

864612 731613 129715 640716 58717

Music Transcription Through Frequency Reassignment

By:

Ryan Koehn

Matt Lopez

Music Transcription Through Frequency Reassignment

By:

Ryan Koehn
Matt Lopez

Online:

< <http://cnx.org/content/col11393/1.1/> >

C O N N E X I O N S

Rice University, Houston, Texas

This selection and arrangement of content as a collection is copyrighted by Ryan Koehn, Matt Lopez. It is licensed under the Creative Commons Attribution 3.0 license (<http://creativecommons.org/licenses/by/3.0/>).

Collection structure revised: December 19, 2011

PDF generated: December 19, 2011

For copyright and attribution information for the modules contained in this collection, see p. 23.

Table of Contents

1 Introduction	1
2 Background	3
3 Frequency Reassignment	5
4 Preparing the Signal	7
5 Analyzing the Signal	9
6 Simple Input	11
7 Complex Input	15
8 Results	19
9 Conclusions	21
Attributions	23

Chapter 1

Introduction¹

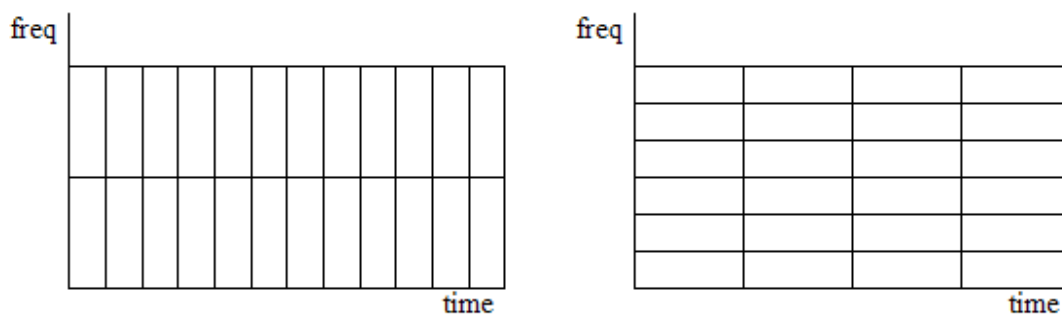
Imagine yourself as a musician, and you hear a song that you would like to play/practice. Either you can go to a music store and buy the score, or order it online; HOWEVER, there is a third option: you can use a program that will take an audio file as an input, and output the notes of the piece. Our group project forms a foundation for such a musical transcription program.

¹This content is available online at <<http://cnx.org/content/m41893/1.1/>>.

Chapter 2

Background¹

The most natural way of analyzing the notes of an audio signal is through a time-frequency representation of it. This time-frequency representation is known as a spectrogram. However, this spectrogram has some resolution issues stemming from the windowing operation used on the signal in order to compute the STFT (Short-Time Fourier Transform) of it, putting it in the form we see: time vs. frequency. This windowing has an effect not unlike the Heisenberg Uncertainty Principle. A wide window improves the frequency resolution, and a narrow window improves the time resolution, thus we can never achieve a higher resolution in both the frequency and time domain (shown in figure below). However, we need to achieve a high resolution in both to analyze the notes with great precision.

