds. These reconstructions should be audibly similar to the true individual sounds prior to mixing. The principle algorithm used should follow the Sinusoidal Modelling approach described by Virtanen and Klapuri [26] and a comparison should be made with using Non-negative Matrix Factorisation to extract elementary structures of the sound as used by Virtanen [25].

## Work Completed

The solution has been structured as a collection of interchangeable C++ code fragments with a command-line tool demonstrating their combined use for sound separation. `WavFileManager` provides a wrapper to the JUCE [3] framework in order to retrieve the stereophonic audio streams from `wav` files and create the corresponding files for the reconstructed outputs. Due to the lack of libraries with appropriate inverse-STFT (short time Fourier transform) functions, I have added implementations of STFT and its inverse in `Transform`, along with Non-negative Matrix Factorisation. The operations on sinusoids worked well with an object-oriented model, giving rise to the `SinusoidalTrajectory` and `SinusoidalTrajectoryPoint` classes. Implementations of Lloyd's algorithm for $k$-means clustering and a soft-clustering equivalent are provided by `Cluster`. The `Separation` class brings all of these together to give a complete solution for the harmonic sound separation problem. `BASSEval` gives some methods to evaluate the separation performance

according to the standard metrics of Signal to Distortion, Interference, Noise, and Artefact Ratios, as well as the methods to run the solution on the validation and test sets.

## Special Difficulties

There are no special difficulties to report from the completion of this project.

# Declaration

I, William Victor Simmons of St. Catharine's College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed [signature]

Date [date]

# Contents

# List of Figures

# Acknowledgements

Thank peeps

# Chapter 1

# Introduction

In this project, I have been investigating a few techniques for harmonic sound separation based on feature extraction and clustering. I aimed to provide a comparison of the performance between using sinusoidal trajectories and spectrogram factor components as the elementary sound structures, as well as evaluating how the accuracy of the reconstructions vary with factors such as signal-to-noise ratio, stereo distance between the sources and pitch offset.

## 1.1 Motivation

The physical world contains a lot of harmonic signal generators, many of which appear as both the desired signal and as noise in our sensor data. As humans, we have evolved to be able to isolate a single sound source from a large combination, enabling us to identify the music played by each instrument in a song or listen to a particular conversation in a crowded room, as evidenced by Cherry's experiments on the "cocktail party problem" [9]. The ability for computers to solve this problem is essential for personal assistant applications. Sound separation can be viewed as a special case of the problem of noise-removal, which is a necessary process for a lot of autonomous systems. The instance of the problem that this project considers is aimed at the requirements of audio analysis and the editing of music.

It is important to note that the general problem of perfectly separating $k$ sounds given fewer than $k$ channels is ill-posed under Hadamard's conditions [12] since solutions are not unique. Sound separation is the inverse problem of mixing (taking $k$ audio streams and combining them additively into a single stream), but it is trivial to construct two sets of different sounds which produce the same result after mixing. This motivates the need for assumptions about the structures of the sounds to limit the number of valid inverses and give a problem that we can reliably solve.

## 1.2 Overview of Sound Separation

One way that we can make the separation problem well-defined is by only allowing a small set of known sounds to be used. This converts the sound separation problem to one

of projecting a sound onto a fixed basis. However, this "informed" approach to sound separation limits the usage to those sounds that the system knows as it cannot generalise outside of this range. Some systems can ask for other hints to aid the separation process. This can come in the form of a human user, for example, specifying the approximate stereo positions of the sound sources, providing the number of sounds present, or by identifying regions of the spectrogram belonging to the same sound.

For this project, I have chosen to consider uninformed sound separation methods (also known in literature as Blind Audio Source Separation) due to the potential for more general applications. However, in order to constrain the problem, I am enforcing the assumption that all of the sound sources are harmonic (at any time, the peaks in the Fourier spectrum occur at integer multiples of some base frequency), forcing there to be some structure that we can exploit in the separation process. Given the focus on musical applications in this project, this assumption often holds since tonal (non-percussive) instruments are harmonic, behaving as one-dimensional oscillators with multiple related resonant frequencies. I am assuming that the system is provided with the number of sounds present since, even if the user is incapable of doing this, we could produce programs that can estimate this quantity based on estimations of onset times or using pitch-detection algorithms (such as looking for strong cepstral peaks to obtain the base frequencies [19]).

The solution produced was designed by following the work of Virtanen and Klapuri [26] which separated sounds by identifying key sinusoids and their amplitude and frequency envelopes, then grouping them based on the similarity of the envelopes to recover the original sounds. Within this investigation, we will consider this under the more general approach of identifying the elementary structures in the sounds (herein referred to as "sound elements") and then clustering them such that those elements from the same sound source are likely to be in the same cluster. The main comparison within the project is between using simple sinusoids and using spectrogram factor components, as in other work by Virtanen [25], as the elements of the sounds.

## 1.3 Current Implementations

Commercial software already exists for extracting instrument tracks from a piece of music and manipulating them. Celemony's Melodyne [1] can split a pholyphonic audio track into its constituent notes and allows users to edit these notes in time and pitch or to manually select all those from a given instrument to isolate it. On the other side of the spectrum, MAGIX's SpecraLayers Pro [5] is an audio editing suite which focuses on enabling control over the spectrogram of the sound. This assists users to separate sounds by providing them with tools, such as the Harmonics Selection tool, enabling them to define the spectral regions occupied by each sound and edit them separately. Whilst this does well at handling spectral leakage (where a pure sinusoid does not fit nicely into a single frequency bin in the Fourier transform, and so spreads over the other nearby bins), it does not perform any automatic separation, opting for a more user-assisted approach.

Whilst this project is focussing on separating audio given two channels, beamforming techniques have been widely used in radar and sonar systems. This uses an array of

sensors, positioned (or artificially weighted and delayed) such that signals arriving from a specific direction can be amplified very well, effectively extracting these from the remaining noise. This does not make the harmonic assumption but requires many more channels than are usually provided for music applications.

## 1.4 Overview of the Dissertation

Over the course of this dissertation I will present the approach I used to investigate the techniques for harmonic sound separation and the results of testing my solutions. The Preparation chapter outlines the decisions involved in the design of a solution. The Implementation chapter goes through each section of the solution, describing how it works in detail and any problems or points of interest which arose when building them. In the Evaluation chapter, I compare the options for each section of the main algorithm and show some typical outputs for inputs with varying degrees of separability. I will finally conclude by discussing the effectiveness of this solution, the successfulness of this project against its goals and expectations, and the current state of the field.

# Chapter 2

# Preparation

For ill-posed problems such as general sound separation, solutions may not be unique and so identifying solutions in a sensible manner mat not be obvious since there is not always a clear goal to aim for. In such situations, it is necessary to consider the properties that we wish our outputs to have and make design choices to drive the solution towards this.

## 2.1 Choice of Overall Process

The principal work by Virtanen and Klapuri [26] adopted the following approach: in each frame of the Short-Time Fourier Transform, spectral peaks are detected and tracked over successive frames to obtain a set of sinusoidal trajectories (pure sinusoids with time-varying frequencies and amplitudes); small breaks within clearly continuous trajectories are interpolated; the trajectories are then grouped into two sets such that the total "perceptual distance" between each pair from the same set is minimised and the sounds are reconstructed from the trajectories. This has been shown to successfully separate a mixture of two harmonic sounds from a monophonic input. I found this method sensible since each significant structure in the spectrogram must have been present in one or both of the original sounds, and so we can reconstruct them by identifying some elementary structures and using assumptions about the sounds to group them sensibly.

It is possible to separate sounds without breaking the audio apart into some element set and recombining them. For instance, we can consider each microphone feed as a linear sum of the individual signals, weighted according to the distances from their sources to the microphone. With an array of $k$ microphones, this gives us $k$ simultaneous equations which is solvable for up to $k$ sounds. We could take this a step further and use differences between arrival times, phases or amplitudes of notable elements of the inputs to estimate the locations of the sound sources, removing the need to provide this information. Methods like this alone can struggle in many situations since the number of separable sound sources is limited by the hardware, movement of the sources can complicate the process, and sounds approaching the microphone array from the same direction cannot be distinguished.

According to the work of Bregman [8], humans have been found to organise auditory scenes based on proximity in time or frequency, harmonic concordance (how likely it is that

two frequencies were produced as harmonics of the same base frequency), common onset, offset, frequency and amplitude modulations and spatial proximity. It can be suggested that the system described here is attempting to mimic human auditory scene analysis by grouping sound elements by these properties.

## 2.2 Choice of Sound Elements

Since our heuristics are based both in the time and frequency domains, it would be wise to select our sound elements as regions of interest from the spectrogram. By assuming that our sound sources are clearly separated in the frequency domain, we find that each region of significant magnitude will correspond to one of the harmonics from one of the sources. Each of these will be a single sinusoid (shown as a thin line on the spectrogram) with some time-varying amplitude and frequency. [26] describes the following measures of distance between a pair of these trajectories:

$d_f$ describes the distance according to the frequency envelope. This captures the idea that musical notes played with vibrato (or natural frequency modulations) should experience similar variations in frequency across its harmonics and this will be different from notes without vibrato or with vibrato at a different rate. Where the two trajectories, $i$ and $j$, are both simultaneously present in the time interval from $t_1$ to $t_2$ ($f_i$ and $f_j$ are the average frequencies of the two trajectories over this interval):

$$d_f(i,j) = \frac{1}{t_2 - t_1 + 1} \sum_{t=t_1}^{t_2} \left( \frac{f_i(t)}{f_i} - \frac{f_j(t)}{f_j} \right)^2 \tag{2.1}$$

$d_a$ is an equivalent distance metric for the amplitude envelopes which is useful to capture the idea that harmonics of the same instrument will decay at a similar rate:

$$d_a(i,j) = \frac{1}{t_2 - t_1 + 1} \sum_{t=t_1}^{t_2} \left( \frac{a_i(t)}{a_i} - \frac{a_j(t)}{a_j} \right)^2 \tag{2.2}$$

Harmonics are supposed to be integer multiples of a common base frequency, so if trajectories $i$ and $j$ are the $a$th and $b$th harmonics of a sound respectively, we would expect $\frac{f_i}{f_j} = \frac{a}{b}$. A distance measure for harmonic concordance between two trajectories can be given by how close their ratio of frequencies is to a simple rational. Assuming that the base tone is identified, $a$ and $b$ can take the ranges from $1, 2, \ldots, \lceil \frac{f_i}{f_{\min}} \rceil$ and $1, 2, \ldots, \lceil \frac{f_j}{f_{\min}} \rceil$ respectively, where $f_{\min}$ is the lowest frequency of any trajectory. The harmonic distance $d_h$ is then given as:

$$d_h(i,j) = \min_{a,b} \left| \log \left( \frac{f_i / f_j}{a/b} \right) \right| \tag{2.3}$$

Virtanen and Klapuri then took a weighted sum $d_{\text{all}}$ of each of these distances and the following was minimised to obtain the sets $S_1$ and $S_2$ of trajectories for the two sounds in the input:

$$(S_1, S_2) = \arg\min_{S_1,S_2} \frac{1}{|S_1|} \sum_{i,j \in S_1} d_{\text{all}}(i,j) + \frac{1}{|S_2|} \sum_{i,j \in S_2} d_{\text{all}}(i,j) \tag{2.4}$$

We can note that, if trajectories $i$ and $j$ are similar, for any trajectory $k$ the distances between $i$ and $k$ should be similar to those between $j$ and $k$. This allows us to describe each trajectory by a high-dimensional feature vector consisting of the distances between it and each other trajectory using each metric. We can then use standard clustering methods to identify the clusters of trajectories for each of the reconstructed sounds. This still requires us to weight each of the distance functions to ensure that the clustering does not ignore some and rely too heavily on others.

Sinusoidal trajectories are some of the smallest possible sound elements that we could take. One could argue that operating on larger structures could save us from some of the grouping work since that will be taken care of by finding these structures. The other option that I have considered is spectrogram factor components of the power spectrogram. Given the spectrogram $X$, Non-negative Matrix Factorisation gives matrices $A$ and $S$ such that:

$$X \approx AS \tag{2.5}$$

In the context of audio, $S$ is the source matrix where each row describes the spread of power across the frequency spectrum and $A$ is the mixing matrix where each column describes how the amplitude of the corresponding source row varies over time. Each product of a mixing column and its source row (herein referred to as a "spectrogram factor component") is independent of the others and so can usefully model an element of the sound.

The goal of NMF is to produce the $A$ and $S$ to minimise some error function describing how similar $AS$ is to $X$. Lee and Seung [15] gave a multiplicative-update gradient descent method which minimises the Euclidean distance $|X - AS|^2$ by performing

$$A_{ti} \leftarrow A_{ti} \frac{\left(XS^T\right)_{ti}}{\left(ASS^T\right)_{ti}} \tag{2.6}$$

and

$$S_{if} \leftarrow S_{if} \frac{\left(A^T X\right)_{if}}{\left(A^T AS\right)_{if}} \tag{2.7}$$

iteratively from some randomly initialised $A$ and $S$ matrices until they converge.

