

Leveraging SAS IML for Audio Source Separation

Katharine Mary Scott, North Carolina State University

ABSTRACT

Audio source separation is an application of digital signal processing that targets the extraction of individual audio sources from composite audio signals. The ability to accurately separate audio sources is crucial for enhancing clarity and quality in environments where overlapping sounds present significant challenges. This study illustrates the application of Empirical Mode Decomposition (EMD) and Hilbert-Huang Transform (HHT) within SAS/IML to dissect a composite audio signal into its constituent sources. Starting with a merged audio file containing signals from different sources, the process begins with an individual analysis of baseline audio files to establish a baseline frequency distribution for each source. Utilizing EMD, the composite signal is decomposed into Intrinsic Mode Functions (IMFs), facilitating the isolation of distinct audio frequencies attributed to each source. The HHT is employed subsequently to scrutinize these IMFs in the frequency domain, ensuring the accuracy of separation. This approach is exemplified by successful separation of the distinct audio frequencies, where specific IMFs demarcate the distinct sources. The methodology concludes with the reconstruction of separated audio files, confirming the practical effectiveness through auditory verification. This presentation illustrates the capabilities of SAS/IML in handling complex audio processing tasks and provides a model for similar challenges in audio source separation. The applications of this technique extend to various fields requiring audio segmentation, such as multimedia editing, forensic analysis, and automated transcription services.

INTRODUCTION

Audio source separation is a critical task in the field of digital signal processing, aimed at extracting individual audio components from composite signals. This capability is essential in various applications where overlapping audio sources can compromise clarity and quality, such as in multimedia production, forensic analysis, and automated transcription services. Traditional methods often face challenges in accurately separating sources, particularly in complex auditory environments. Empirical Mode Decomposition (EMD) and Hilbert-Huang Transform (HHT) are heuristic techniques that have been developed for analyzing and processing non-linear and non-stationary signals. These methods offer practical advantages in handling the intricacies of composite audio signals, enabling the decomposition and subsequent analysis of these signals into their intrinsic mode functions (IMFs). The combination of EMD and HHT facilitates the identification and separation of individual audio sources based on their unique frequency characteristics.

In this paper, we present a method for applying EMD and HHT within SAS/IML to achieve effective audio source separation, specifically focusing on a composite audio file containing both bass and guitar tracks. Please see Figure 1 to see the process flow.

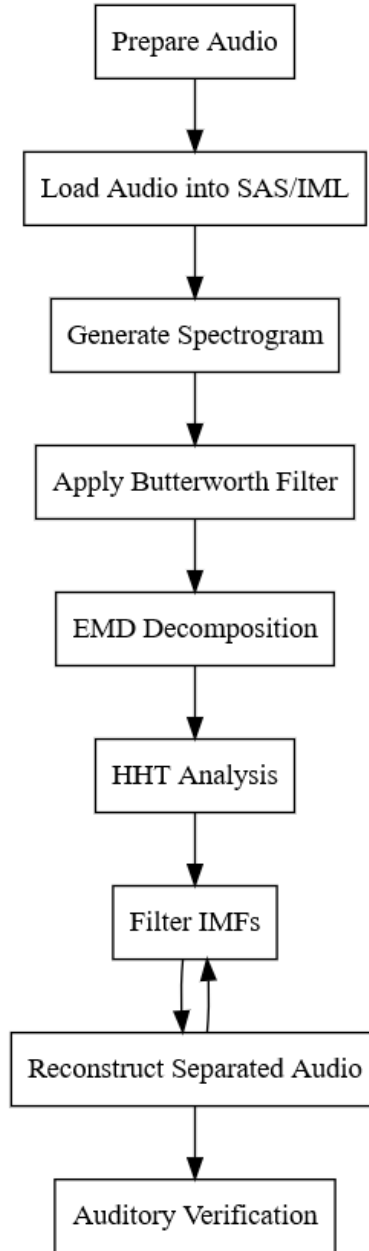


Figure 1

The process begins with the preparation of the audio file, which is converted to a mono format using FFmpeg commands to ensure consistency in the analysis. Once the file is prepared, it is loaded into SAS/IML, where the time and channel data are extracted for initial analysis. A spectrogram of the original series is generated using the spectrogram call, providing a visual baseline of the frequency content over time, which serves as a critical reference for the subsequent decomposition. Following this, the audio signal undergoes a filtering process using a Butterworth lowpass filter designed with the DFDESIGN subroutine. This filtering step is crucial as it smooths the signal and reduces high-frequency noise, facilitating more accurate decomposition. The filtered signal, containing both bass and guitar components, is then subjected to Empirical Mode Decomposition (EMD) using the EMD subroutine in SAS Viya IML. Through this process, the signal is iteratively sifted to isolate different frequency components, resulting in the extraction of multiple Intrinsic Mode Functions (IMFs), each representing distinct aspects of the original audio sources.