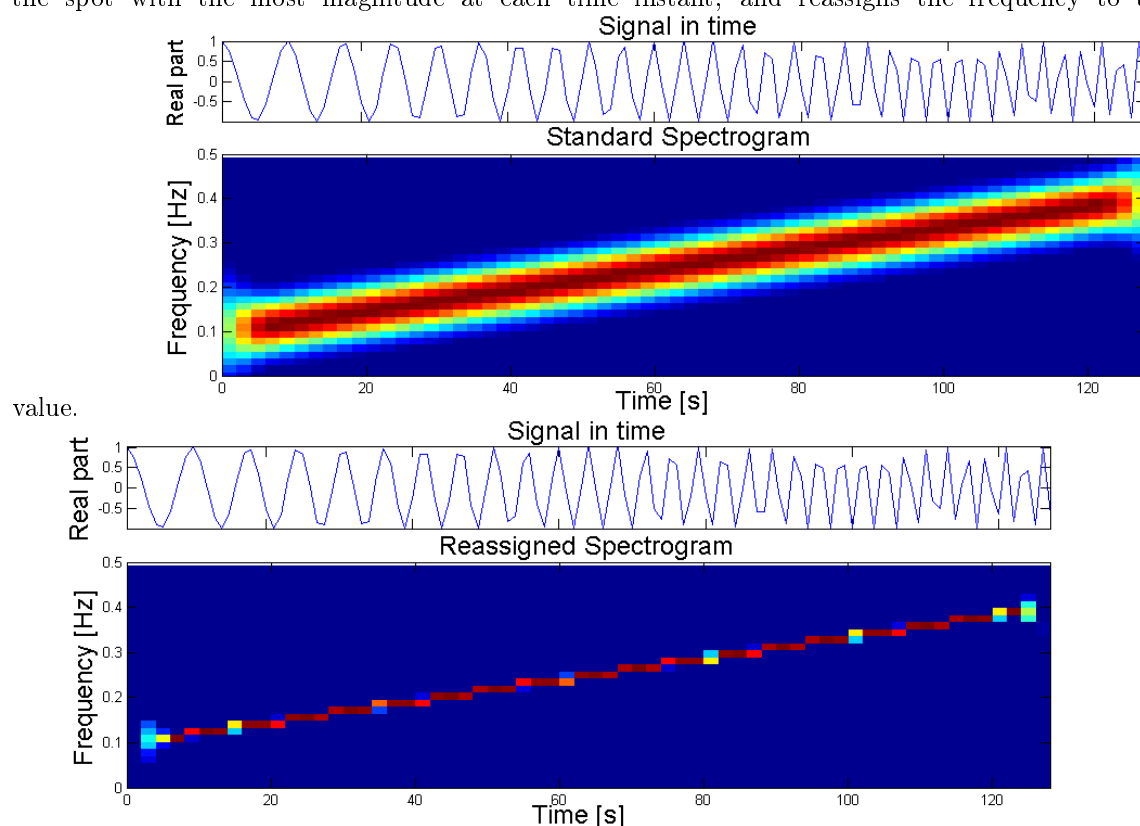


¹This content is available online at <<http://cnx.org/content/m41892/1.1/>>.

Chapter 3

Frequency Reassignment¹

There are a few different methods of achieving this high resolution of the spectrogram; but, one of the most successful in terms of implementation (and the one we used in our project) is known as the Reassignment Method. Worked on by Patrick Flandrin, Francois Auger, and other various people around the world, the Reassignment Method is quite simple in theory, but the implementation and the math behind it are indeed quite complicated. The Reassignment Method takes a “blurry” spectrogram, finds the spot with the most magnitude at each time instant, and reassigns the frequency to this single



Above is the Reassignment method implemented and computed for the very well-known chirp signal. The new, reassigned spectrogram is a much higher resolution than previous one. The wide line of a signal compresses down into a very thin line. Finding the “center of gravity” of the frequency greatly increases the

¹This content is available online at <<http://cnx.org/content/m41894/1.1/>>.

resolution and thus, we are able to analyze notes with much greater precision! With frequency reassignment in our toolbox, we can now continue along the road to reach our overall goal of musical transcription.

Chapter 4

Preparing the Signal¹

4.1 The Method of Music Transcription: Preparing the Signal

Each “module” discussed on this page and the next is a different function in MATLAB.

4.1.1 Calculation of Reassigned Time-Frequency Representation

The method of music transcription is divided into different modules, called by a top module when necessary. First of all, we will need to calculate the reassigned spectrogram using a separate module. The output of the frequency reassignment module is a matrix with n columns and m rows. The columns represent the different time instants and the rows represent the different frequencies, both positive and negative. Each element in the matrix corresponds to the magnitude at that specific frequency at that specific time instant. The code for calculating reassigned spectrograms comes from the Time-Frequency Toolbox for MATLAB created by Patrick Flandrin and Francois Auger.

4.1.2 Eliminating Negative Frequencies

On the top module, all other modules are called and several other functions are made. First we will call the frequency reassignment function to calculate the reassigned matrix of our input signal (the song). Once we have obtained the matrix, we need to get rid of all negative frequencies (since we only need to analyze the positive ones to determine the note). We do this by using a Hilbert transform, which is basically a filter that only passes positive frequencies. The matrix is now ready to be analyzed.

