Extraction of Individual Tracks from Polyphonic Music

Nick Starr

May 29, 2009

Abstract

In this paper, I attempt the isolation of individual musical tracks from polyphonic tracks based on certain criteria, such as trying to select a specific instrument. The heart of the algorithm is the use of Independent Component Analysis to separate the tracks, after which the components are to be grouped into subspaces depending on the criteria desired and recombined to put them back in a listenable format.

1 Introduction

The problem of source separation is one of very general applications. In general, it is the task of being given a set of observed signals, which are assumed to be a linear mixture of some set of source signals, and asked to find the original source signals with no further information. For example, it is used in magnetic imaging of the brain - interfering magnetic signals from other electronic equipment can be filtered out to get a clearer picture of only the magnetic fields originating from the imagine equipment. In this paper, however, source separation is applied to music. A track of music, e.g. a popular song, is a (linear) mixture of individual sources, those sources being the various instruments present. Techniques of source separation can be applied to attempt to isolate those individual sources, or "tracks", from a sequence of fully polyphonic music.

Source separation is also an intriguing area of research since it's obviously something that even a young child's brain is capable of - if, all of a sudden, a loud and distinctive instrument like a cowbell joins the mix, even the most musically uneducated listener will immediately realize the change. Computationally, however, explicitly determining this change is quite challenging and an area of active research.

2 Mathematical Background

It is assumed that the reader is familiar with the concepts of matrices and complex numbers, but other important concepts that may be less generally known will be defined here.

2.1 Matrix Transposition

The transpose of a matrix $\mathbf{M}$, written $\mathbf{M}^T$, can be explained in various ways. One way is to exchange the indices used to refer to individual elements, that is

$$\mathbf{M}_{ij} = \mathbf{M}^T_{ji} \tag{1}$$

Intuitively, what this does is "flip" the values of the matrix across the diagonal. For example:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}^T = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \tag{2}$$

A consequence of this is that the the diagonal values remain unchanged.

2.2 Eigenvalues and Eigenvectors

For a given matrix $\mathbf{M}$, when multiplication of $\mathbf{M}$ by a column vector $\mathbf{v}$ results in a scalar multiple λ of $\mathbf{v}$, then $\mathbf{v}$ is called an *eigenvector* of $\mathbf{M}$, with *eigenvalue* λ. Explicitly, we have:

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v} \tag{3}$$

More generally, a linear operator $\mathbf{L}$ defined on a vector space V (over a field F) is said to have an eigenvector $\mathbf{v}$ with eigenvalue λ when

$$\mathbf{L}\mathbf{v} = \lambda\mathbf{v} \tag{4}$$

Where $\mathbf{v}$ is a member of V and λ is a member of F.

