Theory Informed Automatic Music Transcription Final Report

Liam Caveney V00721857

Lisa Kinoshita V00799002

YiCong Gao V00785753

ABSTRACT

Polyphonic transcription is a challenging problem in music information retrieval. This paper presents the idea of using the rules of music harmony to inform the transcription process. This process is broken down into two stages. First, melody and bass are transcribed using FFT, and rhythm are determined by Bayesian model. The output will be a musicXML, and we will turn it into a midi file. Given these two voices, our inner voices algorithm uses a graph trained by the rules of harmony to infer the inner voices. This graph weighted, to select the most probable solution. The first series of this algorithm used testing data that was a subset of training data and produced the correct inner voices 60% if the time. The second series of tests, where the testing data was excluded from training data, produced an incorrect solution that obeyed the rules of harmony 30% of the time.

1. INTRODUCTION

1.1 Background

Automatic Music Transcription (AMT) is the process of turning a recording or live signal into musical notation. AMT draws on many Music Information Retrieval (MIR) techniques and has relevant applications including annotating live improvised music or oral music for which no score exists [1]. AMT has been explored extensively by researchers of the MIR community. However, the problem of polyphonic transcription, where multiple pitches sound simultaneously, is still a challenging task and remains unsolved [2]. Additionally, utilizing predefined parameters, such as tempo, as a tool to improve AMT has not been greatly explored [3].

Polyphonic transcription, as previously mentioned, is a much harder problem to solve than monophonic transcription, one pitch at a time, due to the overlapping partials [4]. When an instrument sounds, it produces several frequencies that are integer multiples of the fundamental note it is playing. These additional frequencies are called partials, or harmonics. When looking at a frequency spectrum of a polyphonic signal, the harmonics create the issue of determining which frequencies are the sounding notes and which are the resulting harmonics. Several techniques have been developed in attempt to solve this problem. One approach is to use music signal processing techniques such as a smoothing autoregressive model to address overlapping harmonics [5]. Another is to select a set of candidates, generate all combinations of these candidates and then select the best combination based on harmonic amplitudes and spectral smoothness [6]. Furthermore, another approach is to utilize spectrogram factorization techniques to take advantage of the redundancies in music spectrograms [7].

A common form of classical music composition is the four part chorale. The form consists of four voices, which, in order from high to low pitch, are: soprano (or melody), alto, tenor, and bass, as seen in [8, Fig. 1]. Throughout the development of polyphonic music composition, a set of rules were developed that dictate how a voice is allowed to move with respect to the other voices. These rules are known as voice leading and, if followed, result in acoustically pleasing results. Seeing as polyphonic ATM is limited, the incorporation of these rules into software may help produce more accurate and complex transcriptions.

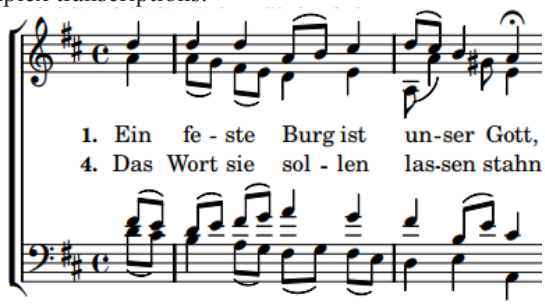

Figure 1. SATB from *Ein feste Burg ist unser Gott*, chorale harmonization by J.S. Bach

1.2 Objective

There are tremendous amounts of research in the area of AMT. Our goal is to transcribe the MIDI data obtained from a signal into music scores. Creating a system that uses available tools to produce a list of notes that are potentially sounding in a given frame. From this list we hope to deduce a melody and bass line as well as considering notes for the inner voices. Using the rules of music theory and harmony, the system will then logically fill in the holes that were not determined in the previous steps.

1.3 Previous Challenges in AMT

There are significant amount of challenges in this field which remain unsolved. For example, L. Su et al. [9] used multi-pitch estimation (MPE) to transcribe choral and symphonic music. They conducted time-frequency analysis and created a concentration of time-frequency to track multiple pitches in a single piece. Though the result of the evaluation is satisfactory for Bach chorales, the evaluation of a romantic symphony was not. Since romantic music tends to have a larger number of chromatic non-chord tones, and their structure and compositional techniques are more advanced and

complicated. Y. Ojima et al. [10] built a hierarchical model to solve MPE and the AMT problem. They combined the spectrum model with the language model to eliminate the errors in musical grammar. The language model they used is based on Generative Theory of Tonal Music (GTTM). However, the key of a piece was not taken into consideration in their model, thus the accuracy of the model is limited.