With the IMFs obtained, the paper proceeds to the application of the Hilbert-Huang Transform (HHT) using the HHT subroutine SAS Viya IML, which analyzes the instantaneous frequencies and amplitudes of these components. The HHT subroutine provides a detailed examination of the frequency-domain characteristics, allowing for the identification of specific frequencies associated with each audio source. This is particularly important for distinguishing between the bass and guitar tracks in the composite signal. Additionally, the HHTSPECTRUM subroutine SAS Viya IML is utilized to generate a time-frequency-energy representation of the IMFs, offering further insights into the signal's structure. Once the relevant IMFs corresponding to the bass and guitar tracks are identified, they are further processed using the DFSOSFILT function, which applies second-order section filtering to isolate these individual sources. The separated sources are then reconstructed into new audio files, each representing one of the original tracks. The effectiveness of the separation process is confirmed through auditory verification, where the reconstructed audio files are played back to ensure that the separation is both clear and accurate.

This paper demonstrates the application of EMD and HHT within SAS/IML as heuristic techniques for audio source separation. The methodology described not only enables the effective separation of bass and guitar tracks but also validates the process through the reconstruction of the separated signals. The following sections provide a detailed account of the implementation and results of applying EMD and HHT within SAS/IML, illustrating the potential of these techniques in addressing audio processing challenges. This work offers a framework for audio source separation.

METHODOLOGY

PREPARE AND LOAD AUDIO

We began the process of audio source separation by preparing the composite audio file, which included both bass and guitar tracks. Initially, the audio file was in stereo format, so we converted it to mono using FFmpeg commands. The following commands were used to create a 10-second mono WAV file:

```
ffmpeg -i .\632493__tubbers__satie-gymnopedie-1partial.wav -t 10 -c copy
guitar_bass.wav

ffmpeg -i .\guitar_bass.wav -ac 1 mono_guitar_bass.wav
```

The purpose of the commands are to make the analysis more presentable for publication.

We then loaded the prepared mono audio file into SAS/IML for further analysis using the %read_wav macro. This macro is specifically designed to read simple WAV files stored in PCM format. It extracts the audio data and stores it in a SAS dataset called mono_guitar_bass with variables corresponding to the time and channel data. In this case, we provided the file path to the mono audio file as an input to the macro:

```
proc iml;
%read_wav(file=D:\sesug2024\mono_guitar_bass.wav, _out=mono_guitar_bass);
quit;
```

GENERATE SPECTROGRAM

The next step in the process was to visualize the frequency content of the original audio signal. We began this step by generating a spectrogram of the original series. The spectrogram provides a time-frequency representation of the audio signal, allowing us to observe the distribution of frequencies over time. This allows for the possibility of increasing the smallest frequency from 48000 Hz. If there are no observable frequencies close to 48000 then we can reduce the size of the data set that is used in subsequent steps. We used the following code to generate the spectrogram:

```
title "Spectrogram of Original Series";
proc iml;
use mono_guitar_bass;
```

```

    read all var {time, channel1};
    close;
    window_length=96000/2;
    call spectrogram(channel1, window_length) truncate=0.33
        sampRate=96000 colorramp= palette("REDS", 7) scale='log';
quit;

```

The spectrogram in Figure 2 shows that the majority of the energy (represented by the darker red areas) is concentrated below 5000 Hz. This indicates that the most relevant and significant parts of the audio signal, such as the bass and guitar tracks, lie within this frequency range. According to the Nyquist-Shannon sampling theorem, to accurately reconstruct a signal, the sampling rate must be at least twice the highest frequency present in the signal. Given that the relevant frequencies are below 5,000 Hz, a sampling rate of 10,000 Hz would be sufficient. However, to further reduce the data size without significant loss of quality, down sampling using a lowpass Butterworth filter at 9,600 Hz would still capture frequencies up to 4,800 Hz.