2.3 Fourier Transforms

The traditional Fourier Transform $F(\nu)$ is found by this equation

$$F(\nu) = \int_{-\infty}^{\infty} f(t)e^{-2\pi ix\nu}dt \tag{5}$$

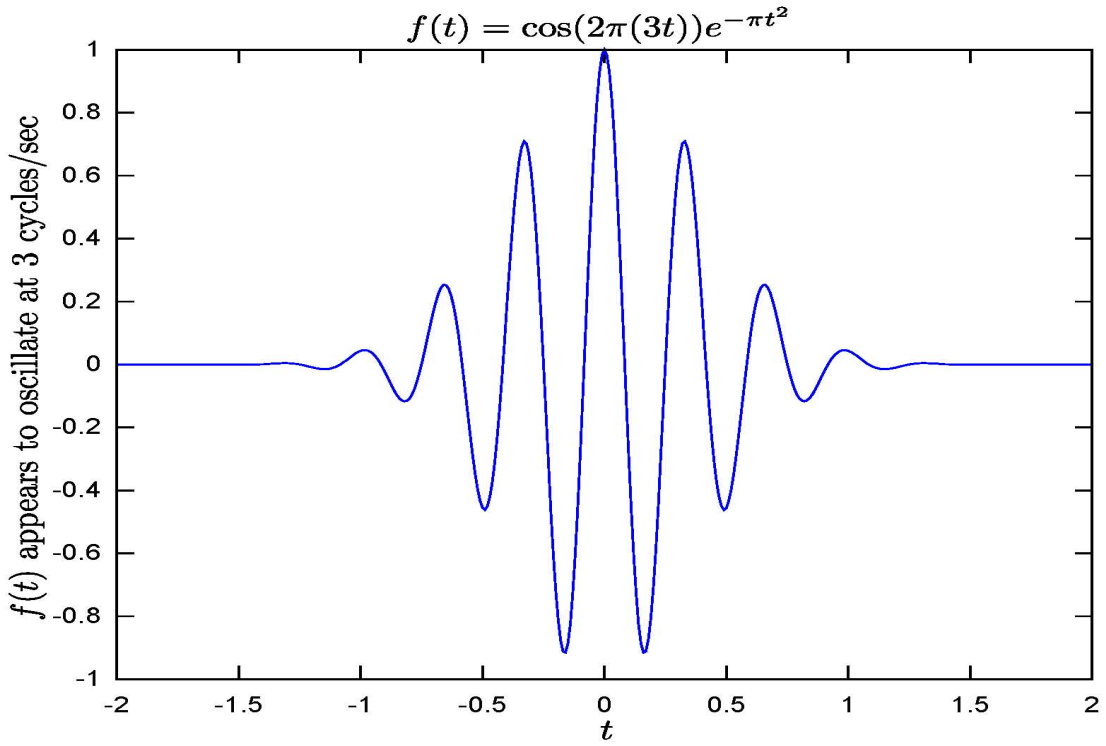

Figure 1: A periodic function with frequency of 3 Hertz.

Intuitively, what this does is transform $f(t)$, a function whose domain is time (t), to $F(\nu)$, a function who domain is frequency (ν). This means that, while f tells us information for a given point in time, F tells us information for a given frequency. This effect can be seen in Figures 1 and 2.

2.4 Short Time Fourier Transform

The Short Time Fourier Transform (STFT) is similar to a classical Fourier Transform. In the continuous case, the signal function $x(t)$ is multiplied by a window function $w(t)$ which is only non-zero for a short period of time. In our case, window used is the Hamming window, or "raised cosine" function, defined as

$$w(n) = \frac{25}{46} - \frac{21}{46}\cos(\frac{2\pi n}{N-1}) \tag{6}$$

Then the traditional Fourier Transform is taken on this new signal function, while "sliding" the window along through time, giving a two-dimensional representation of

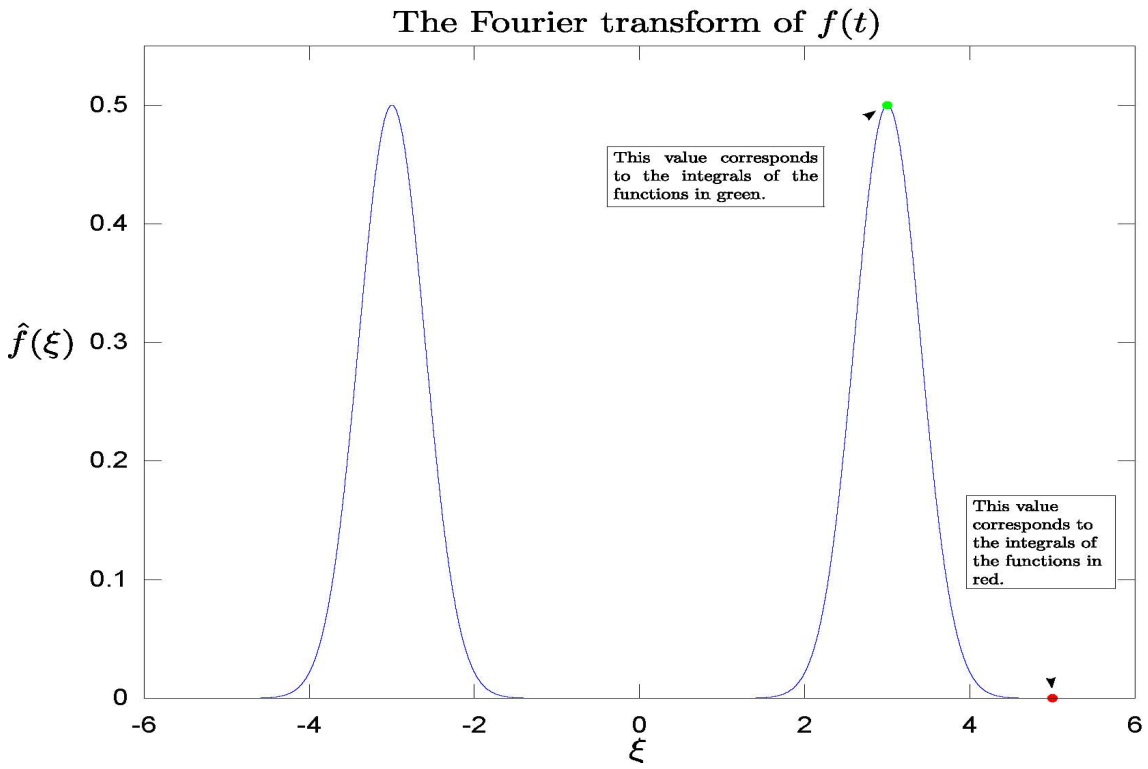

Figure 2: The fourier transform of Figure 1. Note the peaks at $\nu = 3$, -3.