These factors exploit the idea that most musical instruments tend to have a consistent timbre, so most of the variation of the spectrum over time is fairly consistent over frequencies. This does not utilise the harmonic assumption and some sounds do not have a constant frequency spectrum (consider vibrato or transient frequencies); nevertheless they act as good elements and complex sounds can generally be described well by a combination of a few of these factors. I have chosen to investigate the use of these in comparison to sinusoidal trajectories.

These elements are all taken by considering structures in the Short-Time Fourier Transform. Due to the usefulness of both time and frequency information, the intuition would be that a wavelet transform would be more appropriate. However, the need for the transform to be invertible (in order to reconstruct the audio) and the need for a very good

frequency resolution (we would need a very large number of voices per octave to identify harmonics at high frequencies) would require using a complete continuous wavelet transform which, for sound inputs, results in a very large transform which is computationally expensive to process. For this reason, I have chosen to not consider wavelet methods in this project.

## 2.3 Choice of Implementation

At the heart of it, the solution to this problem is one of signal processing. The availability of many transforms and support for linear algebra makes MATLAB an obvious option for the language to develop in. This is a common choice for programming in this field due to the large number of signal processing toolboxes and the ease at which one can quickly create scripts to test new ideas.

However, the element-based flow would see benefits from being able to capture and handle the elements as objects. This can easily be seen in the case of the sinusoidal trajectories, since each of these corresponds to a combination of amplitude, frequency and stereo envelopes with varying lengths between different trajectories based on the onset and offset times. Having an object-oriented approach would allow us to group these together well with the behaviour for the distance metrics between trajectories.

Whilst MATLAB offers some support for classes and objects, I have opted to use C++ instead for its more natural class structure. Since the solution can be described in a very modular fashion, it helps to have good support for functions and being able to group them into classes for each application (e.g. clustering algorithms, transforms, etc.). The strong modular structure of C++ also makes it very extensible which would be useful if I were to continue work on this area or similar forms of sound processing after this project since many of the required components of the solution have very general applications. Nevertheless, I chose to use MATLAB for some quick initial experimentation to check the feasibility of my solution.

For reading and writing audio data from files, I have chosen to use the JUCE framework [3]. Whilst it is designed to aid development of synthesisers and audio effect plug-ins for music production, it offers an easy way of interacting with `wav` files which is sufficient for the needs of this project. The use of matrices in the proposed solution, especially when obtaining sound elements by NMF, calls for the use of a library providing good support for linear algebra. I have chosen to use Eigen for this since it provides a natural syntax and is more than fast and flexible enough for the project's needs.

## 2.4 Software Engineering Approach

### 2.4.1 Requirements

This project requires that I build a working solution to the problem of uninformed harmonic sound separation. In order to evaluate this, it is necessary to produce a set of audio files containing a range of natural harmonic sounds covering a range of expected inputs.

Each aspect of the solution must then be evaluated against these tests and compared to some potential alternatives.

The resulting solution should be able to accept an integer value $k$ and a single `wav` audio file containing $k$ sounds and yield $k$ `wav` files which audibly resemble the original sounds before mixing. Since the target application of the solution is for musical usage, the group delay (difference between the delays on the amplitude envelopes of each frequency) must remain within the human threshold of perceptibility which is as low as 1ms [7]. One of the easiest ways to guarantee this is by preserving the relative phase information between frequencies in the audio.

### 2.4.2 Software Development Process/Methodology

The original plan for this project was to follow a Waterfall strategy with the belief that the main comparison (sinusoidal trajectories against spectrogram factor components for the sound elements) would require the design of two separate solutions which would operate without significant overlap and could be completed with a single iteration of development. After investigating the problem and some initial scoping of the difficulty of the task in MATLAB, I intended to complete each solution separately, then investigate the internal parameters which weight each of the individual distance measures and finally evaluate each of them.

In the initial stages of the project, it became apparent that the two solutions originally planned had a substantial overlap in how they work. During the implementation of these, I spotted several possibilities for further investigation with regards to adding or modifying other parts of the solution. These were investigated in turn before optimising the distance weights, giving a more incremental paradigm.

### 2.4.3 Version Control and Back-up

A Git repository was made for all files related to the project, including source code (without libraries), test audio inputs and outputs and written reports. This repository was synced up to GitHub after each substantial change to the source code or reports. The local repository was regularly copied up to a cloud storage facility and a copy of the source code (with all libraries) was maintained on an external hard drive.

### 2.4.4 Testing

The Philharmonia Orchestra provide a number of sound samples [4] of orchestral instruments under a Creative Commons Licence. These provide a good representation of typical musical sounds which is desirable when building a test set. For the investigation over the weighting parameters, a validation set was formed by taking a small set of these samples and producing every pair combination with some stereo separation. For the final evaluation, a completely distinct set of tests was formed by mixing other samples with random stereo separation. I chose to use the standard measures of Signal to Distortion, Interference and Artefact Ratios for Blind Audio Source Separation (described in detail in the Evaluation section) to give some numerical evidence of the solution's performance

in terms of the accuracy of the separation (how well the sound elements were associated to the correct sound) and the sound quality of the reconstructions.

# Chapter 3

# Implementation

The solution is structured as a collection of static methods, each providing functionality for one stage of the general algorithm. A command-line tool has been produced to apply the solution to a given file. The usage of this is `wvs22Separate <filepath> k <flags>` where `<filepath>` gives the complete filepath of the desired input (this must be in the `wav` file format). The output gives $k$-many files at the same location as `<filepath>` but with `_i` appended to the file name ($i \in \{0, 1, \ldots, k-1\}$). The following flags may be set:

| Flag | Description |
|---|---|
| -sin | Use sinusoidal modelling to extract sound elements (default) |
| -nmf | Use NMF to extract sound elements |
| -hc | Use hard clustering by k-means (default) |
| -sc | Use soft clustering by k-means |
| -mc | Use soft clustering by NMF |
| -nc | Use naive clustering |
| -r | Reversible - add remainder of spectrogram to ensure that the sum of the outputs gives the input |
| -v | Verbose |

This chapter will look at the implementation of each section of the algorithm in turn and discuss any relevant theory or issues encountered as appropriate.

## 3.1 Signal Transforms

The Short-Time Fourier Transform takes the Discrete Fourier Transform over a number of small, overlapping time windows to give a localised view of the frequency spectrum. This is often a standard transform in signal processing libraries, but the need to provide a custom implementation here was motivated by the lack of libraries for C++ offering an inverse STFT function.

Calculating the STFT requires use of a DFT function. The obvious choice in C++ is to use the FFTW library [2], popular for its speed. Since this project was not speed-critical, this was of little importance and so I opted to implement the FFT and its inverse myself to operator directly on the Eigen data structures.

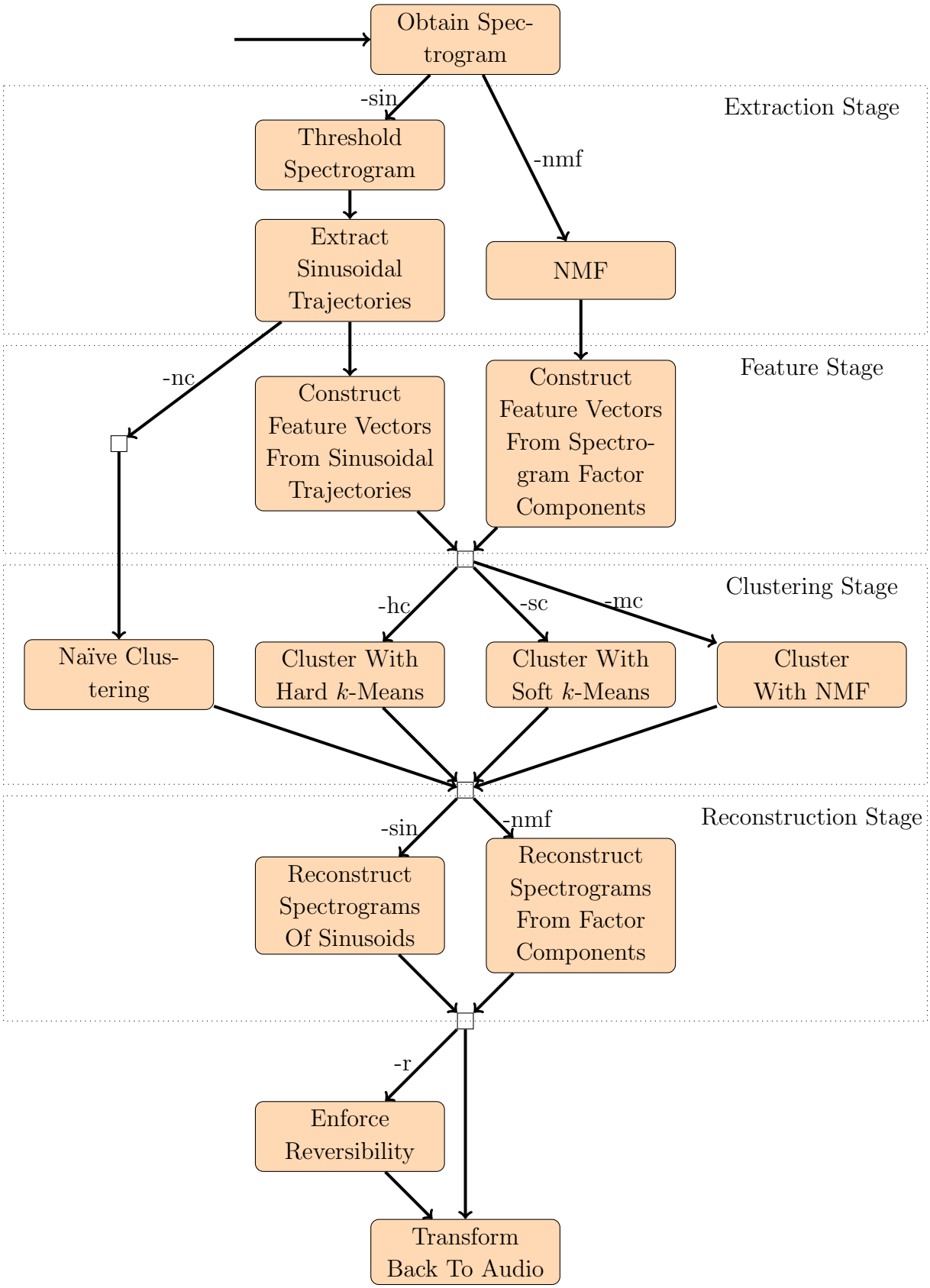


Figure 3.1: Schematic of the overall separation process. Transition annotations describe the flags of the command-line tool required to force execution down that path.

Since these are designed to be used on audio data, the samples were all real-valued, allowing for some well-known optimisations to be made due to the hermitian symmetry in the DFT. For instance, the DFT can be fully described by its first $\lceil \frac{n+1}{2} \rceil$ elements and generated by packing it into a half-sized complex DFT [22]. By discarding the second half (the "negative" frequencies), we can not only decrease the memory consumption and speed of sound element extraction, but also the resulting STFT is more intuitive to work with since two frequencies are in harmonic concordance iff their bin numbers are a simple rational fraction (considering a sinusoid with 23 cycles in a 120 sample window, the negative frequency would have 97 cycles; but despite them being trivially in harmony, their frequency ratio is $\frac{97}{23}$). Also, the inverse DFT can be calculated by taking the conjugate of the input, applying the forwards DFT and then scaling and taking the conjugate of the result. Since we only need real outputs, we can skip this last conjugate step. Equally, when calculating the inverse STFT, we can simply zero-pad the half-DFT and calculate the inverse DFT on that rather than reconstructing the conjugate symmetry since it will not affect the real portion of the result. It could be argued that a transform designed for real-valued data, such as the Hartley transform [13], would be more appropriate, but the computational benefits are marginal considering the aforementioned optimisations we can apply to the DFT.

## 3.2 Sinusoidal Trajectories

For each of the choices of sound elements, I had to consider their extraction, internal representation, conversion to feature vectors for clustering and reconstruction as audio. The first such element model was a collection of Sinusoidal Trajectories, defined as pure sinusoids with time-varying frequency and amplitude. These can each be represented as a vector of points describing the amplitude and frequency in a given time frame, though I added a stereo envelope to capture the positional information available from multiple input channels. These were normalised to aid the calculation of $d_f$ and $d_a$ (see equations 2.1 and 2.2) and reduced to only the time for which the sinusoid is present, massively cutting down the space consumption of storing them.

### 3.2.1 Extraction

In each window of the spectrogram, any sinusoids present will create local maxima in magnitude at their frequencies. The extraction stage operates by iterating through each time frame and identifying all significant peaks, then connecting these up between frames where possible by looking for existing trajectories which could be extended (the difference between the frequencies of the peak in the current frame and the trajectory in the previous frame is proportionally small); where no such trajectory exists, a new one is created starting with the peak. This takes $O(N^2T)$ time to run over an input with $N$ identifiable sinusoids and duration $T$, making it one of the higher-complexity tasks in the solution.