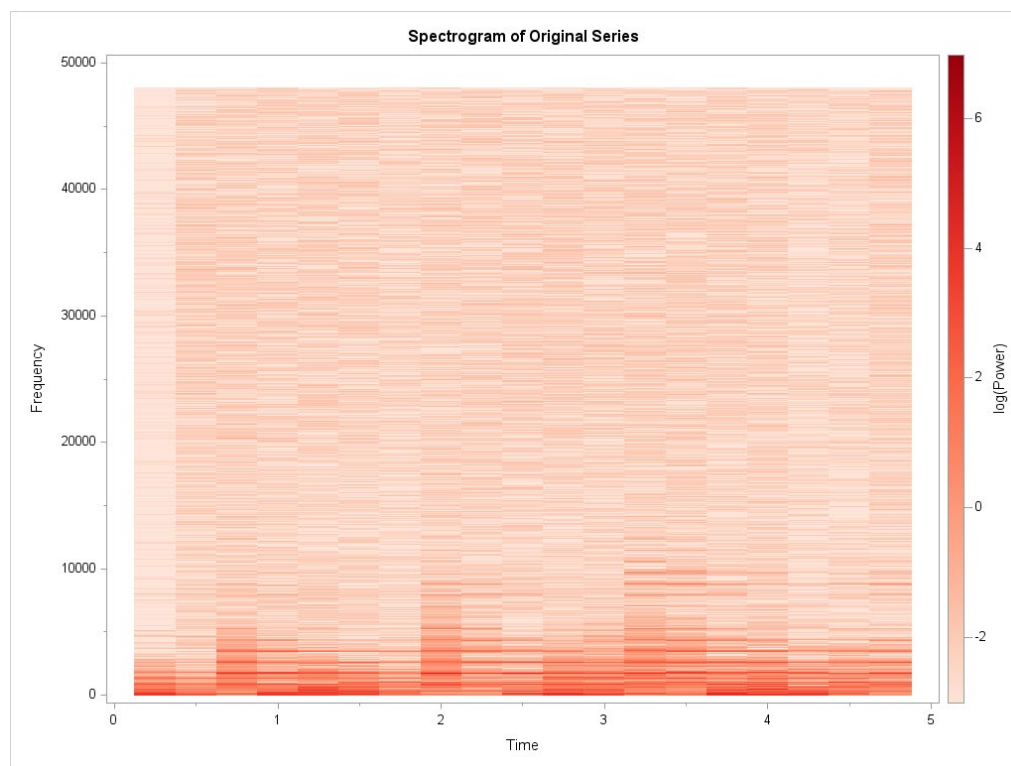


Figure 2

APPLY BUTTERWORTH FILTER

After visualizing the frequency content of the original audio signal using a spectrogram, the subsequent phase in our approach involved filtering the audio signal to emphasize the relevant frequencies, particularly those associated with the bass track. This was executed using a Butterworth lowpass filter, which we implemented in SAS/IML:

```

proc iml;
    use mono_guitar_bass;
    read all var {time, channel1};

```

```

close mono_guitar_bass;
t = time;
x = channel1;
filter_name = "butter";
filter_type = "lowpass";
n = 12;
Wc = 1/5;
call dfdesign(b, a, z, p, k, filter_name, filter_type, n, Wc);
y = dfsosfilt(x, z, p, k);
dim = dimension(y);
len = dim[1];
idx = do(1, len, 5);
time = t[idx];
channel1 = y[idx];
create mono_guitar_bass_new_5 var {time, channel1};
append;
close mono_guitar_bass_new_5;
quit;

```

We began by loading the `mono_guitar_bass` dataset into SAS/IML, which contained the time series data (`time`) and the corresponding audio signal (`channel1`). We extracted these variables for further processing. Next, we set the parameters for the Butterworth lowpass filter. The filter order was defined as 12, determining the steepness of the filter's roll-off, and the cutoff frequency was set to 9,600 Hz, which is 1/5 of the frequency (48,000 Hz) in the original data set. This specific cutoff was chosen based on the spectrogram analysis, which indicated that the significant frequencies were below this threshold. The goal was to focus on the lower frequencies, where the guitar and bass content predominantly resides.

We then employed the `dfdesign` subroutine to design the filter, which calculated the necessary coefficients, zeros, poles, and gain that define the transfer function of the Butterworth filter. With these parameters in place, we applied the filter to the audio signal using the `dfsosfilt` function to process the signal. The filtered signal preserved the lower frequencies identified in the spectrogram, reducing the impact of higher frequencies that were less pertinent to the both tracks. We stored the downsampled time and filtered audio signal in a new SAS dataset, `mono_guitar_bass_new_5`. This preparation allowed the Empirical Mode Decomposition (EMD) to run on a smaller dataset with relevant frequencies.

EMPIRICAL MODE DECOMPOSITION

Empirical Mode Decomposition (EMD) is a method used to decompose a time series into several intrinsic mode functions (IMFs) and a residual component. An IMF represents oscillatory patterns with narrow frequency bands, while the residual captures any remaining noise or trends. The EMD algorithm operates through a process called sifting, which iteratively identifies and refines IMFs. The sifting process begins by treating the input signal as the first potential IMF, and through multiple iterations, the algorithm identifies local extrema, creating upper and lower envelopes using cubic splines. The mean of these envelopes is then subtracted from the signal to refine the potential IMF. This process is repeated until certain criteria are met, such as reaching a maximum number of iterations or meeting a convergence threshold, after which the potential IMF is finalized.

Each iteration of the sifting process checks if the refined potential IMF meets the conditions to be classified as an IMF. If it does, it is accepted, and the next potential IMF is derived by subtracting the identified IMF from the original signal. This iterative refinement continues until the process identifies all relevant IMFs or specific stopping conditions are met. These conditions include the similarity between the number of extrema and zero-crossings, reaching a predetermined maximum number of IMFs, or when the energy ratio of the current IMF falls below a specified threshold, indicating that further decomposition would be uninformative. This ensures that the decomposition is efficient and meaningful.