1.4 Algorithms

The algorithms that have been used to solve challenges in the AMT field have become closer to those in the field of machine learning (e.g. unsupervised learning)[9], [10].

1.4.1 Key Detection

Detecting key profiles is crucial in tonal music. D. Hu et al.[11] have built an unsupervised learning model to accomplish the task of Automatic Key-finding. Hu used an algorithm that is based on Latent Dirichlet Allocation (LDA). LDA is model that collects vast amounts of text documents. Each document has a mixture of topics, and each word within every topic are considered as attributes. Similar topics tend to have similar attributes. The model treats each piece as a "document" and the notes in each rhythmic segment as a "word". Pieces in same keys tend to have similar sets of pitches.

An alternation approach is also mention in Hu's paper. The algorithm is introduced by Krumhansl and Kessler: Create a vector that contains 24-key profiles (12 major and 12 minor keys). First, collect the pitch class information of a piece. Second, compute the key profile that has the highest correlation with this collection(vector) [11].

1.5 Related Works

Many works of transcribing classical music (from Baroque to Romantic era) and vocal music have been studied in the AMT field. R. Nishikimi et al. [12] introduce a fundamental trajectory algorithm that estimates the the soprano and the bass from polyphony by extracting the most predominant harmonic components. They use YIN, a correlation based method for frequency estimation, to decrease the error rate. The evaluation of results shows that Nishikimi's method of sub-harmonic summation joined with HMM is more accurate than majority-vote method. Majority-vote method is a method that corrects the estimation result with the most frequently occurence output. An existing application of majority-vote method is implemented and available online [13].

2. TIMELINE AND GOALS

2.1 Project Specification Timeline and Goals

2.1.1 Monophonic Transcription

The first objective is to create a working transcription system that receives an audio file as input, segments the file into frames, converts the frame into a frequency spectrum, determines which pitch is sounding and converts that pitch to midi.
Completion date: October 27th, 2017.

2.1.2 *Melody and Bass*

The next objective is to modify the system to detect two voices from the frequency spectrum. This will be done from further research and implementing techniques discussed in the introduction.
Completion date: November 3rd, 2017.

2.1.3 *Infer Inner Voices, simple input*

At this point it is unclear whether it is best to attempt to draw any further voices from the frequency spectrum or to move straight to a musical analysis on the melody and bass lines. Following analysis, the system will be able to determine likely pitches for the inner voices, creating a complete transcription. Note, this inference may not always prove accurate and may have to make decisions without guaranteeing perfect accuracy. The spectral analysis will be simpler using simple input, or pitches all created by sine waves. Then we will test using midi generated audio files.
Completion date: November 10th, 2017.

2.1.4 *Infer Inner Voices, real instrument input*

The final test is to see if our system can work on recordings of real instruments. Using real instruments gives less predictability in the frequency spectrum so this may require further acoustical study.
Completion date: November 17th, 2017.

2.1.5 *Score Output*

After generating MIDI from the transcription, the final task is to generate a score. Our aim is to create a score that provides the correct enharmonic version of a note. We will also have to determine key and time signatures.
Completion date: November 24th, 2017

2.2 Progress Report Modified Timeline and Goals

Our stretch goal is to have a working score output system that will use the outer voice detection algorithms trained by using the input of the same Bach chorals which we used to train the inner voice algorithm. Since, we will use Bach chorals for testing, we believe that this will provide the most accurate result. Our expected outcome would be to use simple rules to eliminate the outer voice. For example, in this case, we can use the rules of four part writing of harmony to strictly limit the outcome of our AMT algorithm of the outer voice. In the worst case scenario, we should be able to output a score version of the input audio with the accuracy of the result being compromised using tools we have mentioned. Our modified timeline is as follows.

Task	Completion Date
Rhythm tracking	November 20th, 2017
Testing traversal	November 20th, 2017

algorithm	
Score transcription	November 24th, 2017
Score output using outer voice detection	December 7th, 2017

3. TOOLS AND DATASETS

3.1 MIDI

MIDI (Musical Instrument Digital Interface) will be used throughout our implementation for its ease of manipulation and modification of music files [14].

3.2 Music21

Music21 is an object oriented toolkit for transforming music in symbolic forms. Among many other functions, Music21 can generate score notation and graph statistical data in music (e.g. pitch class, density and duration [15]).

3.3 DFT

SciPy library will be used in our program to conduct Discrete Fourier transform and other digital signal processing [16].

3.4 music21.corpus.chorales.Iterator()

Music21 provides a built in dataset of 347 bach chorales [17]. The iterator is an easy way to access this dataset and is used in training and testing our system.