Any noise will have random variations in magnitude over the frequency spectrum which will generate a large number of peaks and consequently many irrelevant elements. This is reduced by thresholding the low values in the spectrogram to zero, at the cost of

removing some of the more subtle harmonics. The effects of varying this threshold are discussed in the Evaluation chapter.

Identifying the peak frequency bin will quantise the frequency of the sinusoids, but this can mean that motion in the frequency envelope can be lost for sinusoids of low frequency, or emphasized if it oscillates around a boundary. This will clearly bias the frequency distance measure, making it not pitch invariant. To counteract this, the solution interpolates between the frequency bins by fitting a quadratic through the peak bin and its two neighbours and finding the turning point to estimate the true peak frequency.

Similarly, the quantisation over frequencies can affect the amplitude of the peak due to the scalloping loss. If the frequency of a sinusoid aligns with one of the DFT bins, its power is placed entirely in that bin. When it does not align with any bin, spectral leakage occurs as the power is spread over the bins near the sinusoid's frequency. As the total power is constant, this means that the peak bin amplitude will be decreased. This loss in the peak value will increase as the true frequency moves further away from the nearest bin. This variation in the obtained values means that the amplitudes (and consequently the amplitude distance measure) are not frequency invariant. I have chosen to decrease the variability in this by considering the power spectrogram (squared magnitude of the STFT) and take the sum of the power in the nearest two bins to the peak frequency as the power of the trajectory. Other solutions to this problem exist such as using a Flat-Top window function instead of a Hamming window when computing the STFT, since this is especially designed to reduce the effects of scalloping.

### 3.2.2 Features

In addition to the distances used by Virtanen and Klapuri (see equations 2.1, 2.2, and 2.3), I added a few more metrics to make better use of the information we have available. Let the value of each trajectory's stereo envelope $s$ at a given time to be the proportion of the energy passing through the left channel. We can then define a spatial distance $d_s$ as:

$$d_s(i,j) = \frac{1}{t_2 - t_1 + 1} \sum_{t=t_1}^{t_2} \left(s_i(t) - s_j(t)\right)^2 \tag{3.1}$$

There may not always exist a time interval at which a pair of trajectories are both present. In these cases, the values of $d_f$, $d_a$ and $d_s$ are undefined. I have introduced constant miss penalties to account for this case to indicate that the non-overlapping trajectories are unlikely to be part of the same sound.

The solution from [26] uses the approximate onset times of the trajectories to give an initial estimate of the grouping which is then optimised considering the distance functions. After this initial grouping, the onset times are not used, but they often act as a very good identifier that two trajectories are from the same/different sounds. In the solution, I used a simple distance measurement of the absolute difference between the onset times.

$$d_o(i,j) = |on_i - on_j| \tag{3.2}$$

The implementations of the various distance functions are generally clear from the equations. When calculating the harmonic distance (equation 2.3), it can be noted that not all pairs of $a$ and $b$ values must be tested since, for a given $a$, there is a single minimum in $\left|\log\left(\frac{f_i/f_j}{a/b}\right)\right|$ as we vary $b$, occurring when $b$ is either $\lfloor\frac{af_j}{f_i}\rfloor$ or $\lceil\frac{af_j}{f_i}\rceil$. This gives us only two values of $b$ to test for each value of $a$.

The final feature vectors were of the form:

$$v_i = (D(i,0), D(i,1), \ldots, D(i,N-1)) \tag{3.3}$$

where

$$D(i,j) = (d_f(i,j), d_a(i,j), d_h(i,j), d_s(i,j), d_o(i,j)) \tag{3.4}$$

Calculating these feature vectors also takes $O(N^2T)$ time.

### 3.2.3 Reconstruction

The trajectories model a single line in the spectrogram. Rather than producing a sine wave modulated by the various envelopes, I decided to copy the STFT data from the input over the line described by the trajectory as this would preserve the phase of the inputs and allow the audio generation to be done in a single batch. This is done for a small range around the peak frequency to account for some of the spectral leakage. These can also be weighted by the probability of membership to the output obtained through soft clustering.

## 3.3 Spectrogram Factor Components

### 3.3.1 Matrix Factorisation

Whilst the NMF procedure described by Equations 2.6 and 2.7 will converge for any random initial $A$ and $S$, my implementation starts by taking a random selection of rows from $X$ as initial source rows in $S$ and some columns of $X$ as initial mixing columns in $A$. If the initial mixing columns line up with some of the sinusoids in the audio, it will immediately extract their amplitude envelope and other sinusoids with similar envelopes can quickly be added in to this factor component. If the initial source rows are taken from moments where only one sound exists, it can immediately extract that sound's spectrum as a factor component, eliminating it from the rest of the input. With these heuristics, the solution is able to converge very rapidly.

When setting up the NMF problem, we can specify the desired size of $A$ and $S$ (i.e. the number of factor components we want). In the simple case, we would aim for $k$ components when the solution is told that there are $k$ sounds in the input. Since these components assume that the frequency spectrum of the sound is constant with respect to time apart from global amplitude changes, they may not accurately capture more complex sounds. In practice, we can get a good representation using a linear combination of a few components, so when setting up the NMF problem we will ask for more than $k$

components and then cluster these. In my implementation, the number of matrix factor components requested is left as a modifiable parameter.

The test for convergence used was a comparison between the squared Euclidean distance errors after the previous and current iterations. If the improvement in the error drops below the threshold parameter, the process is stopped.

### 3.3.2 Features

A useful effect of the factorisation is that the source rows and mixing columns immediately give us a way of judging the similarity between two factor components, since those from the same sound are likely to have similar amplitude envelopes or spectra. We can hence construct feature vectors for clustering by concatenating the source rows and mixing columns.

### 3.3.3 Reconstruction

Reconstructing the audio from a combination of spectrogram factor components is simple since we can sum them, square root, and take the inverse STFT. In my implementation, I am calculating the NMF over the combined power spectrum of the input channels, so I am weighting each value in the final STFT to match the stereo spread of power in the input and copying the phase of the inputs. Performing the reconstructions when using either model of sound elements requires $O(KNT)$ time for $K$ outputs, $N$ elements, and input duration $T$.

## 3.4 Clustering Algorithms

The clustering stage of the overall algorithm is intended to identify which of the sound elements were from the same source using some of the element properties. Detailed investigations of specific clustering algorithms are beyond the scope of this project, but it is still of interest to compare the use of a few different types of clustering. The main comparison to be made here is between hard and soft clustering.

### 3.4.1 Hard Clustering

Hard clustering assigns each of the feature vectors to exactly one of the clusters - in the context of sound separation, it assumes each sound element came from a single sound source and so is mapped to that source only, giving us completely distinct and independent outputs. Lloyd's algorithm for $k$-means clustering [16] is a popular choice and the one that I implemented here. Hard clustering is typically most useful when the feature vectors of data from different classes are easily separable.

### 3.4.2 Soft Clustering

When the data is less separable, it is often useful to use soft clustering which assigns a probability of each data point being in each cluster. The standard example of soft $k$-means clustering, and what is used in my solution, is an adaptation of Lloyd's algorithm [17] which assigns points $\mathbf{x}_i$ to clusters with centroids $\mu_j$ according to the softmax function

$$z_{ij} = \frac{e^{-\beta|\mathbf{x}_i - \mu_j|^2}}{\sum_l e^{-\beta|\mathbf{x}_i - \mu_l|^2}} \tag{3.5}$$

for some stiffness parameter $\beta$, then calculates the centroid positions as means of the points weighted by the $z_{ij}$ values. The intuition behind using a soft clustering method here is to imitate how humans tend to perform sound separation, since we do not fully eliminate the other sounds, but the sound of focus is clearly emphasised. Using this could also make our solution better at coping with misclassifications of key sound components since the result will still contain them, albeit somewhat quieter since they have been given a lower weight.

### 3.4.3 Clustering by NMF

Interestingly, Non-negative Matrix Factorisation can also be used as a soft clustering technique [27], viewing the source matrix as a description of the features linked to each cluster and the mix matrix as a description of how well each data point fits each cluster. Since I had already produced an implementation of NMF for this project, it was little effort adding it in here as a clustering option.

### 3.4.4 A Naïve Solution

A naïve algorithm for harmonic sound separation might attempt to identify the base frequencies of the individual sounds and then look for the corresponding overtones and extract them. The "naïve" clustering option that I have added to the solution is not a general clustering algorithm but simply mimics this naïve algorithm and hence can only be used with sinusoidal trajectories. It starts by taking the unlabelled trajectory with the highest mean amplitude as the seed for a sound. Any trajectory with sufficient harmonic concordance (the harmonic distance from the seed is below some threshold parameter) is then labelled to indicate that it belongs to that sound. This is repeated until we have the desired number of sounds or we have run out of sinusoids. Any remaining sinusoids are then assigned to the sound which minimises the maximum harmonic distance between it and any sinusoid already labelled under that sound.

### 3.4.5 Discussion

Besides the naïve option, all of these clustering algorithms require an element of randomness when initialising the cluster centroids/matrix factors. A poor choice here can result in the algorithms only finding local optima, rather than the global optimum. In early tests, I noticed that some random seeds caused the results to be almost randomly mixed, with all output sounds sounding like a filtered mixture of all of the original sounds and when using other seeds it successfully separated them. This was especially the case with the soft clustering methods, where many of the sound elements were split evenly amongst the outputs. In this sense, the better results were those where the soft clustering gave "harder" cluster assignments (intuitively, this suggests that it identified some good distinguishing features and is more confident about its assignments). In order to have a better chance of getting a good assignment, the solution repeats the soft clustering steps multiple times with different initialisations and selects the results which maximises the

sum of squares of the assignment probabilities. I was not able to find a heuristic for a good hard clustering.

The clustering stage is typically where the solution spends most of the processing time, with a cost of $O(KN^2)$ (with sinusoids) or $O(KNT)$ (with spectrogram factor components) for each iteration with hard, soft or NMF clustering (of which the number of iterations may not be bounded so nicely) or $O(KN + N^2)$ with the naïve method.

## 3.5 Other Stages

### 3.5.1 File I/O

For file handling, JUCE provided a very simple interface for reading and writing with `.wav` files via `int` buffers. The only thing left to do was to convert between the fixed-point `int` representation used in the file and the floating-point `double` representation used in the rest of the solution.

### 3.5.2 Enforcing Reversibility

For any separation process, reversibility is a good property to have; that is, recombining the outputs gives something approximately equal to the input. For both of the sound element models being used in this project, they may not fully cover the STFT of the input audio, whether by missing some part of the sounds or by correctly eliminating some noise. I added the ability for the solution to be reversible by subtracting the output STFTs from the input and then splitting this remainder evenly across the outputs. It is conceivable that some applications may find this property undesirable and prefer that we guarantee everything in a given output is from the same sound source, so I have left this as an optional part of the process.

## 3.6 Parameter Choices

At many points in the solution's code, key constants were kept as static parameters for the usual reasons of ease of modification and readability. For many of them, changing their value can significantly affect the time taken for the separation program to run or the quality of the outputs, so it is important to find suitable values for them.

When calculating the STFT, we need to consider the window size, number of frequency bins and the hop size (the number of samples by which the window shifts along with each hop). These directly affect the temporal and frequency resolution of the analysis. Increasing the window size will improve frequency resolution at the expense of some temporal resolution since we will be reducing the amount of spectral leakage (the bandwidth of the window function decreases) as we are integrating over a longer period of time. Increasing the number of frequency bins via zero padding and decreasing the hop size can improve the frequency and temporal resolutions respectively but will give a larger matrix as a result, making it slower to process. A sensible choice of values would be to derive them from the resolution of the human auditory system. Some studies have shown that humans integrate sound over periods of 200-300ms [11] (corresponding to between 8000 and 13000 samples at a 44.1kHz sampling frequency) but we can also detect gaps in sounds of as low as 3ms

[21] (132 samples, though this varies with frequency up to about 22ms or 970 samples at lower frequencies, where the fundamental frequencies of common musical notes lie). This has driven a choice of 1024 samples for the hop size to capture the gap detection and a window size of 8192 samples. Because of the interpolation between frequency bins for identifying the frequency of sinusoids, we do not need a tremendous frequency resolution from the STFT so I chose to not use any zero padding in order to keep the matrix size manageable, giving 8192 frequency bins (though this is effectively halved when we ignore the negative frequencies). Fixing these parameters makes computing the DFT of each window a constant cost, reducing the overall time complexity of the STFT computation to $O(T)$ (for input duration $T$).

The aim of the thresholding of the STFT prior to sinusoid extraction is to control the number of elements used in order to make clustering run in a sensible time, especially since the time complexities of several stages are quadratic in this respect. The threshold was varied throughout initial testing, then I set it to a value that I found to give around 20 sinusoids from each of the individual sounds. Similarly, the number of spectrogram factor components obtained via NMF was set to 10 (regardless of the input $k$), though the effects of changing these two parameters are discussed in the evaluation chapter.