¹This content is available online at <<http://cnx.org/content/m41960/1.1/>>.

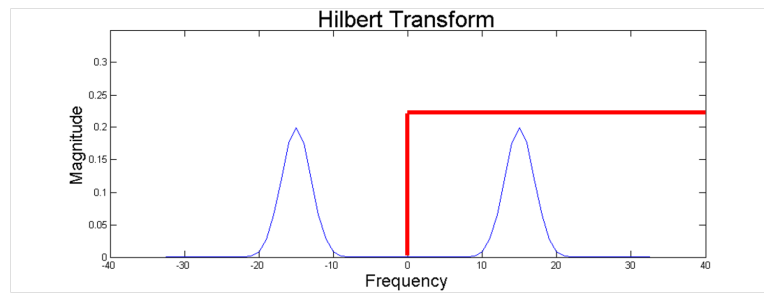


Figure 4.1

Fig. 1 A Hilbert Transform eliminates the negative frequency component

Chapter 5

Analyzing the Signal¹

5.1 The Method of Music Transcription: Analyzing the Signal

5.1.1 Distinguishing Between Notes

Our second module (`get_note`) is the one that will analyze the signal at a specific time, split the signal into the different notes, and get the frequencies of each note. Since each element of the matrix corresponds to a magnitude, the frequency with the maximum magnitude at a specific time instant represents the frequency of the signal at that time. Therefore, we need to check the maximum magnitude for every column and determine the frequency. Sometimes, however, some columns will not have a frequency because no note is being played at that time instant, but it will still have a maximum magnitude at a frequency because it has noise. To deal with this, we have set a threshold magnitude that determines whether a column has a note to be analyzed or not. If the maximum magnitude at a column is below the threshold, the frequency at that time will not be considered a note, therefore it will not be analyzed.

To identify a note, we use simple edge detectors. When the signal changes frequency, the program will analyze each column until it detects another change in frequency. The range of time instants obtained from this process will represent a single note, and the frequency of the note will be the average of the maximum amplitude frequencies of each time instant in the given range. Each frequency of each note will be inputted to the `identify_note` module one at a time for it to output the note name. To prevent any imperfections, we have also created a time threshold, such that, if the length of a note (time instants between edge detections) is lower, such note will not be analyzed because it will not be considered a note.

NOTE: The frequency and duration thresholds work together to distinguish between notes and noise. There may be a few consecutive time instants for which noise is greater than the amplitude threshold. However, no signal should have noise that is consistently greater than the amplitude threshold for the duration of the duration threshold or longer. As such the two thresholds work in tandem to eliminate notes from the transcription process.

NOTE: The duration threshold does involve a tradeoff: while noise is eliminated, the user is prevented from analyzing a file that contains notes that are less than 100 time instants long. Any such note would be treated as noise. As such, the program may be unable to correctly handle very fast rhythms in low quality signals.

5.1.2 Identifying Notes

Our next module will basically identify the note. It takes a frequency as an input and it outputs a string with the name of the note being played. We have created a library with different frequency ranges that correspond to different notes. The module will find the range to which the input frequency corresponds and

¹This content is available online at <http://cnx.org/content/m41934/1.1/>.

input output the string for that range. This process will be done for every note that is given by the `get_note` module.

This identification process would not be possible if we only had a regular spectrogram, since the precision is very low, causing a very high error that would lead to wrong identification of a note. The frequency reassignment process is an extremely essential part of this method.

5.1.3 Generating Synthetic Audio Files

As an extra module, we have created a simple function to create tones, in order to test our project. In this function, we have different square waves with different frequencies. We simply create a vector with the notes that we want and we have a song to be analyzed.