This SAS/IML code is designed to perform Empirical Mode Decomposition (EMD) on the audio signal output from applying the Butterworth lowpass filter:

```
proc iml;
  use mono_guitar_bass_new_5;
  read all var {time channel1};
  close;
  if nrow(time) = 0 then do;
    print "Error: time matrix is empty or not properly read.";
    stop;
  end;
  SampRate = mean((time[2:10] - time[1:9]));
  pi = constant("pi");
  fs = 1 / SampRate;
  signal = channel1;
  music_rows = nrow(signal);
  time_rows = nrow(time);
  num_IMFS=7;
  optn = j(1, 4, .);
  optn[1] = num_IMFS;
  call EMD(IMF, residual, signal, optn);
  IMFCols = ncol(IMF);
  print(IMFCols);
  title "Intrinsic Mode Functions of Guitar and Bass combination";
  call panelSeries(time, IMF)
    label={"time" "IMF"}
    grid={x y}
    nrows= IMFCols
    option="lineattrs=(color=CXDC143C)";
quit;
```

The code begins by loading the `mono_guitar_bass_new_5` dataset, which contains the down sampled time series (`time`) and the audio signal (`channel1`). The `use` statement opens the dataset, and the necessary variables are extracted for analysis. After reading the data, a validation step is performed to ensure the time matrix is not empty, as indicated by the condition `if nrow(time) = 0`. If the matrix is empty or not properly read, an error message is displayed, and the procedure stops, ensuring that the input data is correctly loaded before moving forward with the analysis.

Next, the number of intrinsic mode functions (IMFs) to be extracted is set to 7, a value assigned to the variable `num_IMFS`. An options vector `optn` is created, initialized with four elements, with the first element set to the `num_IMFS`. This options vector is then passed to the `EMD` subroutine which decomposes the `channel1` signal into multiple IMFs and a residual. The IMFs are stored in the IMF matrix, and the residual, representing the remaining low-frequency trends or noise, is stored separately.

To visualize the IMFs, the call `panelSeries` subroutine is used. to generate a grid of plots, each representing one IMF over time in Figure 3.

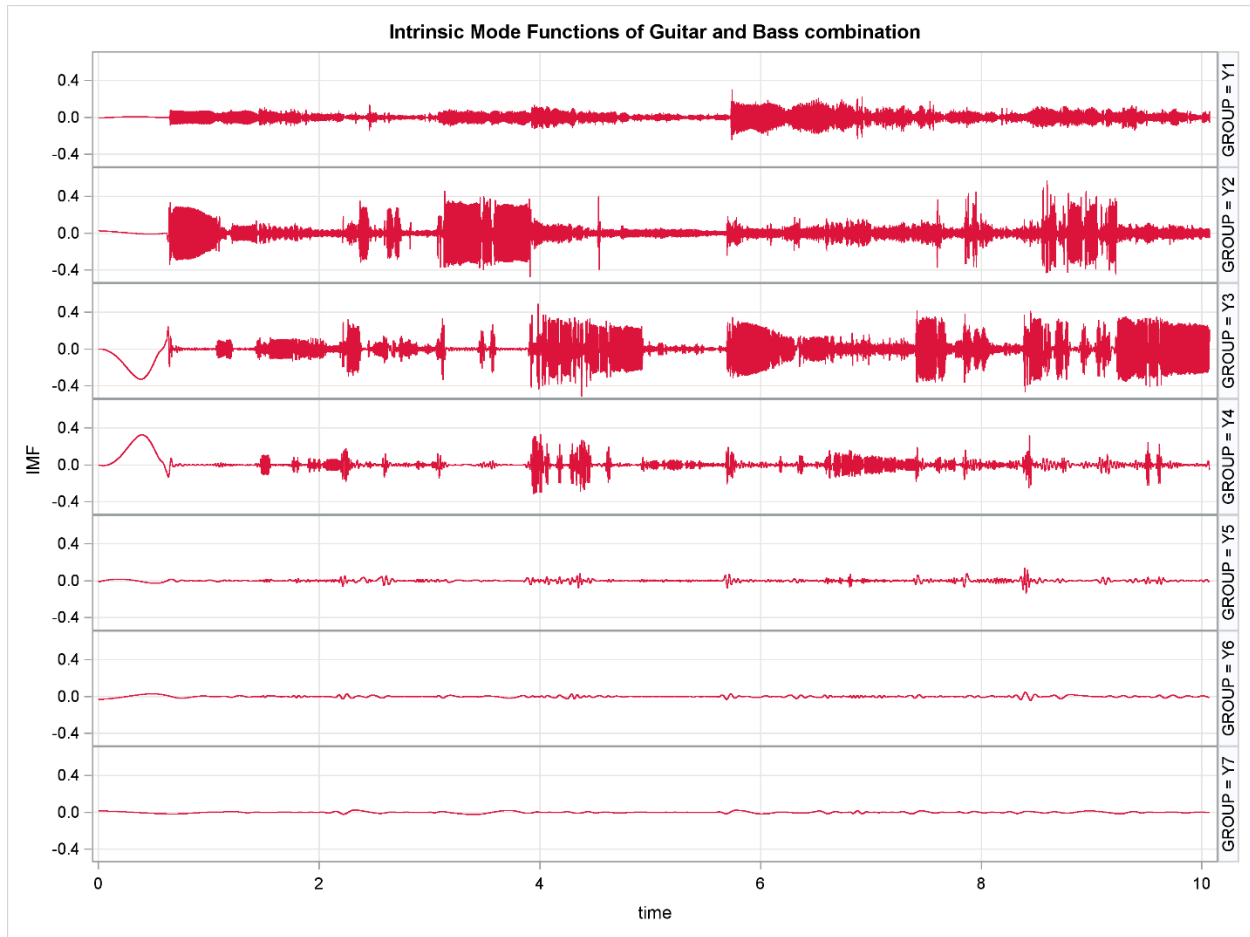


Figure 3

Based on the plot, The audio signal is concentrated within the first four intrinsic mode functions (IMFs). These IMFs capture the key oscillatory patterns and frequencies associated with the guitar and bass tracks. When computing the HHT we will restrict the number of IMFs to be 4.