The weights given to each of the distance measurements between sinusoidal trajectories when obtaining their feature vectors for clustering and the stiffness parameter of the soft clustering technique were set in order to maximise the average SDR on a validation data set. This was performed by taking five of the instrument samples (all of which were of different instruments being played at different notes to consider the best case performance) and combining each pair of them together with some stereo separation (varying between zero and absolute separation). Each parameter was varied, holding all others constant, until the optimum was identified and this process repeated until they all converged.

## 3.7 Software Engineering Practice

As mentioned in the Preparation chapter, I had originally planned to follow a Waterfall strategy, implementing two distinct solutions but because they were so similar and shared a significant portion of the process it became a more incremental development. After building the full solution through with the sinusoidal trajectories and hard clustering, it was then extended, generalising the code where needed, to add the other options as I spotted the opportunity for investigation. Retrospectively, it would have been a smoother development process if all investigated comparisons had been identified at the start of the project and the code kept as general as possible to make it easier to extend and try alternative methods for each section of the process.

When writing the code, using Eigen for the matrix and vector operations helped greatly since the aspects of the process that were well suited to the original MATLAB code could easily be mapped across as it maintains a MATLAB-like syntax and behaviour.

Even with what turned out to be a relatively small code base, providing documentation during the writing of the code was extremely useful for navigation and keeping check of the assumptions made about parameters and inputs. Full method descriptions and

assumptions on the argument formats were given in the header files and brief inline comments in the code were useful to section longer procedures. Since one of the features of C++ driving me to use it for this project was the scope for modularity and ease of code reuse, I made sure to separate out highly reusable methods (e.g. the signal transforms, NMF and general clustering algorithms) from the less reusable code (e.g. the sound-separation specific code).

Time management is always a crucial performance measure on a software engineering project. In my project proposal, I outlined eight work segments relating to the development of the solution and a further five for the dissertation. Due to inconsistent workloads outside of this project, some delays were experienced with regards to meeting the deadlines set out for the development segments. In particular, the initial MATLAB investigation on using sinusoidal trajectories was not completed until four weeks after my proposed deadline, having the knock-on effect of the following two segments being finished two weeks late as well. However, the latter half of the development was completed without major time setbacks, meaning that the goals of the project did not have to be compromised due to deadline constraints and the production of the solution was completed on time.

# Chapter 4

# Evaluation

For any software product it is vital to assess its performance in terms of resources used (time, memory or energy), its quality of output and its usability in order to gauge how useful and potentially marketable it will be. Since this project is aimed at comparing a selection of interchangeable process components, this section will focus using numerical tests to accomplish this comparison.

## 4.1 Method Comparisons

No standard benchmark data set exists for harmonic sound separation, but there is a set of standard metrics [23] which can be suited to any test data. Each output can be described as

$$\hat{s}_i = s_i + e_{\text{interf}} + e_{\text{noise}} + e_{\text{artef}} \tag{4.1}$$

where $s_i$ is the signal from the target source and $e_{\text{interf}}$, $e_{\text{noise}}$ and $e_{\text{artef}}$ are the error signals corresponding to interference (contributions from the other sounds), noise (assuming we know what the added noise signal was), and artefacts respectively. After decomposing the output into these terms, we can then compute the following metrics:

$$\text{SDR} = 10 \log_{10} \frac{|s_i|^2}{|e_{\text{interf}} + e_{\text{noise}} + e_{\text{artef}}|^2} \tag{4.2}$$

$$\text{SIR} = 10 \log_{10} \frac{|s_i|^2}{|e_{\text{interf}}|^2} \tag{4.3}$$

$$\text{SNR} = 10 \log_{10} \frac{|s_i + e_{\text{interf}}|^2}{|e_{\text{noise}}|^2} \tag{4.4}$$

$$\text{SAR} = 10 \log_{10} \frac{|s_i + e_{\text{interf}} + e_{\text{noise}}|^2}{|e_{\text{artef}}|^2} \tag{4.5}$$

In most cases, we will not be considering the addition of noise, so we can assume that $e_{\texttt{noise}}$ is zero and we will not look at the Signal to Noise Ratio. The remaining metrics each describe the performance of the system in a different way. The Signal to

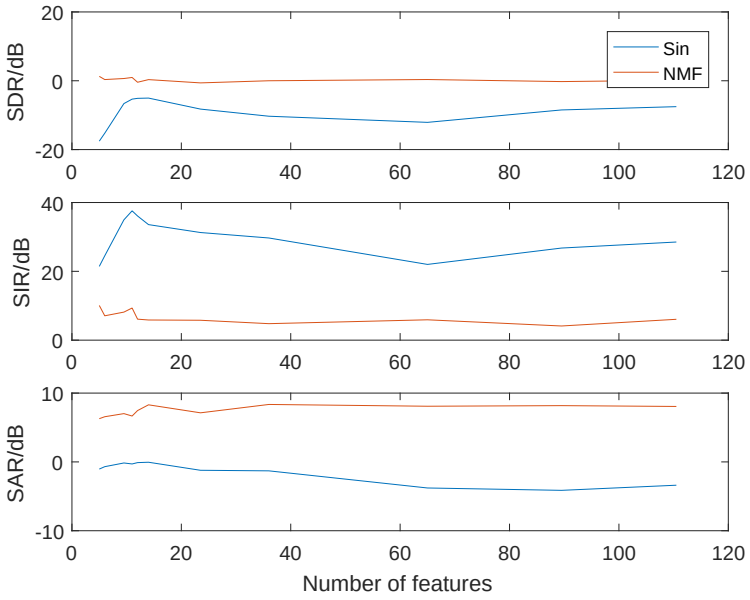


Figure 4.1: Separation performance against the number of elements extracted per sound. For the spectrogram factor components, the same number of elements were extracted on every input from the set, whereas with sinusoidal trajectories the threshold was constant, giving a range of actual sinusoid quantities so the average was taken.

Interference Ratio indicates how well the sound elements were clustered and so can be viewed as the precision of the separation. The Signal to Artefact Ratio indicates how well the outputs actually fit the inputs, consequently measuring the quality with which the sound elements can actually capture the full sounds. The Signal to Distortion Ratio is somewhere between these and gives the overall quality of the outputs.

To give some context on the values given for each of these statistics, compact discs are capable of storing audio data with a SNR of up to 90dB [10]. Humans have been found to identify words in spoken sentences with 50% accuracy in the presence of other conversations with SNRs as low as 2.4dB [14] and obtain some of the speech cues at $-12$dB SNR in white noise [18]. Since we have evolved to detect speech very well, it would be understandable for us to have a lower sensitivity for other sounds such as the musical instruments used here.

Unless otherwise stated, the tests done for producing the graphs and statistics were obtained using a test set comprising of 40 mixtures, each with two sounds from a selection of 20. These samples were evenly distributed between 5 different instruments and were selected to give a good spread over the musical notes. The pairings in the test set were chosen at random such that each sound was used equally often.

### 4.1.1 Sound Element Choice

Both sinusoidal trajectories and spectrogram factor components have their theoretical benefits when it comes to separation. In practice, using the factor components gives better SDR and SAR values per element extracted which is as expected since each one is

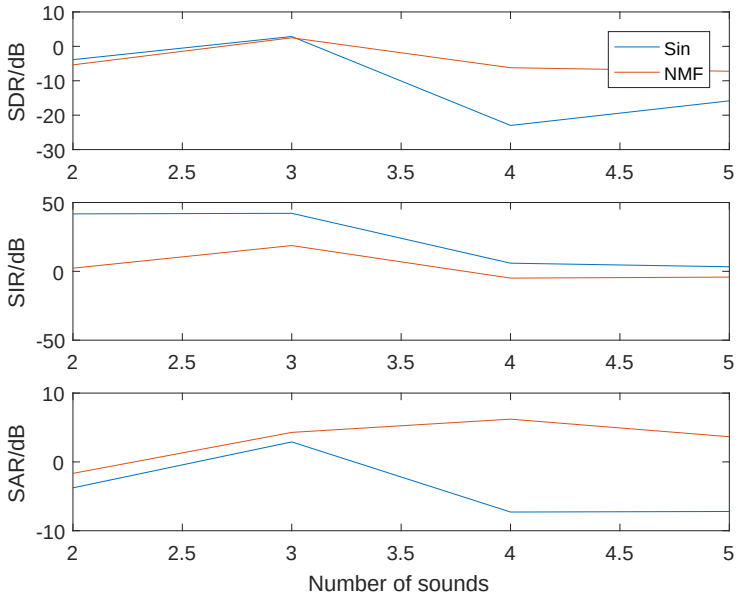


Figure 4.2: Separation performance against the number of sounds in the mix. These values were taken over a single example where a complex auditory scene was gradually built up (so the sounds used in each test were also present in all later tests).

able to potentially capture a significant proportion of the information and it could take many trajectories to represent a single spectrogram factor component to within some error margin. However, the finer control over regions of the spectrogram and the guarantee that each feature originates from a single source (except for the case of colliding harmonics which are much harder to separate in either case) given by the sinusoidal trajectories allow them to generally give SIR values several orders of magnitude larger.

A surprising result is that, with both options, increasing the number of elements extracted did not always improve the quality of reconstruction or separation. For example, having many "broken" trajectories (where the amplitude varies around the threshold and so may be cut into a sequence of short trajectories) will greatly increase the number of features obtained but most of them will not relate well to one another due to not overlapping for very long in time. This can push the separation towards removing these small artefacts from the mix rather than removing one of the sounds. Also, after obtaining the peaks of the actual harmonics, the next most significant spectral regions are likely to be from spectral leakage around the actual harmonics. These fake sinusoids are unlikely to be clustered with the real sounds since they will not have great harmonic concordance. This latter case is an issue with spectrogram factor components as well, since they will eventually start to not actually resemble either of the individual sounds.

When separating sounds from a more complex auditory scene, as in the tests used for Figure 4.2, the general trend is that the SIR decreases as the number of sounds increases since there is a much greater likelihood of colliding harmonics (where harmonics from multiple sounds played at the same time occur at the same frequency). This is reasonable from a human perspective as we will generally struggle in the scenario of the cocktail party problem as the number of simultaneous nearby speakers increases and it would often be a

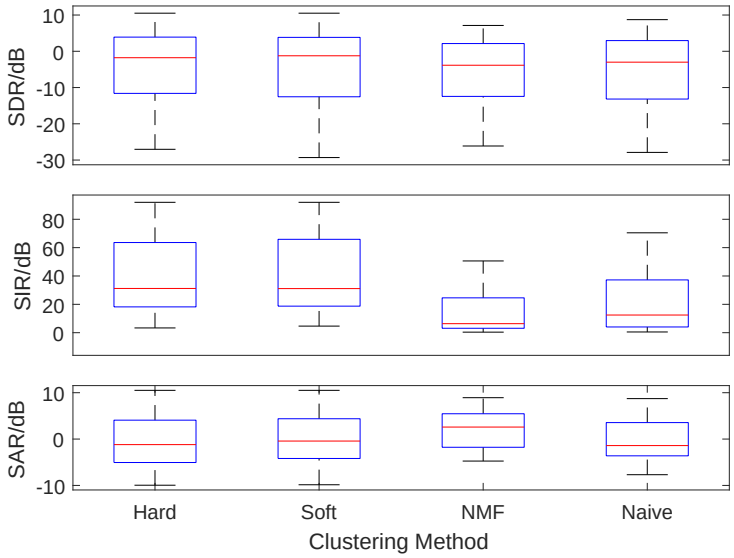


Figure 4.3: Separation performance against clustering method when using sinusoidal trajectories as the sound elements. These were obtained with only a single test for the soft and NMF clustering options rather than multiple times and taking the "hardest".

considered a very difficult task to identify the notes played by the third trumpet in a full orchestra. However, the tests do not show this consistently since it is difficult to ensure that all sounds in a scene are equally easy to separate from the rest, resulting in the late addition of an easily separable sound improving the average quality.

Listening to a selection of the outputs from these tests, those obtained by extracting sinusoids often sounded heavily low-pass filtered since the higher frequencies were typically discarded first. This effect was often so extreme that the results appeared somewhat synthetic as opposed to the natural sounds of a physical instrument. This was typically not the case for the results from applying NMF to the spectrogram as the full spectrum was considered. In a lot of cases, the instruments were recognisable but the bleeding of one sound into the other's output is often noticeable. This sometimes came in the form of the interfering sound picking up the target sound's amplitude envelope, even if it is not a natural envelope for that instrument, creating an unusual experience.