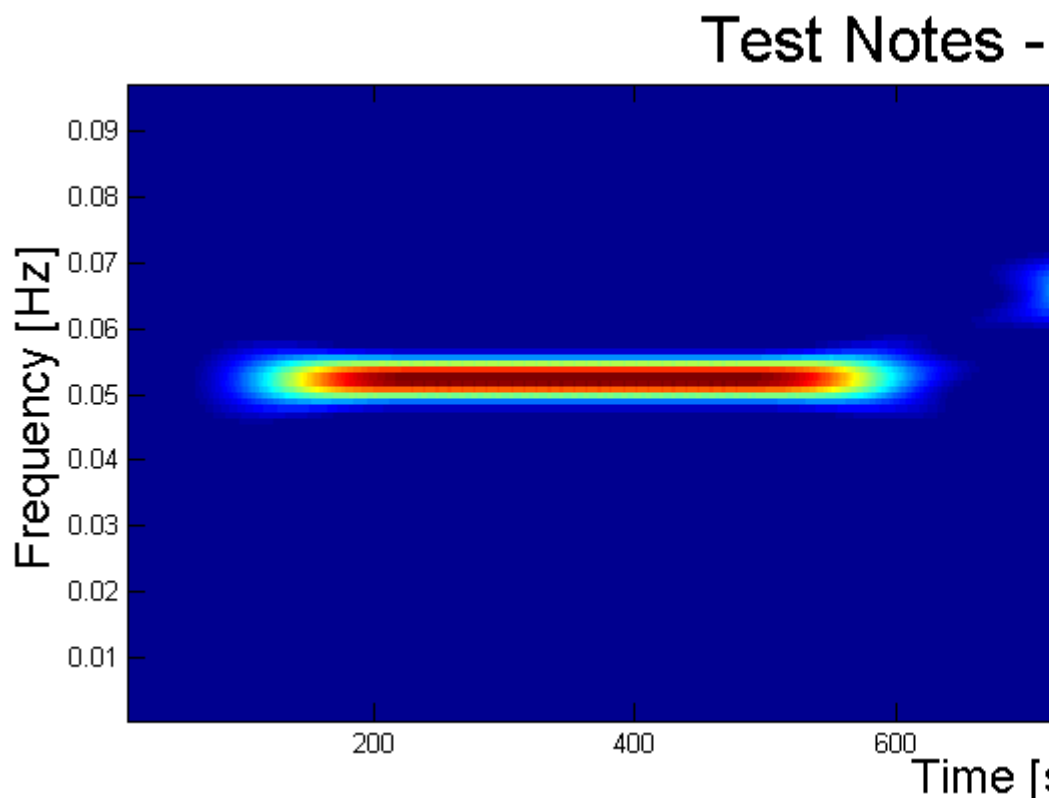
Chapter 6

Simple Input¹

6.1 Base Input

We developed the music transcription system by analyzing a simple input of a “C5” followed by an “E5.” The letter corresponds to the musical note, and the number corresponds to the octave (a “C4” is lower than a “C5” and so forth; a “C4” is “Middle C”). The input is comprised purely of tonal components, and no chords are used

6.1.1 Traditional Spectrogram



``,height!`,height!`

¹This content is available online at <http://cnx.org/content/m41964/1.1/>.