HHT ANALYSIS

Next SAS/IML's Hilbert-Huang Transform (HHT) functionality is used to analyze these IMFs:

```
proc iml;
  use mono_guitar_bass_new_5;
  read all var {time channel1};
  close;
  if nrow(time) = 0 then do;
    print "Error: time matrix is empty or not properly read.";
    stop;
  end;
  SampRate = mean((time[2:10] - time[1:9]));
  pi = constant("pi");
```

```

fs = 1 / SampRate;
signal = channel1;
music_rows = nrow(signal);
time_rows = nrow(time);
num_IMFS=4;
optn = j(1, 4, .);
optn[1] = num_IMFS;
call EMD(IMF, residual, signal, optn);
IMFCols = ncol(IMF);
print(IMFCols);
title "Intrinsic Mode Functions of Guitar and Bass combination";
call panelSeries(time, IMF)
    label={"time" "IMF"}
    grid={x y}
    nrows= IMFCols
    option="lineattrs=(color=CXDC143C)";
freqRes = .;
freqRange = .;
energyThreshold = 0.0;
format = 'SPARSE';
call HHT(Amp, Phase, HHTFreqs, time, IMF, fs);
Energy = Amp##2;
title "Instantaneous Frequencies of IMFs for the Guitar and Bass
combination";
call PanelScatter(time, hhtFreqs[, 1:4])
    colorvar = Energy[, 1:4]
    colorramp = palette("REDS", 7)
    nrows = 1
    option = "markerattrs=(symbol=CircleFilled size=3)";
HHTcols=ncol(hhtfreqs);
Energycols=ncol(Energy);
EMD_HHT=j(time_rows,HHTcols+Energycols+IMFCols+1);
EMD_HHT[,1]=time;
EMD_HHT[,2:HHTcols+1]=HHTFreqs;
EMD_HHT[,HHTcols+2:HHTcols+1+EnergyCols]=Energy;
EMD_HHT[,HHTcols+2+EnergyCols:HHTcols+EnergyCols+IMFCols+1]=IMF;
HHTNames = "HHTFreq1":("HHTFreq"+strip(char(HHTcols)));
EnergyNames = "Energy1":("Energy"+strip(char(EnergyCols)));
IMFNames = "IMF1":("IMF"+strip(char(IMFCols)));
varNames = "time" || HHTNames || EnergyNames || IMFNames;
create EMD_HHT from EMD_HHT[colname=varNames];
    append from EMD_HHT;
close EMD_HHT;
title "Hilbert Huang Spectrum of IMFs for the Guitar and Bass
combination";
call HHTSPECTRUM(spectrum, specfreq, time, IMF, fs,
    freqRes, freqRange, energyThreshold, format);
call Scatter(time, specfreq)
    colorvar=spectrum
    colorramp = palette("REDS", 7)
    option="markerattrs=(symbol=CircleFilled size=3)";

quit;

```

The main objective is to use the time-frequency information output by the HHT and the Hilbert-Huang Spectrum (HHS) to separate the bass frequencies from the guitar frequencies. The call HHT subroutine directly computes the instantaneous amplitude, phase, and frequency for each IMF, using the previously calculated sampling frequency to ensure accurate time-frequency analysis. SAS/IML calculates the energy of each IMF by squaring the instantaneous amplitude, allowing us to measure the signal's strength at each point in time. This allows for us to identify the specific frequency ranges corresponding to the bass and guitar tracks using SAS/IML's call PanelScatter subroutine. We can visualize the instantaneous frequencies and their energy levels, allowing us to distinguish between bass and guitar. This plot is in Figure 4.