### 4.1.2 Clustering Method

The expected differences between using the hard and soft $k$-means clustering methods were that soft clustering would improve the SAR because the reconstructions would be less distorted (misclassified elements from the target sound will still be present) at the expense of the SIR (more elements from other sounds will be partially present). This was evident in the results when using spectrogram factor components. However, I found that there was little difference in the overall quality of the results between using the hard and soft clustering with sinusoidal trajectories. This most likely arose because they were separately optimised for SDR which might have favoured harder assignments for sinusoidal

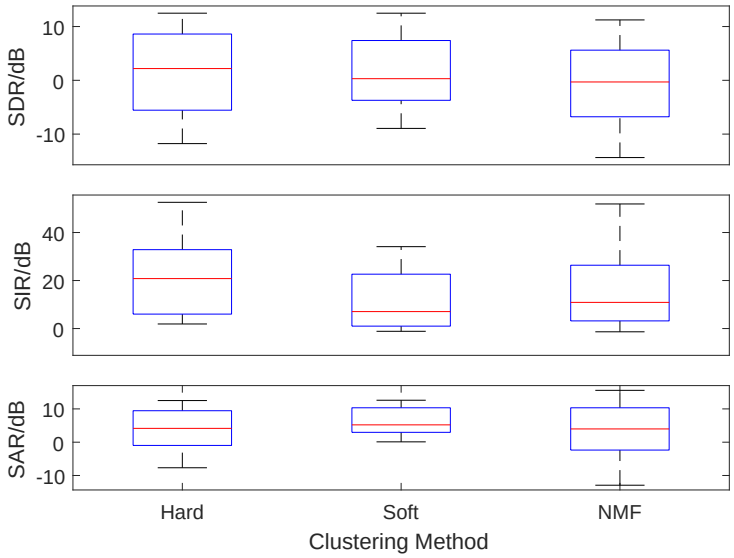


Figure 4.4: Separation performance against clustering method when using spectrogram factor components as the sound elements.

trajectories and softer assignments for spectrogram factor components. This may be because of their finer structure ensuring, where possible, that each element was only from a single sound which can naturally give rise to greater separability and consequently harder assignments.

Whilst the distributions of scores did make some general changes, there were a large number of the tests where the scores were virtually the same in both cases since the best soft assignment identified was negligibly different from the hard assignment. This is to be expected since the soft assignments should only be needed when there are colliding harmonics which is not expected to be the case for all possible inputs.

Theoretically, clustering by NMF should be equivalent to soft $k$-means but the results show it to act somewhat differently. Due to the lack of "stiffness" control when using NMF it would treat the problem the same for both sound element options. From the observed results, it would appear as though it is clustering softer than the optimal for sinusoids and harder than the optimal for spectrogram factor components.

The naïve clustering method was found to offer similar results to the hard and soft $k$-means but with a lower average SIR. Since it only makes use of the harmonic concordance between sinusoids whereas the other clustering options used a number of different dimensions of comparison, this result comes as no surprise. This method gave the largest difference out of all options with regards to how the outputs sounded, in many cases giving some of the clearest reconstructions of one of the instruments, even if the interference was slightly higher.

From the options presented in this investigation, the results would indicate that using spectrogram factor components with soft clustering should be preferred if the purity of the original sounds should be prioritised since it maximises the Signal to Artefact Ratio. On the other hand, the method that looks most promising for accuracy of the separation

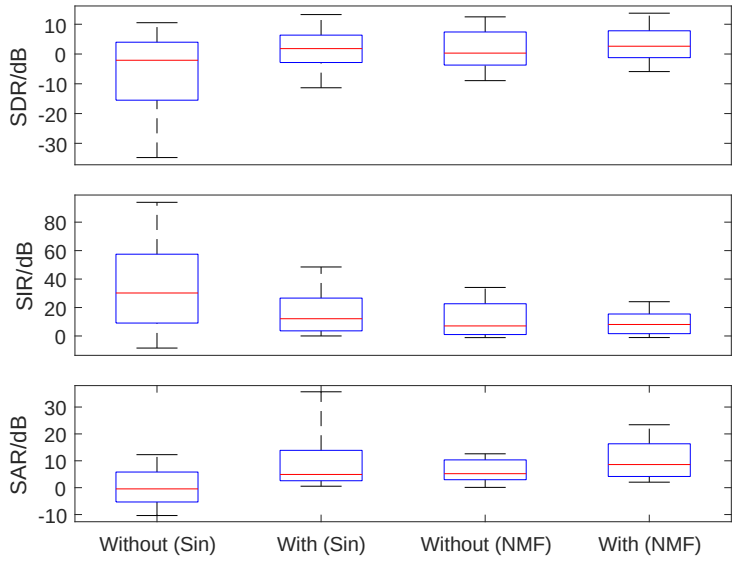


Figure 4.5: A comparison of the separation performance with and without the reversibility property enforced for each choice of sound element.

(i.e. maximises the SIR) is by taking the sinusoidal trajectories and separating them by soft clustering.

### 4.1.3 Reversibility

The desired effect of adding the reversibility property is to eliminate artefacts from the combined outputs (and hence some of the artefacts from each individual output). In achieving this, we sacrifice some of the quality of separation by allowing some of the remainder, which may be noise or interference from other sources, to be included in each output. Whilst this is also the intended result of using soft clustering, this can have more dramatic effects, especially in the case where very few sound elements are extracted.

Figure 4.5 shows the typical effects of enforcing a reversible separation. With both element options, we obtain a significant increase in SAR as both the artefacts in each output are reduced and the projections onto the target and interference error are increased as the original sounds are better captured. In fact, on this test set, the reduction of artefacts was much greater than the increase in interference, giving a higher SDR for some tests that had previously been handled poorly.

Predictably, the effect of using this feature was more noticeable in the sinusoidal modelling case. It substantially reduced the "filtered" effect, making the outputs sound much more natural and realistic, even if interference was inescapable.

## 4.2 Case Analysis

The remainder of this chapter discusses the performance of the solution against a number of properties of the input. These are primarily targeted at aspects of the solution when

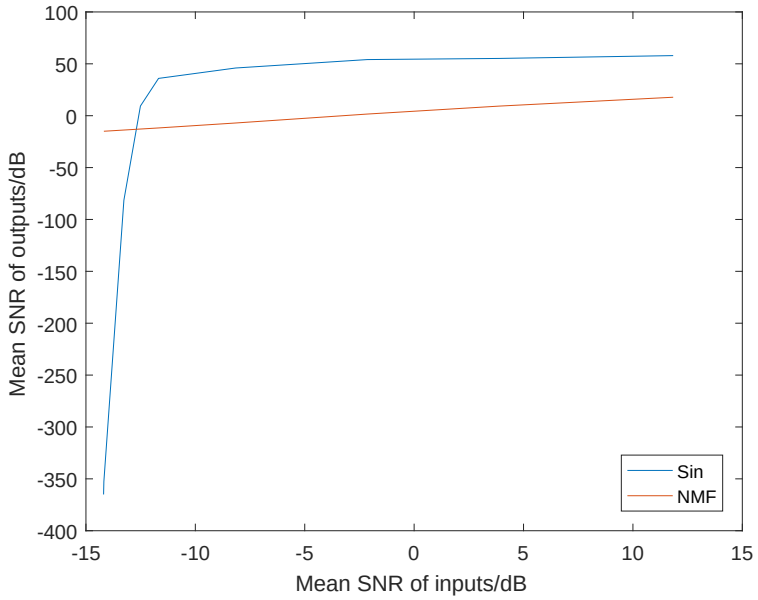


Figure 4.6: Noise reduction at different levels when using both sound element options. The standard test set was used but a sample of white noise is added. Since the amplitude of the individual instrument samples vary, the SNR varies over the inputs and so was averaged for each level of noise.

using sinusoidal trajectories so we will focus on results acquired with them and soft clustering without forcing reversibility.

### 4.2.1 Performance Against Noise

When modelling the sound using sinusoids, I am explicitly thresholding the STFT at a given level. When the noise sits below that level, none of it will be extracted as features and hence it will not appear in the outputs. However, as we decrease the SNR in the input, some of the noise will start to push over the threshold. This will not only add elements of the noise to the output, but it may cause worse clustering, adding more interference and reducing the level of the target sound. As the noise takes over, masking effects can be observed whereby the signal is not detected at all, hence reducing the level of the desired signal in the outputs to virtually zero. After reaching this point, increasing the noise level further will simply force more of it over the threshold which increases this effect. Since white noise has a flat spectrum, most of it exceeds the threshold at a similar time, causing the catastrophic drop in SNR shown in Figure 4.6.

On the other hand, the spectrogram factor components are designed to extract the most significant patterns in the spectrogram. As white noise was used, we would expect this to have an approximately flat and constant power spectrum which would allow it to be extracted by a single component. This means that the clustering results are not greatly affected by the noise level. However, since it is unable to eliminate this noise it will always be present in the outputs at approximately the same level as at the input.

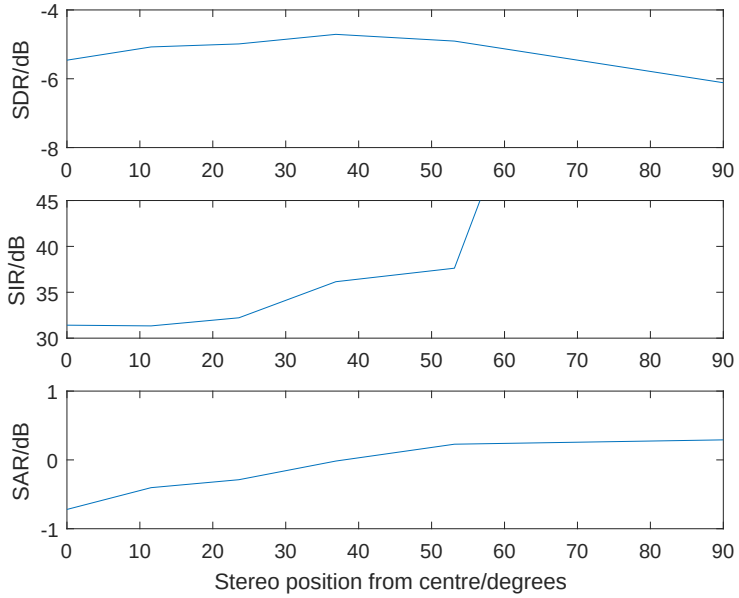


Figure 4.7: Separation performance against the stereo distance between the sounds in the input. These were obtained by using the standard test set but with fixed, as opposed to randomly selected, stereo positions. The stereo panning was achieved by applying a constant power pan-rule. For a deviation of $p$ degrees from the centre, the left channel is scaled by $\cos(p + 90)$ and the right by $\sin(p + 90)$. In this test, the two sounds were separately panned by $+p$ and $-p$ degrees, so there is no separation at $p = 0$ and complete separation at $p = 90$.

### 4.2.2 Performance Against Stereo Separation

A greater stereo distance between two trajectories gives greater confidence that they are from different sources. This is demonstrated by the system as the SIR increases with stereo distance as in Figure 4.7. One could say that any non-zero stereo distance would indicate that a pair of trajectories are not related but in practice this is not the case. We will generally observe some error in the estimate of the stereo position due to noise and random variations. This means that it would be sensible to allow some tolerance with respect to stereo distance. Furthermore, if the trajectory corresponds to colliding harmonics, the stereo position will not necessarily match that of either source, so we may still wish to consider grouping together trajectories with large stereo distances. This is captured by my system in how the sharp increase in SIR caused by the stereo information dominating occurs at a rather large stereo distance.

### 4.2.3 Performance Against Frequency Offset

The frequency of a sinusoid is important at both the feature extraction and clustering stages of the solution. If the frequencies of two sinusoids are too close together, there is a chance that the STFT peaks caused by one of them might mask the peaks of the other and so it is not extracted at all or, in the extreme case, they collide and so a single trajectory is extracted representing both of them. When two sinusoids are close in

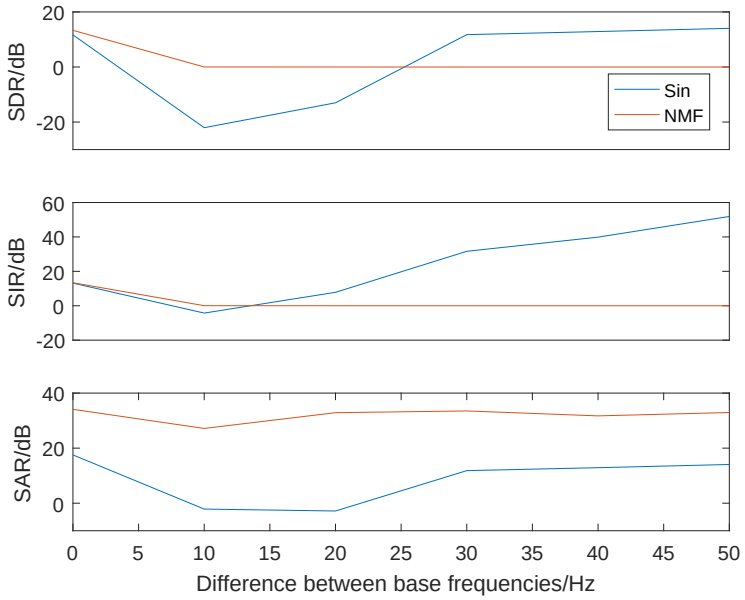


Figure 4.8: Separation performance against frequency offset between the sounds. The inputs to this were a 465Hz square wave and a sine wave at some slightly higher frequency.

frequency, those other sinusoids having harmonic concordance with one are likely to have good concordance with the other, making it harder to notice which one is from which sound. This is demonstrated by the general trend in Figure 4.8. The interesting result from this test was that having the sinusoids collide improved the quality of the outputs, compared to when they are at slightly different frequencies.