Fig 1 The traditional ‘‘smeared’’ spectrogram of the base input

6.1.2 Reassigned Spectrogram

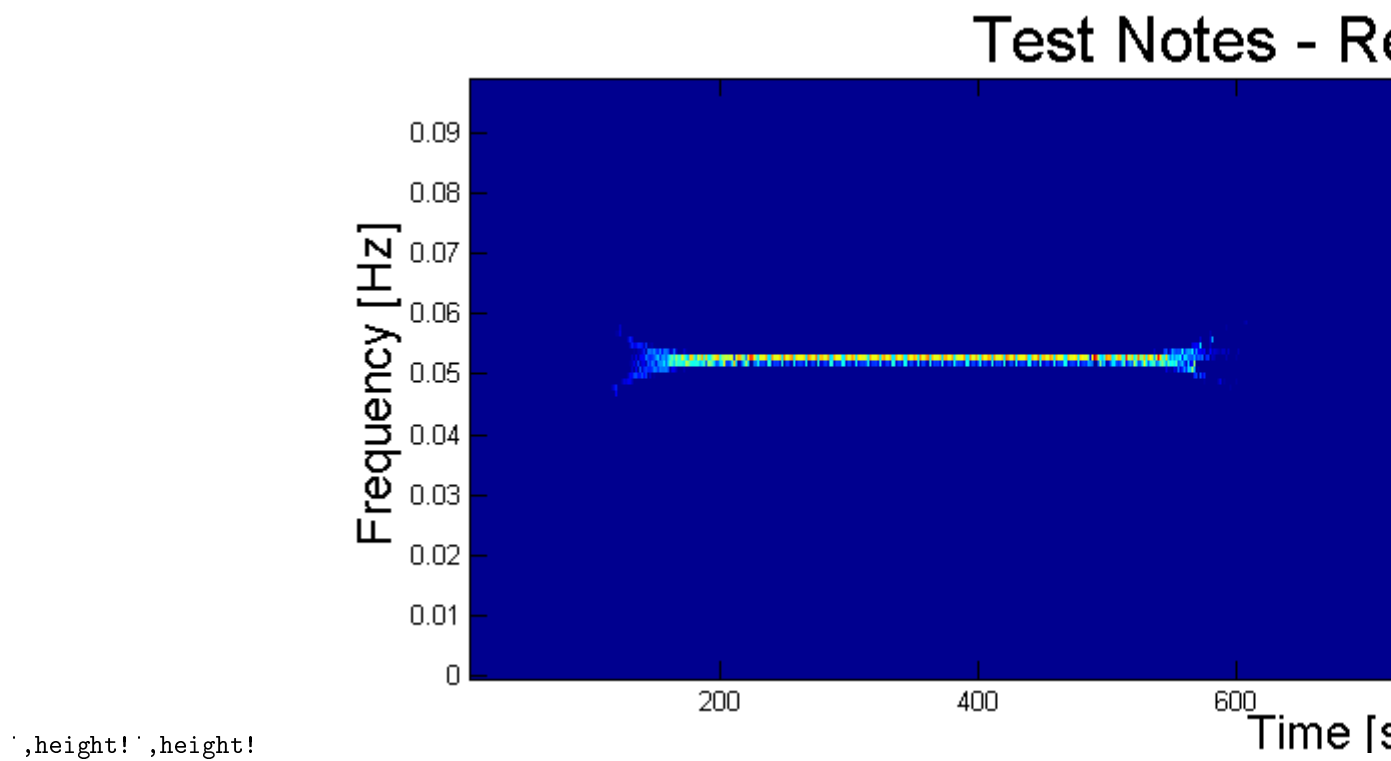


Fig 2 The higher-resolution ‘‘Reassigned’’ spectrogram of the same input

6.1.3 System Output

Ultimately, our system correctly identified the two notes being played as a ‘C5’ followed by

```
EDU>> treblemaker(song,1024)

notes =

    'C5 E5 ' % m41964;base result.png;;;6.0;8.5;
```

Fig 3 MATLAB prints out the correct result

Chapter 7

Complex Input¹

7.1 Complex Input

Once the program was successfully developed, it was run on a recognizable tune: the first four m

7.1.1 Traditional Spectrogram

¹This content is available online at <<http://cnx.org/content/m41890/1.1/>>.