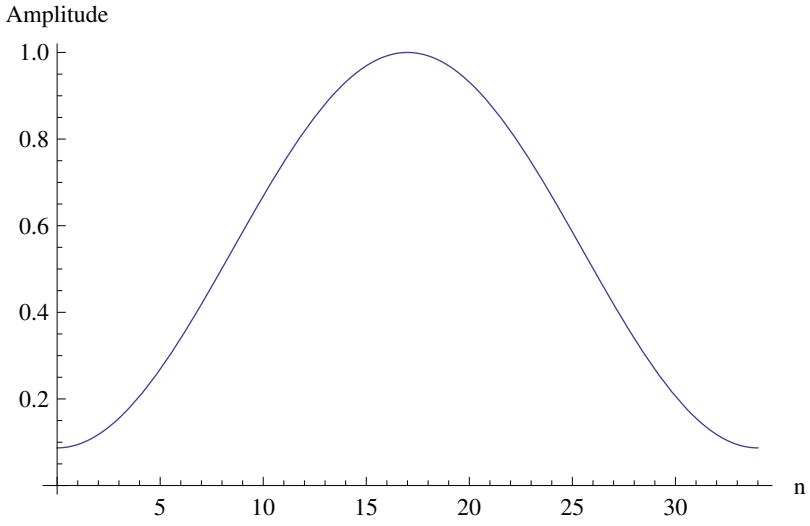

Figure 3: Plot of the hamming window for $N = 35$

the signal. Explicitly, we have that

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i\omega t}dt \tag{7}$$

Where i is the imaginary unit $\sqrt{-1}$, τ is the "slow" time, that is, the time which is being "slid" along the domain, and ω is the frequency. Where the one-dimensional Fourier Transform takes a function from the time domain to the frequency domain, the STFT gives us X as a function of both time and frequency, although the time is taken in a different sense than in the original signal.

However, in order to actually compute this transform, the discrete variant must be used. For the discrete case, there is a change in notation $t \to n$ and $\tau \to m$, and then we have the discrete transform:

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-i\omega n} \tag{8}$$

This transform can be inverted by the following formula:

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{m=-\infty}^{\infty} X(m, w)e^{i\omega n}d\omega \tag{9}$$

2.5 Singular Value Decomposition

As described in [1], the Singular Value Decomposition (SVD) of a matrix $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{10}$$

Where $\mathbf{D}$ is a diagonal matrix of singular values in decreasing order, $\mathbf{U} = (\mathbf{u}_1, ..., \mathbf{u}_m)$, or the "row basis", contains the eigenvectors of $\mathbf{X}\mathbf{X}^T$, and analogously $\mathbf{V} = (\mathbf{v}_1, ..., \mathbf{v}_n)$, or the "column basis", contains the eigenvectors of $\mathbf{X}^T\mathbf{X}$.

2.6 Source Separation

Source separation is the general framework in which the problem of extracting sound tracks is framed. However, source separation apples to more general problems, such as magnetic brain imaging or voice recognition. In the source separation model, we have a known vector of observed signals, $\mathbf{x}$, which is assumed to be the result of

multiplying an unknown matrix $\mathbf{A}$, called the "mixing matrix", with an unknown vector of source signals, $\mathbf{s}$, or $\mathbf{x} = \mathbf{As}$. What this means is that each individual measured signal is assumed to be a linear combination of all the available signals, with the coefficients encoded in the matrix $\mathbf{A}$. In index notation (with n signals):

$$x_i = \sum_{j=0}^{n} A_{ij} s_j \tag{11}$$

The goal of source separation is to determine the source vector $\mathbf{s}$ purely from knowledge of the observation vector $\mathbf{x}$. This is done by determining $\mathbf{A}$'s inverse, the "unmixing matrix", so that:

$$\mathbf{A}^{-1}\mathbf{x} = \mathbf{A}^{-1}\mathbf{As} = \mathbf{s} \tag{12}$$

Or, in index notation:

$$s_i = \sum_{j=0}^{n} A_{ij}^{-1} s_j \tag{13}$$

3 Technologies Used

3.1 C?

Originally, I had planned to do this project using the C programming language, along with the GNU Scientific Library (GSL). However, given the amount of time in the school year and the task I was attempting, the combination of C and GSL proved to be too much. As anyone who's used it can tell you, the simplest of tasks in C can result in strange errors that are difficult to pin down, and GSL compounds this by using a lot of very confusing conventions and having very little useful documentation.