The issues with frequency do not only occur when the base frequencies of the sounds are close, but when two sounds happen to harmonise well. Figure 4.9 demonstrates this effect as we observe drastic drops in performance when the inputs have intervals of 3, 6, 8 or 12 semitones. However, one would expect that intervals of 4 and 7 would have particularly poor performance as the ratio of frequencies are approximately $\frac{5}{4}$ and $\frac{3}{2}$ whereas those identified are close to $\frac{6}{5}$, $\frac{7}{5}$ and $\frac{8}{5}$ (ignoring the $\frac{2}{1}$ from the octave). It is unfortunate that the same principle of what makes combinations of musical notes sound appealing also makes them harder to separate.

### 4.2.4 Performance Against Time Offset

The onsets of the sounds are important in this separation process when using sinusoidal trajectories in the onset distance and in determining the overlapping regions for the frequency, amplitude and stereo envelope distances (in the latter case, the normalisation for the overlap duration should not cause a great difference when varying the onset time slightly). As expected, the solution showed a gradual increase in SIR as the onset distance was increased, but I also observed an improvement in SAR immediately after introducing a small delay between the starts of the sounds.

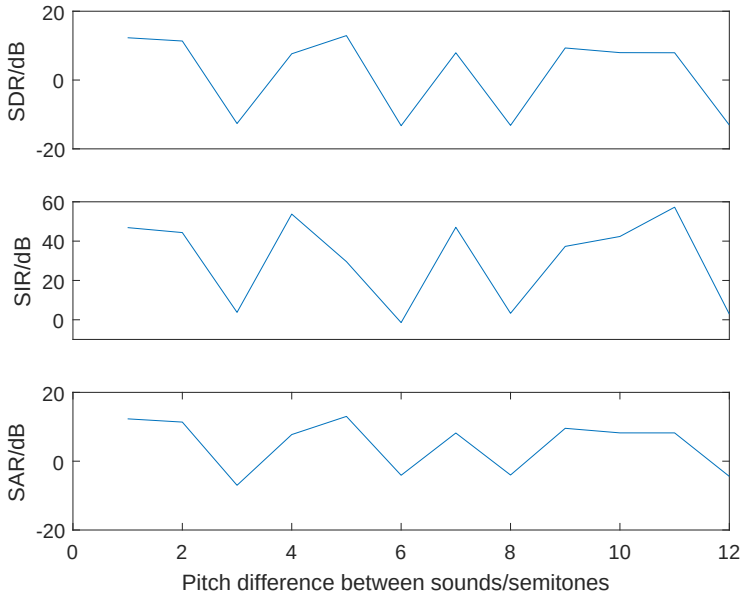


Figure 4.9: Separation performance against pitch offset between the sounds. The input here was a combination of two square waves; one at 465Hz (approximately corresponding to the MIDI note A♯4) and the other at some number of semitones above.

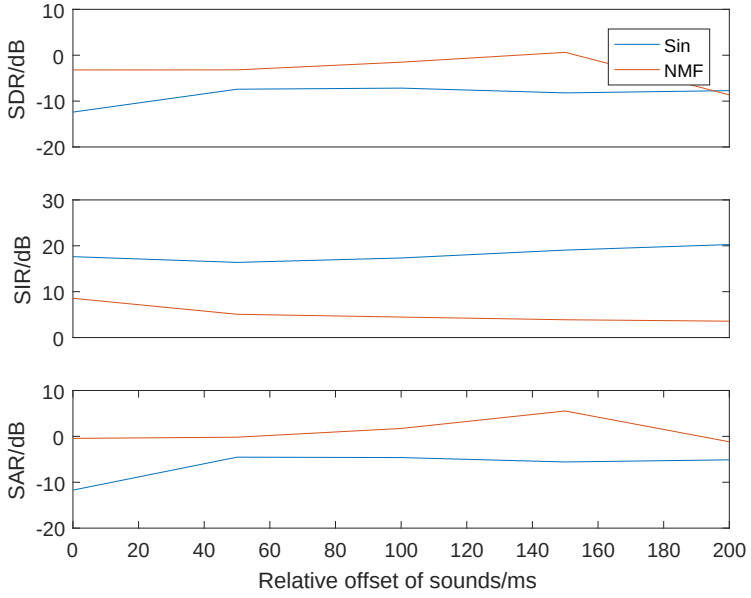


Figure 4.10: Separation performance against time offset. Two instrument samples were selected from the collection and aligned such that they appeared to be played at the same time. One was then progressively delayed to give the different test inputs.

### Computational Performance

Since time taken to run is of key importance to usability of software, it is worthwhile considering here. The main factor affecting the time taken is the number of sound elements extracted, increasing the mean time per test in the standard test set from 6.5s with an average of 14 sinusoids per output to 12.4s with 110 per output[1]. Given that the inputs were typically around 2s in duration, this is still not suitable for real-time separation. However, it would be feasible to develop variants of most stages of the process to accept real-time inputs by working on a few windows of the STFT at a time and then enforcing some continuity constraint between these frames. Simply doing this would also allow the solution to better separate sounds that do not overlap in time since the information about the earlier sounds would have been discarded before receiving the later ones, making it able to cope with complete musical pieces rather than just small snippets.

[1]Recorded on a personal computer with an Intel® Core™ i7 processor and 8GB RAM.

# Chapter 5

# Conclusion

The main goals of this project were to explore and compare the uses of sinusoidal modelling and Non-negative Matrix Factorisation for extracting elementary structures within harmonic sounds in the context of sound separation. Since beginning this investigation I have produced a tool which is capable of using either sound element model to separate a simple mixture of harmonic sounds. I have gone on to extend this solution to consider the use of a number of different clustering methods, particularly comparing the application of both hard and soft clustering techniques. I have evaluated the performance of the solution using several metrics and provided some interpretations of these results using insight about the human ability to perform this task and the design of the system.

Concerning the management of this project, I have planned and undergone a substantial research and implementation task involving the design of a small software solution to an academically interesting problem. I maintained a fluid specification as the investigation grew to include multiple degrees of comparison, many of which allowed for improvements in the quality of the solution under different metrics. Whilst time management was an issue at some points in the project timeline, any setbacks were rectified and the latter half of the work segments were completed to schedule.

Although this dissertation marks the end of the project, there are a few potential improvements on the solution that could still be made. The addition of an interactive user interface for assisted clustering could yield vastly improved separation results as humans are very good at spotting the kinds of patterns that are exhibited in the spectrum of an instrument and identifying possible issues like colliding harmonics or spectral regions with significant noise or artefacts. I would plan to do this by presenting the spectrogram of the input, highlighting each sinusoidal trajectory according to the cluster to which it is assigned, then allow the user to manually fix the clusters of some trajectories, to which the system can respond by rerunning the clustering procedure and updating the suggestions. Another further extension would be to incorporate stereo information into the extraction of spectrogram factor components by using tensor decomposition (an extension to NMF allowing it to be used on tensors of higher dimensions) or by running NMF on a concatenation of the spectrograms of the left and right channels.

The solution presented here was based on the works of Virtanen and Klapuri [26] on sound separation by sinusoidal modelling and by NMF [25] but the state of the art in the field has since moved on. Each of these techniques has been improved on with

the introduction of smoothness constraints [24] and High Resolution NMF [6]. More specialised cases have been introduced for human speech [20] which directly tackle the cocktail party problem and the applications of personal assistants and responding to voice-commands. For musically focussed separation, Celemony's Melodyne [1] is a notable leader amongst the commercial solutions, providing suitable quality for most editing and sampling needs. The solution I have presented in this project demonstrates some of the principles of sound separation that make it feasible to obtain results of a high quality. Whilst the high level of artefacts and interference may not suit this solution to separation for sampling purposes, the ability to focus on emphasizing instruments in a musical piece makes it more suitable as a tool to aid musical transcription or for balancing the levels of instruments in a poorly mixed track.

# Bibliography


[1] Celemony. `http://www.celemony.com/en/start`. Accessed: 2017-03-19.

[2] Fftw. `http://www.fftw.org/`. Accessed: 2017-03-19.

[3] Juce. `http://www.juce.com`. Accessed: 2017-03-19.

[4] Philharmonia orchestra sound samples. `http://www.philharmonia.co.uk/explore/sound_samples`. Accessed: 2017-03-19.

[5] Spectralayers pro. `http://www.magix-audio.com/gb/spectralayers-pro/`. Accessed: 2017-03-19.

[6] Roland Badeau. Gaussian modeling of mixtures of non-stationary signals in the time-frequency domain (hr-nmf). In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, pages 253–256. IEEE, 2011.

[7] J Blauert and P Laws. Group delay distortions in electroacoustical systems. *The Journal of the Acoustical Society of America*, 63(5):1478–1483, 1978.

[8] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[9] E Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the acoustical society of America*, 25(5):975–979, 1953.

[10] Bruce Fries and Marty Fries. *Digital audio essentials*. " O'Reilly Media, Inc.", 2005.

[11] Stanley A Gelfand and Harry Levitt. *Hearing: An introduction to psychological and physiological acoustics*, volume 4. Marcel Dekker New York, fifth edition, 1998.

[12] Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, 13(49-52):28, 1902.

[13] Ralph VL Hartley. A more symmetrical fourier analysis applied to transmission problems. *Proceedings of the IRE*, 30(3):144–150, 1942.

[14] Mead C Killion, Patricia A Niquette, Gail I Gudmundsen, Lawrence J Revit, and Shilpi Banerjee. Development of a quick speech-in-noise test for measuring signal-to-noise ratio loss in normal-hearing and hearing-impaired listeners. *The Journal of the Acoustical Society of America*, 116(4):2395–2405, 2004.

[15] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[16] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[17] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, fourth (version 7.2) edition, 2003.

[18] George A Miller and Patricia E Nicely. An analysis of perceptual confusions among some english consonants. *The Journal of the Acoustical Society of America*, 27(2):338–352, 1955.

[19] A Michael Noll. Cepstrum pitch determination. *The journal of the acoustical society of America*, 41(2):293–309, 1967.

[20] Mikkel N Schmidt and Rasmus Kongsgaard Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *Spoken Language Proceesing, ISCA International Conference on (INTERSPEECH)*, 2006.

[21] Michael J Shailer and Brian CJ Moore. Gap detection as a function of frequency, bandwidth, and level. *The Journal of the Acoustical Society of America*, 74(2):467–473, 1983.

[22] H V Sorensen, D Jones, Michael Heideman, and C Burrus. Real-valued fast fourier transform algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(6):849–863, 1987.

[23] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 14(4):1462–1469, 2006.

[24] Tuomas Virtanen. Algorithm for the separation of harmonic sounds with time-frequency smoothness constraint. In *Proc. Int. Conf. on Digital Audio Effects (DAFx)*, pages 35–40, 2003.

[25] Tuomas Virtanen. Sound source separation using sparse coding with temporal continuity objective. In *ICMC*, pages 231–234, 2003.

[26] Tuomas Virtanen and Anssi Klapuri. Separation of harmonic sound sources using sinusoidal modeling. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II765–II768. IEEE, 2000.

[27] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM, 2003.

# Appendix A

# Typical Outputs

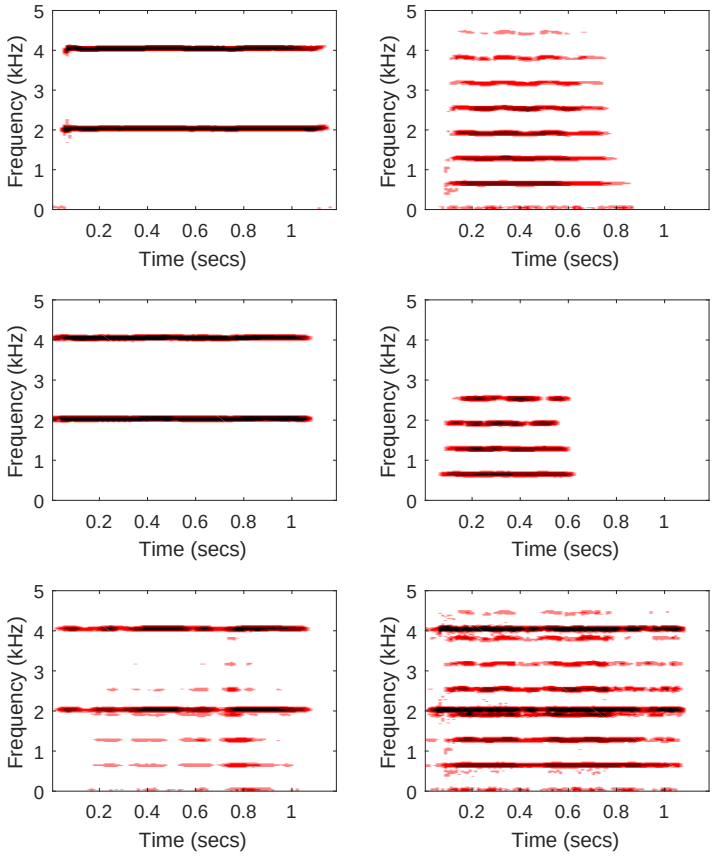


Figure A.1: Example outputs where Sinusoidal Modelling outperforms NMF. Top row: original sounds; middle row: output using sinusoidal trajectories; bottom row: output using spectrogram factor components. Here, whilst the violin (right) was not represented by as many sinusoids as one would expect, all were correctly clustered and the outputs have a remarkable likeness to the originals. On the other hand, the results when using NMF were both dominated by the clarinet (left) - one sounded as though it had correctly isolated the clarinet, and the other sounded similar to the original mixed input.