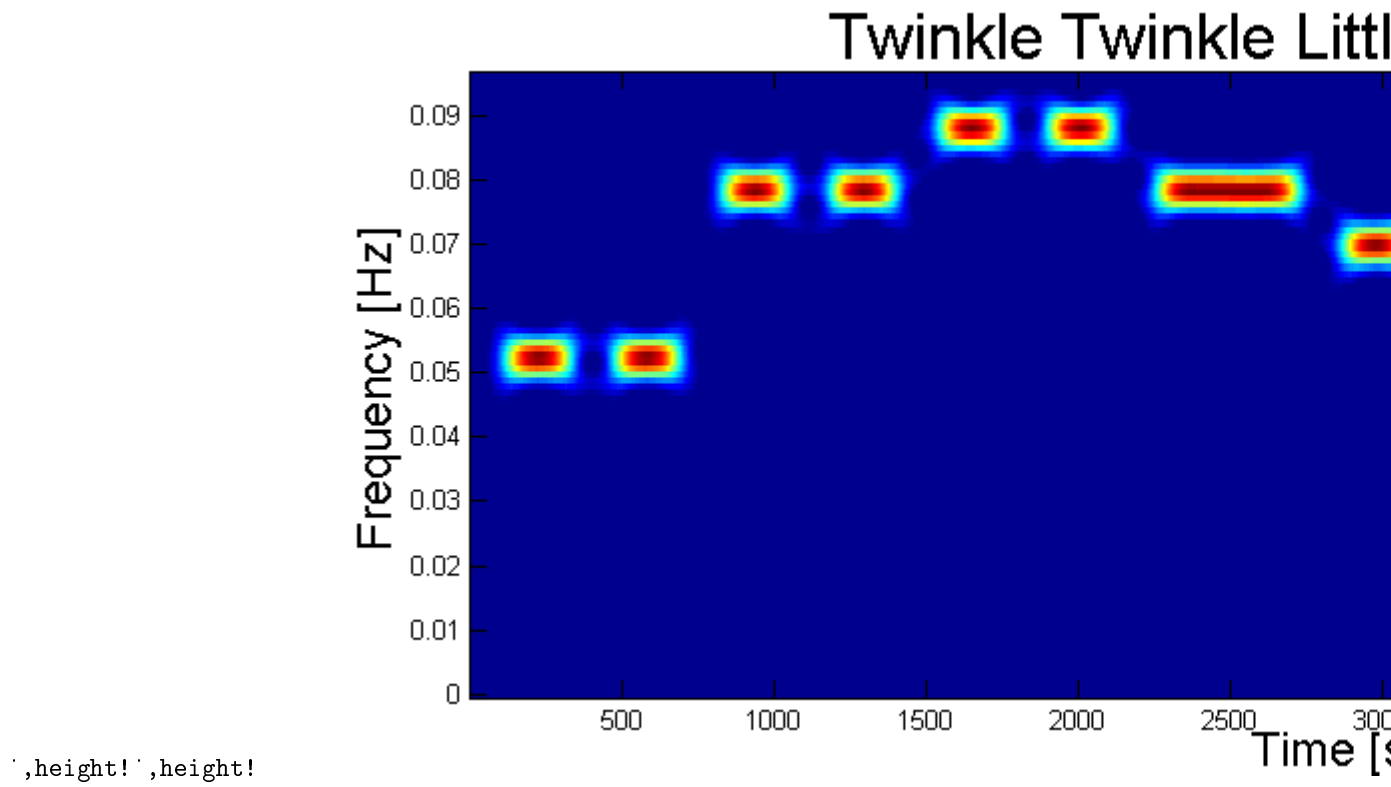


Fig 1 The traditional ‘‘smeared’’ spectrogram of ‘‘Twinkle Twinkle Little Star’’