3.2 Mathematica

So, my solution has been to switch to Mathematica. While Mathematica is certainly slower than C, the language is much better suited to mathematical tasks, which is what a lot of this project is. In addition, as a weakly-typed functional programming language, as opposed to C's strong typing and procedural style, it is much easier to draft workable programs quickly in Mathematica than C, and to simply spend time polishing them later. A lot of work in mathematical and scientific literature is also done in Mathematica, so there's plenty of example code to work with.

4 Description of Algorithm

4.1 Decomposition

The initial input is raw audio data. It is transformed to the spectral domain via a Short Time Fourier Transform (STFT) with n bins and m frames, from equation (7). This complex data is split into moduli $\mathbf{X}$ and phases $\mathbf{\Phi}$. The phases are only used at the end to resynthesize the original audio. Then, $\mathbf{X}^T$ is decomposed according to (9).
Next, a new matrix $\mathbf{T}$ is calculated from the following equation (where $\overline{\mathbf{D}}$ is a sub-matrix of the upper d rows of $\mathbf{D}$):

$$\mathbf{T} = \overline{\mathbf{D}}\mathbf{V}^T \tag{14}$$

This matrix $\mathbf{T}$ is then multiplied with the original moduli data $\mathbf{X}$ to produce $\overline{\mathbf{X}}$, a reduced-rank, maximally informative representation of the data:

$$\overline{\mathbf{X}} = \mathbf{T}\mathbf{X} \tag{15}$$

After the pseudo-invese $\mathbf{A}^{-1}$ is calculated, and using the following two equations

$$\mathbf{E} = \mathbf{A}^{-1}\overline{\mathbf{X}} \tag{16}$$

$$\mathbf{F}^{-1} = \mathbf{A}^{-1}\mathbf{T} \tag{17}$$

Then the individual sources can be recovered by multiplying one column of $\mathbf{F}$ with the proper row of $\mathbf{E}$. In index notation:

$$\mathbf{S}_c = \mathbf{F}_{uc}\mathbf{E}_{cv} \tag{18}$$

Where $u \in [1, n]$, $v \in [1, m]$ and $c \in [1, d]$.

4.2 Classification

In the classification step, the individual components contained in the decomposition step are evaluated by some criteria to separate them into appropriate subspaces for recombination. Unfortunately, I have not reached this stage of our program yet - however, there are examples of techniques in the literature. For example, in [3], components representing percussive events were selected by analyzing the amplitude envelope as it varies in time. Harmonic envelopes feature repeated plateaus, whereas a percussive enveloped is more "on-and-off" - the attack is quick and the decay is linear.

4.3 Recombination

Once the desired frequency components have been selected, they can be converted back into musical sound through an inverse STFT. Before performing the inverse transform, the frequency components must be recombined with the appropriate phase data, saved from the original transform as $\mathbf{\Phi}$.

5 Results and Conclusion

Unfortunately, due to a number of setbacks and difficulties, this project proved to be beyond the scope of a single year high-school research project. As discussed above, I had numerous difficulties with C and the GSL; eventually, I realized that that approach was not going to produce results in time, so I tried to switch to Mathematica in the hopes that I'd be able to get at least a prototype working quickly. However, I had more issues with Mathematica, particularly with file input and output. So, I don't really have a successful prototype for my project. However, I do have a good amount of functionality written, that would be particularly useful if I wanted to continue this research later.

References

[1] Weisstein, Eric W. "Singular Value Decomposition." From *MathWorld* - A Wolfram Web Resource. http://mathworld.wolfram.com/SingularValueDecomposition.html

[2] Weisstein, Eric W. "Fourier Transform." From *MathWorld* - A Wolfram Web Resource. http://mathworld.wolfram.com/FourierTransform.html

[3] Christian Uhle, Christian Dittmar, and Thomas Sporer, 4th International Symposium on Independent Component Analysis and Blind Signal Separation Nara, Japan, 2003.

[4] Olivier Gillet, Gaël Richard, Drum Track Transcription of Polyphonic Music Using Noise Subspace Projection