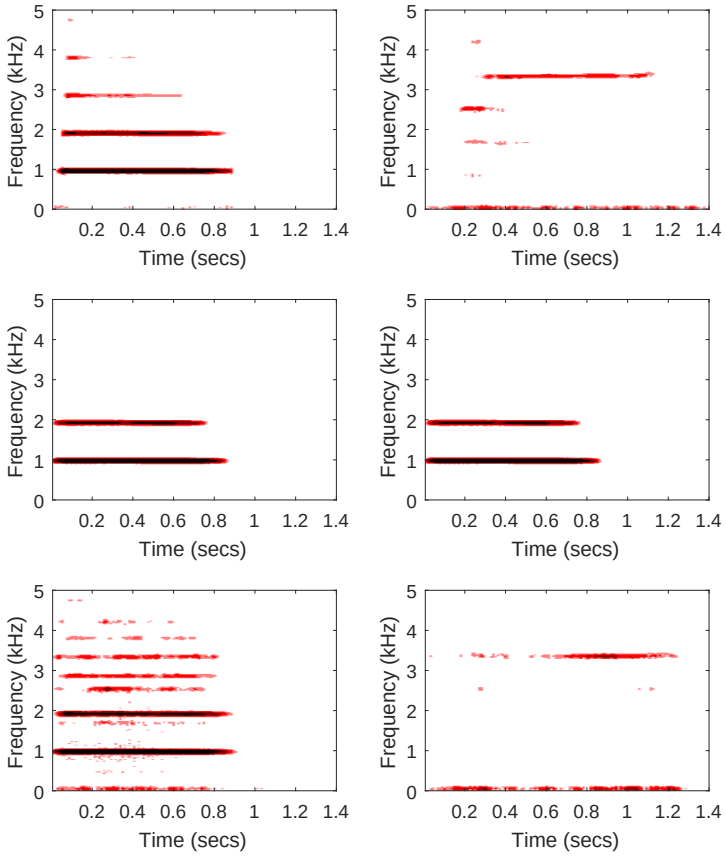


Figure A.2: Example outputs where NMF outperforms Sinusoidal Modelling. Top row: original sounds; middle row: output using sinusoidal trajectories; bottom row: output using spectrogram factor components. The particular violin sample here (right) was too quiet for the solution to detect any sinusoids from it, leaving the only ones in the system being a few from the clarinet (left). Since there were so few sinusoids identified, the outputs were not recognisable as either instrument. Since NMF does not require any thresholding of the spectrogram, it is able to detect the quiet violin note and produce an output that clearly had the clarinet removed, although the other one still had some noticeable elements of the violin in the background.

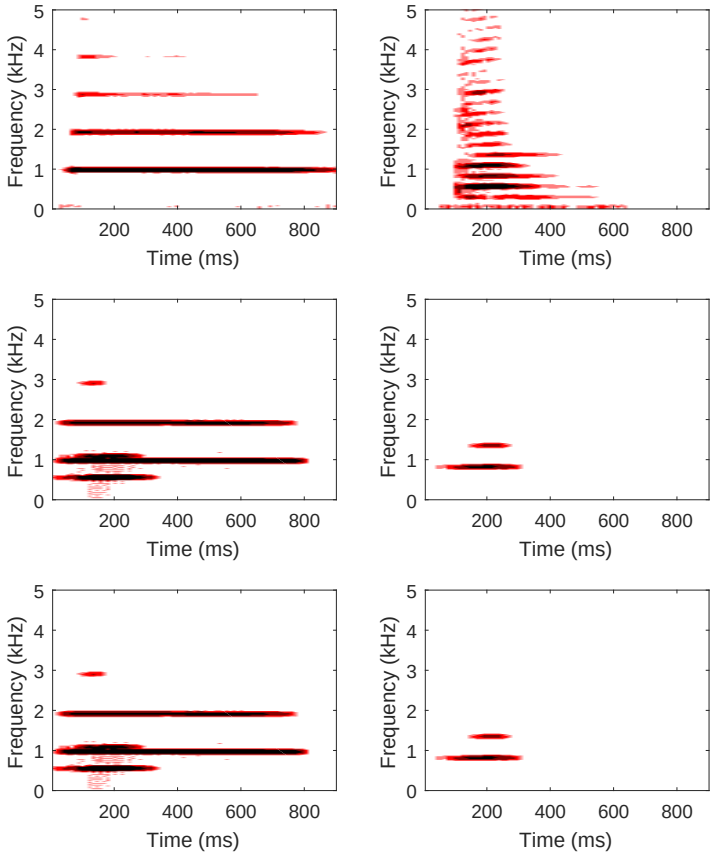


Figure A.3: Example outputs for hard and soft clustering of Sinusoidal Trajectories. Top row: original sounds; middle row: output using hard clustering; bottom row: output using soft clustering. As demonstrated here, the heuristic of taking the "hardest" assignment allows the hard and soft clustering to be practically identical. The violin sample (right) was not represented by enough sinusoids and so, whilst filtered versions could be heard in all outputs, none were recognisable as a violin. The main issue is that the base frequency was not identified, so the judgement of harmonic concordance would not have been accurate for the sinusoids from the violin. Despite this, the clarinet (left) was completely removed from one of the outputs in both cases and the reconstruction in the other was reasonably accurate.

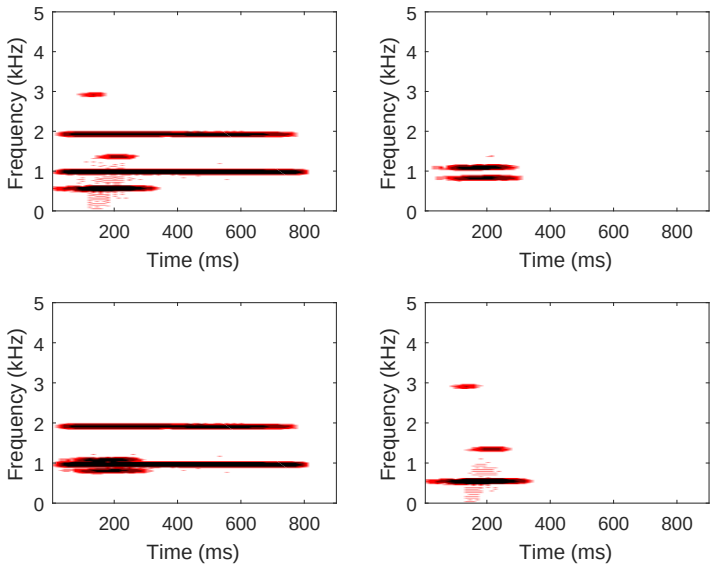


Figure A.4: Example outputs for NMF and naïve clustering of Sinusoidal Trajectories. Top row: output using NMF clustering; bottom row: output using naïve clustering. These correspond to the same input as in Figure A.3. Whilst NMF identified a different assignment from the previous examples, it suffers the same problems in terms of audible quality. The naïve approach took the first seed as the second harmonic of the violin which is not the best choice for separating these two particular sounds. The interesting issue here is that the fourth harmonic of the violin was clustered with the clarinet, despite being double the frequency of the other seed. This was because it has poor concordance with the other sinusoids already assigned to the violin output and is relatively close to the clarinet's base frequency. This example did, however, give the clearest reconstruction of the clarinet.

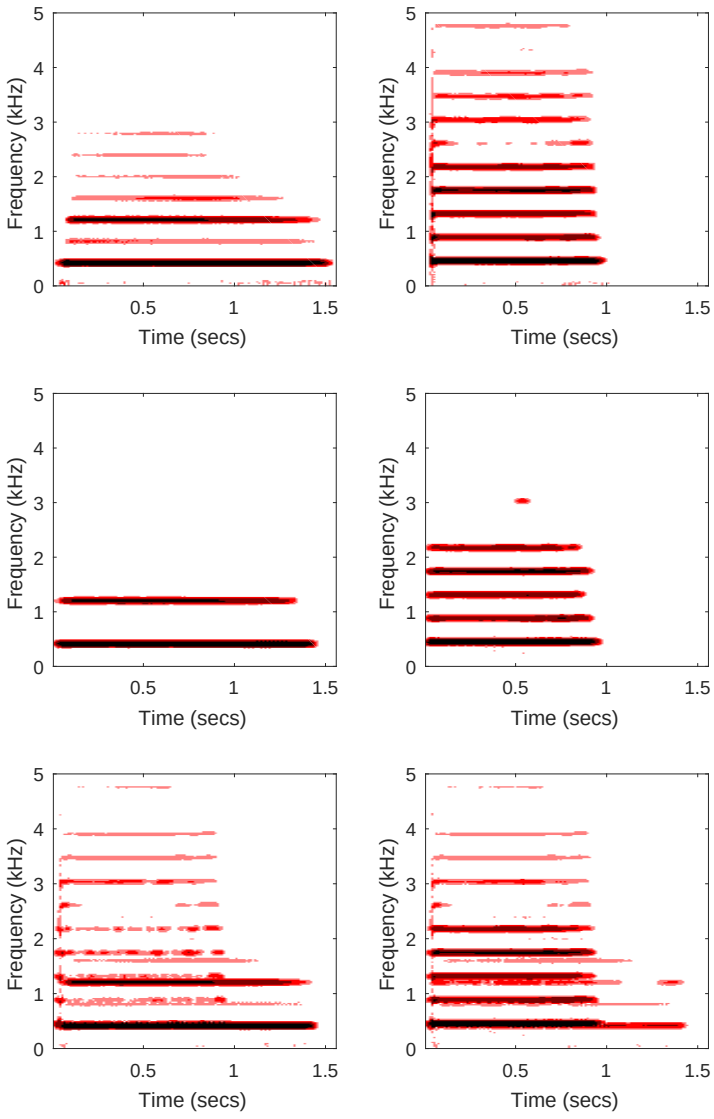


Figure A.5: Example outputs with and without the reversibility property enforced. Top row: original sounds; middle row: output using sinusoidal trajectories; bottom row: the same output when enforcing reversibility. Note how the first outputs have relatively simple spectra because few sinusoids were extracted, whereas the second pair show much richer sounds, not looking as glaringly different from the originals. The significant increase in interference is obvious but each looks more like the overall mixture with one or the other of the originals noticeably emphasized as per the intention of this feature.