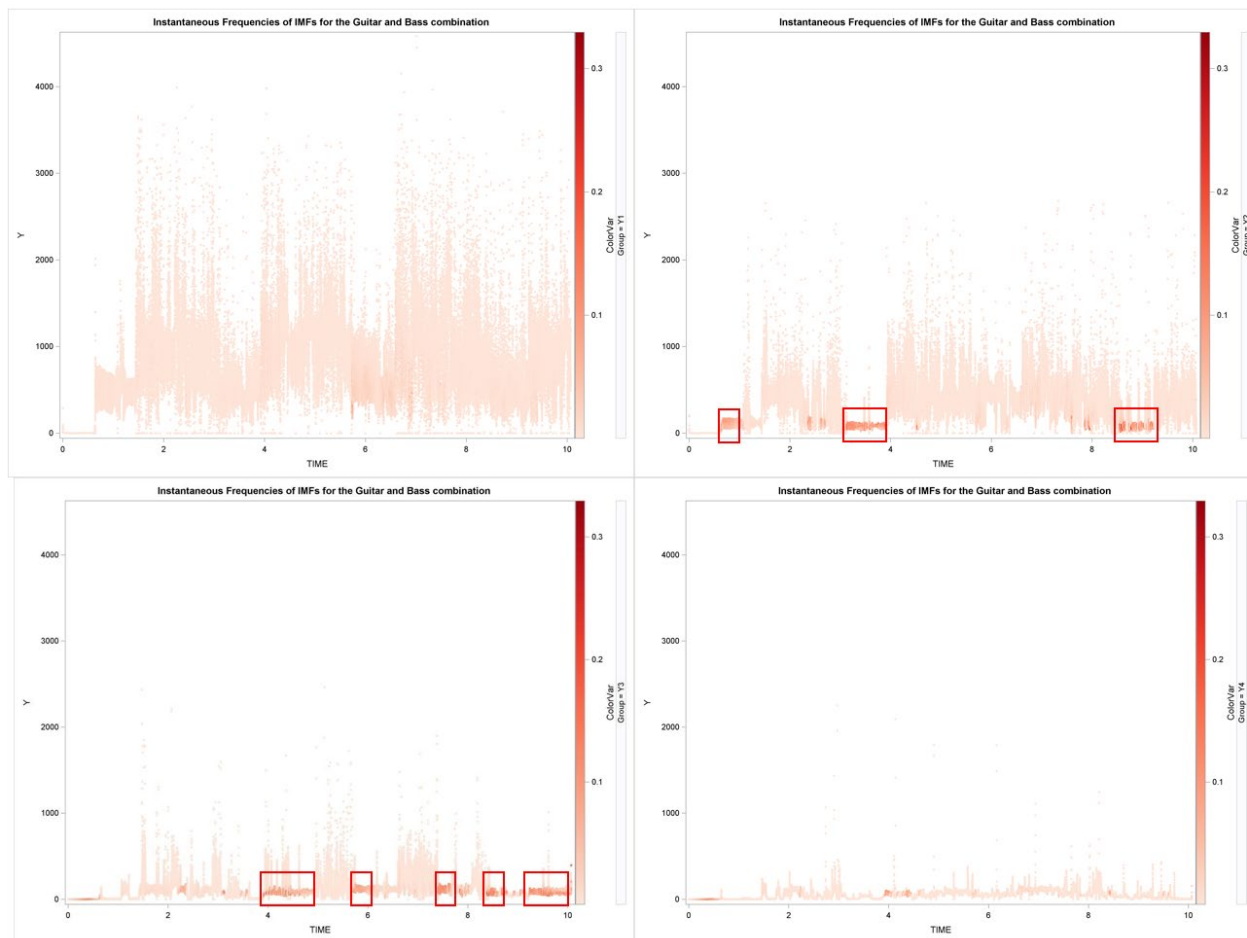


Figure 4

The HHT instantaneous frequency plots provides a view of the audio signal, showing how the bass and guitar frequencies are distributed over time. In the first IMF (top left), the plot captures higher frequency content, characteristic of the guitar sound, with activity concentrated in the upper frequency range.

In contrast, the second IMF (top right) displays a mix of both high and low frequencies. The red boxes in this plot highlight specific regions where lower frequencies, typical of the bass, are more prevalent. This suggests that IMF 2 contains both guitar and bass frequencies, with distinct segments where the bass frequencies are more pronounced.

The third IMF (bottom left) clearly represents the bass frequencies, capturing the lower frequency content with the red boxes marking areas where the bass is most concentrated. The fourth IMF (bottom right) shows minimal activity across the frequency spectrum and thus does not contribute meaningfully to the representation of either the bass or the guitar frequencies.

Finally, the call HHTSPECTRUM function in SAS/IML generates the Hilbert-Huang Spectrum (HHS), providing a comprehensive view of how energy distribution across frequencies changes over time. This spectrum, visualized using the call Scatter function, shows how the bass and guitar components are separated within the IMFs in Figure 5.