7.1.2 Reassigned Spectrogram

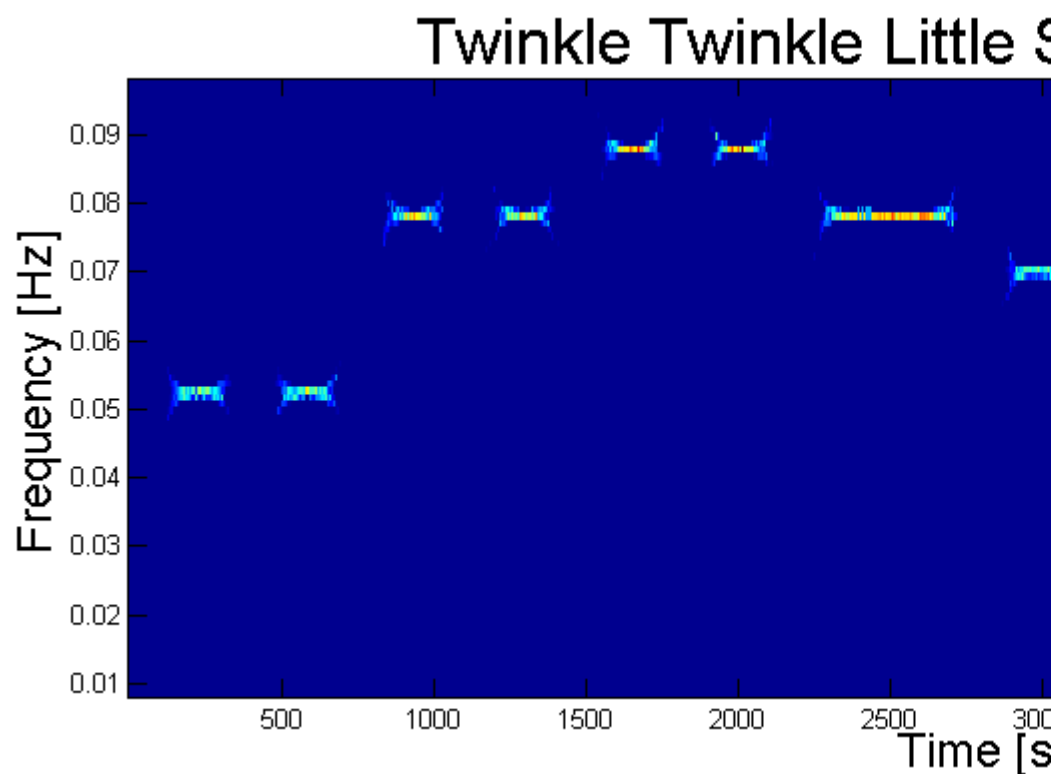


Fig 1 The reassigned spectrogram of “Twinkle Twinkle Little Star”

7.1.3 System Output

Again, the system correctly identified all of the notes of the input signal without error. The

Here is the MATLAB output:

```
EDU>> treblemaker(song,1024)

notes =

    'C5 C5 G5 G5 A5 A5 G5 F5 F5 E5 E5 D5 D5 C5 '
```

Fig 3 The MATLAB output accurately identifying each note

Chapter 8

Results¹

8.1 Results

Ultimately, we succeeded in creating a foundational music transcription system. The system succe

Tonal inputs (as opposed to a hand clap, a drum hit, etc.) Single notes (as opposed to chor

Other limitations include:

Size of the MATLAB signal vector (tradeoff between signal length and signal quality)

¹This content is available online at <<http://cnx.org/content/m41883/1.1/>>.

Additionally, the system does not currently print out the duration of the note. However, during the development of the system, the possibility of improving upon these limitations are expanded on in the Conclusions section.

8.1.1 Tradeoff between signal length and quality

The current method for calculating reassigned time-frequency representation of a signal is comp

Chapter 9

Conclusions¹

9.1 Conclusions

After completing the project, we concluded the following:

Frequency reassignment is necessary for music transcription There is significant room to improve

Ultimately, it's clear that this music transcription technology has great potential for expansion

¹This content is available online at <<http://cnx.org/content/m41891/1.1/>>.

9.1.1 The necessity of Frequency Reassignment

After attempting music transcription with both reassigned time-frequency representations of signals,

9.1.2 Expanding on the Foundation

If this project were expanded, the next steps would include expanding the range of inputs that

Additionally, provision could be made for recorded signals by either improving the computation

Attributions

Collection: *Music Transcription Through Frequency Reassignment*

Edited by: Ryan Koehn, Matt Lopez

URL: <http://cnx.org/content/col11393/1.1/>

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Introduction"

By: Matt Lopez

URL: <http://cnx.org/content/m41893/1.1/>

Page: 1

Copyright: Matt Lopez

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Background"

By: Matt Lopez

URL: <http://cnx.org/content/m41892/1.1/>

Page: 3

Copyright: Matt Lopez

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Frequency Reassignment"

By: Matt Lopez

URL: <http://cnx.org/content/m41894/1.1/>

Pages: 5-6

Copyright: Matt Lopez

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Preparing the Signal"

By: Ryan Koehn

URL: <http://cnx.org/content/m41960/1.1/>

Pages: 7-8

Copyright: Ryan Koehn

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Analyzing the Signal"

By: Ryan Koehn

URL: <http://cnx.org/content/m41934/1.1/>

Pages: 9-10

Copyright: Ryan Koehn

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Simple Input"

By: Ryan Koehn

URL: <http://cnx.org/content/m41964/1.1/>

Pages: 11-13

Copyright: Ryan Koehn

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Complex Input"

By: Ryan Koehn

URL: <http://cnx.org/content/m41890/1.1/>

Pages: 15-18

Copyright: Ryan Koehn

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Results"

By: Ryan Koehn

URL: <http://cnx.org/content/m41883/1.1/>

Pages: 19-20

Copyright: Ryan Koehn

License: <http://creativecommons.org/licenses/by/3.0/>

Module: "Conclusions"

By: Ryan Koehn

URL: <http://cnx.org/content/m41891/1.1/>

Pages: 21-22

Copyright: Ryan Koehn

License: <http://creativecommons.org/licenses/by/3.0/>

Music Transcription Through Frequency Reassignment

This project involves implementing a basic music transcription system through the use of the rea

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course material.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges