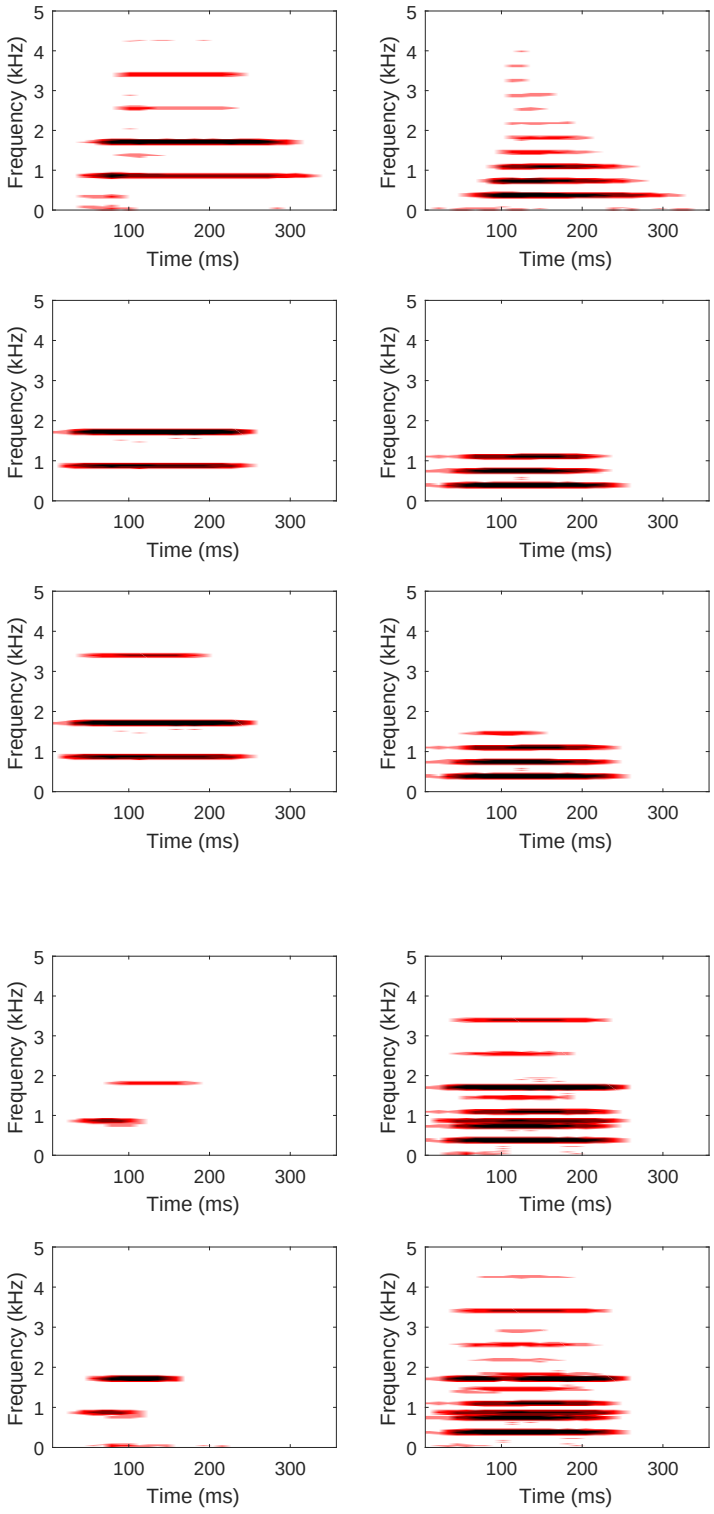


Figure A.6: Observations on increasing the number of sinusoidal trajectories extracted. Top row: original sounds; other rows use 5, 7, 12 and 42 sinusoids respectively. Note how adding a few will improve the quality of the results in every aspect but after some point the separation process can no longer correctly distinguish between sinusoids from each sound; it simply isolates a few rogue ones and leaves the rest in an increasingly accurate reconstruction of the complete mix.

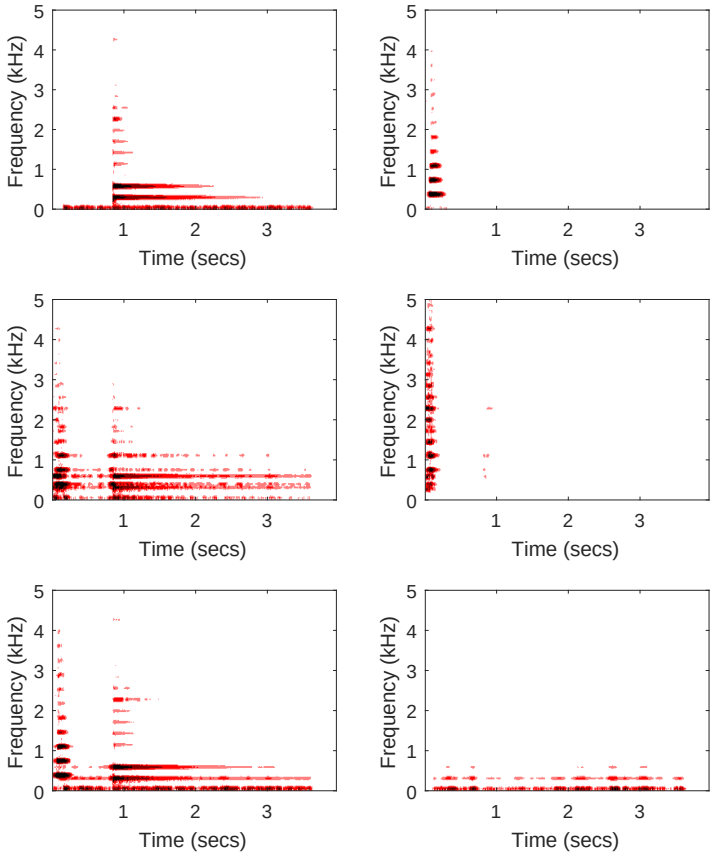


Figure A.7: Observations on increasing the number of spectrogram factor components extracted. Top row: original sounds; other rows use 2 and 10 components respectively. When we take too few, we can often get a good separation but a lot of artefacts are produced, although the original instruments are still recognisable. However, we can never guarantee that each component models part of only one of the sounds, hence why both runs produce an output that sounds like both instruments played twice. Components like this make clustering very difficult as any component from a single sound will be similar to these mixed components, so we will end up with one output resembling the complete mix and the other containing only artefact components.

# Appendix B

# Project Proposal

William Victor Simmons
St. Catharine's College
wvs22

Part II Individual Project Proposal

## Techniques for Separation of Harmonic Sound Sources

11 October 2016

**Project Originator:** William Victor Simmons

**Project Supervisor:** Dr D.J. Greaves

**Director of Studies:** Dr S.N. Taraskin

**Overseers:** Dr T.G. Griffin and Dr P. Lio

## Introduction and Description of the Work

The ability to isolate and focus on an individual sound source for identification and localisation is an ability humans possess which has proved non-trivial to simulate computationally. In particular, the "cocktail party" problem [1] is the instance of this sound separation problem for a collection of human voices in conversation. Good solutions to such problems are desirable in the fields of music and audio analysis and editing, amongst others.

The key focus of this project is to investigate and compare a selection of techniques for sound separation. The specific case of this problem being considered is where the system is provided with knowledge of the number of sound sources present and that they are purely harmonic sounds but the input contains sources with varying waveforms, time offset, base frequency and stereo positions.

## Resources Required

No special resources will be required. I intend on using my own PC (Windows 10 Home, Intel Core i7-3537U, 8.0GB RAM). I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure. In particular, all documents, sound samples and source code will be stored on a Microsoft OneDrive cloud space and version control handled through git with a repository on GitHub and further backups made to a dedicated flash drive after each work slot (see timetable). In the event of hardware damage to my PC, the MCS PCs shall suffice.

For the sound samples in use, I shall be obtaining a variety of synthetic sounds from tools within the free Audacity software and more natural sounds from the website of the Philharmonia Orchestra (provided under a Creative Commons licence), composing these using Audacity.

## Starting Point

Prior to starting this project, I have read the contents of the Part II Digital Signal Processing course in addition to reading the referenced papers on Sinusoidal Modelling. The intention of the initial research period is to familiarise myself with Non-negative Matrix Factorisation methods and an another alternative method for the extension.

## Substance and Structure of the Project

The main technique at the centre of this project is using Sinusoidal Modelling [2] to separate sinusoidal trajectories (sinusoids with slow amplitude and frequency modulation over time) into collections representing the original sources based on some relative "distance" between the trajectories.

Non-negative Matrix Factorisation [3] has been shown to be useful for sound separation methods [4] and so a solution using this will also be designed and implemented. This will be compared to the first solution under statistical and perceptive tests of quality.

Since the Sinusoidal Modelling method does not preserve phase information, the similarity between the separated output and the original sounds will be determined based on the average cross-correlation between their power spectra. We can also consider the cross-correlation of their spectrograms to capture the similarity of the sounds as they evolve through time. These will be averaged over the training set created from synthetic mixes of single note recordings of musical instruments and considering how this performance is affected by the mixing parameters.

As, in many circumstances, the output of a solution to sound separation would be used for human consumption, it would be beneficial to know how perceptually different the

output and the original sounds are. For this purpose, some human tests will be carried out to judge the quality of the reconstruction.

A third implementation of another technique may be completed as an extension if given sufficient time.

The solutions will be developed as standalone C++ applications for easy extensibility if they are to be used in further study. However, prototype solutions will be created using Matlab to ensure the principles of the solutions are suitable and aid design of the final versions. Whilst the intention of the C++ solutions is to be more extensible, quality of the separation should not be lost in doing so. The cross-correlation tests will be carried out on both the prototypes and the solutions to gauge their respective accuracies.

In most multimedia applications, it is often viewed as appropriate to discard details as long as the result is not perceptually invasive, hence the prevalence of lossy compression for visual and audio data. Human tests will be conducted to consider how perceptually different the original and reconstructed sounds are by comparing how identifiable they are. Hearing ability typically peaks before the age of 25 [5], so the testing population will be of this age group as they are more likely to be able to detect any distortion in the reconstructions.

# Reference

[1] *Some Experiments on the Recognition of Speech, with One and with Two Ears*, C. Cherry, Imperial College, University of London (`http://www.ee.columbia.edu/ dpwe/papers/Cherry53-cpe.pdf`)

[2] *Separation of Harmonic Sound Sources Using Sinusoidal Modeling*, T. Virtanen and A. Klapuri, Tampere University of Technology (`http://www.cs.tut.fi/sgn/arg/music/tuomasv/sssep.pdf`)

[3] *Algorithms for Non-negative Matrix Factorisation*, D.D. Lee and H.S. Seung (`https://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf`)

[4] *Sound Source Separation Using Sparse Coding with Temporal Continuity Objective*, T. Virtanen, Tampere University of Technology (`http://www.cs.tut.fi/sgn/arg/music/tuomasv/icmc2003.pdf`)

[5] *Presbycusis Values in Relation to Noise Induced Hearing Loss*, A. Spoor, University of Leiden

# Success Criteria

The project's success will be judged by having completed the following:

- A solution to the discussed sound separation problem using the Sinusoidal Modelling technique should be designed and implemented.
- A similar solution to the discussed sound separation problem using NMF should be designed and implemented.
- A collection of test sound files should be designed and assembled to cover a range of values for the test parameters (additive noise levels and relative pitches, time offsets and stereo positions between the sources).
- Measurements of performance of the solutions should be obtained using the test set.
- Human tests should be performed to measure for the audible similarity between the original sounds and their reconstructions after separation and perceptive quality of the separation.
- The dissertation describing this project must be written.

## Timetable and Milestones

The work units for this project will be split into segments of typically a fortnight in length. The planned starting date is Thursday 20th October 2016.

### Segment 1: 20th October - 2nd November

Read around the topics of Sinusoidal Modelling, Non-Negative Matrix Factorisation and other techniques for sound separation. Prepare sound sample set for tests.

**Deliverable:** A selection of individual harmonic sounds, at a range of pitch and time offsets, and sound files containing synthetic mixes of pairs of these.

### Segment 2: 3rd November - 16th November

Prototype the Sinusoidal Modelling solution in Matlab.

**Deliverable:** A Matlab source code file which performs sound separation using Sinusoidal Modelling and the output files from successful separation of a subset of the test files.

### Segment 3: 17th November - 30th November

Prototype the NMF solution in Matlab.

**Deliverable:** A Matlab source code file which performs sound separation using NMF and the output files from successful separation of a subset of the test files.

### Segment 4: 1st December - 28th December

Rebuild the Sinusoidal Modelling solution in C++.

**Deliverable:** The C++ source code files which perform sound separation using Sinusoidal Modelling and the output files from successful separation of a subset of the test files.

### Segment 5: 29th December - 18th January

Rebuild the NMF solution in C++.

**Deliverable:** The C++ source code files which perform sound separation using NMF and the output files from successful separation of a subset of the test files.

### Segment 6: 19th January - 1st February

Slack time for covering any setbacks when building the solutions. Investigate and optimise the distance weightings in the Sinusoidal Modelling solution based on the majority of the test set. Write the progress report.

**Deliverable:** A set of graphs describing the effects of changing the weightings on the quality of separation and reconstruction. A completed progress report ready for submission.

### Segment 7: 2nd February - 15th February

Use the remainder of the test data set to investigate how both solutions handle different pitch intervals between the sounds, additive noise levels and quantity of sounds.

**Deliverable:** A collection of statistics describing the performance of both solutions under a range of values for each of these properties of the test data.

### Segment 8: 16th February - 1st March

Slack time for producing evaluation statistics. If this is not needed, then this period will be for the extension task of implementing another alternative algorithm.

### Segment 9: 2nd March - 15th March

Begin the first draft of the dissertation, starting with the Introduction, Preparation and Conclusion sections.

**Deliverable:** The first draft of the Introduction, Preparation and Conclusion sections of the dissertation.

### Segment 10: 16th March - 5th April

Complete the first draft of the dissertation.

**Deliverable:** The first draft of the Implementation and Evaluation sections of the dissertation.

### Segment 11: 6th April - 26th April

Acquire feedback from first draft and apply this into second draft of the dissertation.

### Segment 12: 27th April - 10th May

Make final adjustments and polish the dissertation.

### Segment 13: 11th May - 19th May

Slack time for dissertation writing in case of delays due to Tripos preparation.

**Deliverable:** The completed dissertation.