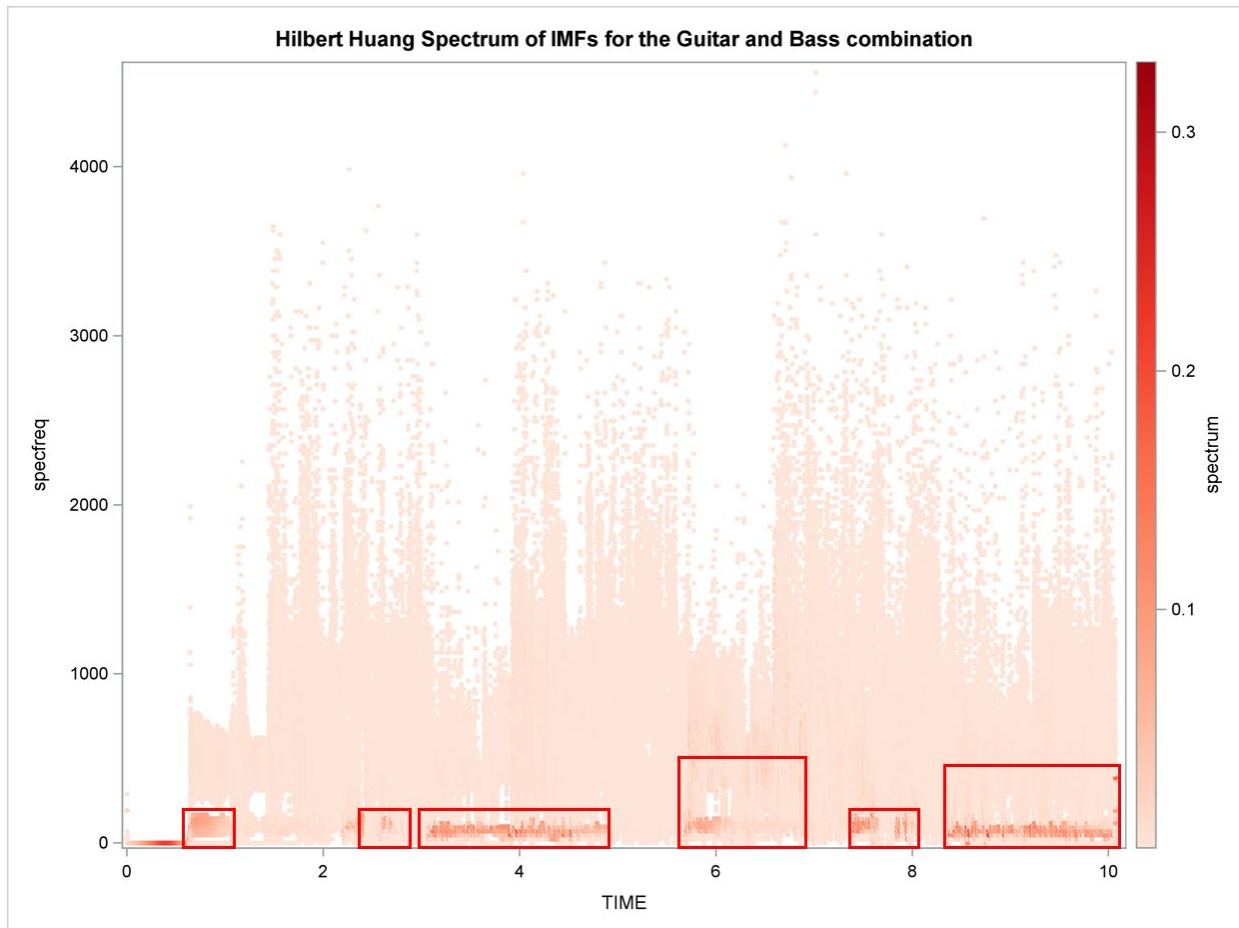


Figure 5

The highlighted sections within the red boxes further confirm the separation of the bass from the guitar frequencies. The red-boxed regions clearly indicate where the lower frequencies, associated with the bass, are concentrated. These areas correspond to distinct times within the signal in line with the where the bass is used in the original WAV file. In contrast, the higher frequencies, which correspond to the guitar, are more dispersed across the time frame and are less prominent in these highlighted regions.

The plot suggests that to accurately reconstruct the guitar sound, we should apply a Butterworth lowpass filter with a 3200 Hz cutoff to the first IMF. This cutoff frequency is appropriate because it captures the higher frequency content typical of the guitar while excluding the lower frequencies associated with the bass. For the bass, the plot indicates that applying a Butterworth lowpass filter with a 480 Hz cutoff to the

second and third IMFs would be effective. This lower cutoff frequency isolates the bass frequencies, allowing for a clear separation from the guitar sound.

In practice for more complicated separation problems achieving optimal separation requires iterative adjustments to the filtering process. Iteration could involve fine-tuning the cutoff frequencies or reassessing which IMFs are most representative of the sounds. By repeatedly refining these parameters, the separation between the bass and guitar components can be improved, leading to more accurate and distinct reconstructions of the individual sounds.

FILTERING IMFS AND RECONSTRUCTING SEPARATED AUDIO

This section details the process of filtering the Intrinsic Mode Functions (IMFs) and reconstructing the separated audio tracks for the guitar and bass. The following SAS IML code is used:

```
proc iml;
  use EMD_HHT(keep=time IMF1 IMF2 IMF3);
  read all var {time, IMF1, IMF2, IMF3};
  close EMD_HHT;
  t = time;
  x1 = IMF1;
  x2 = IMF2;
  x3 = IMF3;
  filter_name = "butter";
  filter_type = "lowpass";
  n = 12;
  wc1 = 1/3;
  wc2 = 1/20;
  call dfdesign(b, a, z, p, k, filter_name, filter_type, n, wc1);
  y1 = dfsosfilt(x1, z, p, k);
  call dfdesign(b, a, z, p, k, filter_name, filter_type, n, wc2);
  y2 = dfsosfilt(x2, z, p, k);
  y3 = dfsosfilt(x3, z, p, k);
  len= nrow(y1);
  print(len);
  idx = do(1, len, 3);
  time = t[idx];
  channel1 = y1[idx];
  create guitar var {time, channel1};
  append;
  close guitar;
  len= nrow(y2);
  print(len);
  idx = do(1, len, 20);
  time = t[idx];
  channel1 = y2[idx] + y3[idx];
  create bass var {time, channel1};
  append;
  close bass;
quit;
```

The code starts by loading the relevant IMFs (IMF1, IMF2, and IMF3) from the EMD_HHT dataset, which represent the decomposed components of the original audio signal. IMF1 primarily captures the higher frequency content associated with the guitar, while IMF2 and IMF3 contain the lower frequencies linked to

the bass. To isolate the guitar frequencies, a Butterworth lowpass filter with a cutoff frequency of 3200 Hz is applied to IMF1. The filtered guitar signal is then downsampled to 3200 Hz and saved in the guitar dataset. For the bass frequencies, a Butterworth lowpass filter with a cutoff frequency of 480 Hz is applied to IMF2 and IMF3. These IMFs contain the bass frequencies, and the 480 Hz cutoff ensures the exclusion of higher guitar frequencies. The signals are then combined and downsampled to create the final bass signal, which is stored in the bass dataset. The filtered and downsampled signals for both guitar and bass are saved separately, allowing for the reconstruction of the individual audio tracks. This process may require iteration to fine-tune the filter parameters or the selection of IMFs to achieve the best separation of guitar and bass components, ensuring the reconstructed audio signals are clear and distinct. Finally, the filtered and downsampled signals for the guitar and bass are saved as separate WAV files using the %write_wav macro:

```
%write_wav(indata=work.guitar,sampling_frequency=3200,_out=%nrquote(V:\sesug2024\mono_guitar_bass\guitar.wav) );

%write_wav(indata=work.bass,sampling_frequency=480,_out=%nrquote(V:\sesug2024\mono_guitar_bass\bass.wav) );
```

The guitar signal is saved as guitar.wav. Similarly, the bass signal, is saved as bass.wav.

CONCLUSION

We demonstrated the effective application of SAS/IML for audio source separation by leveraging Empirical Mode Decomposition (EMD) and Hilbert-Huang Transform (HHT) techniques. We successfully isolated the guitar and bass frequencies from a composite audio signal by employing targeted filtering methods, specifically using Butterworth lowpass filters with carefully chosen cutoff frequencies. The process involved iterative refinement of IMFs, visualization through the HHT spectrum, and precise filtering to achieve clear separation of the audio components. Finally, the separated audio tracks were reconstructed and saved as distinct WAV files, confirming the efficacy of handling audio processing tasks within SAS/IML. This approach not only highlights the flexibility of SAS/IML in signal processing but also provides a framework for similar challenges in audio source separation.

REFERENCES

- Flandrin, P., Rilling, G., & Gonçalves, P. 2004. "Empirical Mode Decomposition as a Filter Bank." *IEEE Signal Processing Letters*. 10.1109/lsp.2003.821662.
- Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N. C., Tung, C. C., & Liu, H. H. 1998. "The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-Stationary Time Series Analysis." *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*. 10.1098/rspa.1998.0193.
- Rilling, G., & Flandrin, P. 2008. "One or Two Frequencies? The Empirical Mode Decomposition Answers." *IEEE Transactions on Signal Processing*. 10.1109/tsp.2007.906771.
- Wu, Z., & Huang, N. E. 2009. "Ensemble Empirical Mode Decomposition: A Noise-Assisted Data Analysis Method." *Advances in Adaptive Data Analysis*. 10.1142/s1793536909000047.
- Torres, M. E., Colominas, M. A., Schlotthauer, G., & Flandrin, P. 2011. "A Complete Ensemble Empirical Mode Decomposition with Adaptive Noise." *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic: IEEE. Available at <https://ieeexplore.ieee.org/document/5947265>.
- SAS Institute Inc. 2024. *SAS/IML User's Guide: Digital Signal Process*. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_054/imlug/imlug_dsp_toc.htm.

SAS Institute Inc. 2024. *SAS/IML User's Guide: Language Reference: DFDESIGN Call*. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_054/imlug/imlug_langref_sect113.htm.

SAS Institute Inc. 2024. *SAS/IML User's Guide: Language Reference: DFSOSFILT Call*. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_054/imlug/imlug_langref_sect119.htm.

SAS Institute Inc. 2024. *SAS/IML User's Guide: Language Reference: EMD Call*. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_054/casimllang/casimllang_common_sect112.htm.

SAS Institute Inc. 2024. *SAS/IML User's Guide: Language Reference: HHT Call*. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_054/casimllang/casimllang_common_sect160.htm.

SAS Institute Inc. 2024. *SAS/IML User's Guide: Language Reference: HHTSPECTRUM Call*. Available at https://documentation.sas.com/doc/en/pgmsascdc/v_054/casimllang/casimllang_common_sect161.htm.

Tubbers. 2024. *Gymnopédie No. 1* [Audio file]. Available at <https://freesound.org/people/tubbers/sounds/632493/>.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Laura Gonzalez, Arin Chaudhuri, and Yuwei Liao, all part of the IoT Division, Advanced Analytics Department, for their invaluable support and expertise. As developers of the IML calls used in this article, their guidance was instrumental in applying and understanding these tools for the purpose of audio separation. I am also profoundly grateful to Jonathan Duggins, Associate Teaching Professor at North Carolina State University, for his encouragement to engage in the research process and for his scholarly guidance, which has been crucial in the development of this article. Any errors in this paper are entirely my own.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Katharine Mary Scott
Kmscott6@ncsu.edu