3.5 LilyPond

LilyPond is a free music engraving program which creates sheet music following traditions of classical music engraving [18].

4. APPLICATIONS

4.1 Music Transcription Software

Exploring AMT as a multidisciplinary task, such as in our case with musicology and computing, can not only provide further progress in the research of AMT but also broaden the applications of AMT technology. One application of AMT that has gained popularity in recent years is the online music transcription software, such as ScoreCloud [19], that is able to transcribe music uploaded by the user. Softwares such as ScoreCloud can greatly improve with AMT technology that could effectively transcribe a variety of music rather than just monophonic transcription.

4.2 Automatic Piano Reduction of Orchestral Score

Despite the many instruments present in an orchestra, orchestral writing can often be compressed down into four part writing as seen in Figure 1. There has always been the desire to reduce orchestral scores into music playable on one piano, for example, ballet music was often reduced to one or two piano parts so dancers could rehearse without the entire orchestra. With the focus on four part writing, there is potential for this project to be used to create piano reductions from orchestral audio recordings.

5. ROLES

All members	Monophonic transcription, score output, documentation
Liam	Inner voice algorithm
Lisa	Melody and bass, simple input
Yi	Melody and bass, real instrument input

6. TRANSCRIPTION PROCESS

In this section, we will be discussing our process for polyphonic transcription. This process is based on two steps: extracting the melody and bass voice, and inferring inner voices using an inner voice algorithm.

6.1 Melody and Bass Transcription

Our approach to pitch extraction and rhythm tracking (analysis) can be broken down into small steps. First, we cleaned our metadata by using Audacity and converted the .wav file into an array in Python by using Scipy. Second, we divided the array into smaller chunks. The buffer size we chose to use is the standard 44.1kHz. Then, we applied FFT to each window of the original audio file. The result of FFT gave us the frequency appearance in each frequency range. We then mapped the frequencies into midi note. Lastly, to figure out what the duration of each note, so we need to do a rhythm analysis, and we have to turn the midi note and duration we got into musical score.

6.1.1 Pitch extraction and read in audio

As we briefly described above, we first had to clean the metadata in Audacity. This step is necessary since we want to extract the audio without errors. To do this, we used the metadata editor and deleted all the metadata. To read the file in Python, we used the function scipy.io.wavfile.read() from the Scipy library which reads the file into an array, where index 0 is the sampling rate and index 1 is the audio file. After this step, the audio file is ready to be processed in Python.

6.1.2 Buffering and FFT

For buffering, we used scipy.signal.hann() from the SciPy library. This gave us the windows without any gaps between them. Then we applied FFT to each window to extract the frequency values.

6.1.3 Mapping frequency to midi note

To map the frequency values to midi notes, we used the frequency to midi equation shown below:

$$\text{MIDI} = 69 + 12\log_2\left(\frac{f}{440Hz}\right) \quad (1)$$

We then mapped our values to the closest midi number by using the frequency/midi chart shown below:

MIDI SPN	frequency (Hz)	MIDI SPN	frequency (Hz)	MIDI SPN	frequency (Hz)	MIDI SPN	frequency (I
21 A0	27.5	49 C#3	138.6	77 F5	698.5	105 A7	3520
22 A#0	29.1	50 D3	146.8	78 F#5	740	106 A#7	3729.3
23 B0	30.9	51 D#3	155.6	79 G5	784	107 B7	3951.1
24 C1	32.7	52 E3	164.8	80 G#5	830.6	108 C8	4186
25 C#1	34.6	53 F3	174.6	81 A5	880		
26 D1	36.7	54 F#3	185	82 A#5	932.3		
27 D#1	38.9	55 G3	196	83 B5	987.8		
28 E1	41.2	56 G#3	207.7	84 C6	1046.5		
29 F1	43.7	57 A3	220	85 C#6	1108.7		
30 F#1	46.2	58 A#3	233.1	86 D6	1174.7		
31 G1	49	59 B3	246.9	87 D#6	1244.5		
32 G#1	51.9	60 C4	261.6	88 E6	1318.5		
33 A1	55	61 C#4	277.2	89 F6	1396.9		
34 A#1	58.3	62 D4	293.7	90 F#6	1480		
35 B1	61.7	63 D#4	311.1	91 G6	1568		
36 C2	65.4	64 E4	329.6	92 G#6	1661.2		
37 C#2	69.3	65 F4	349.2	93 A6	1760		
38 D2	73.4	66 F#4	370	94 A#6	1864.7		
39 D#2	77.8	67 G4	392	95 B6	1975.5		
40 E2	82.4	68 G#4	415.3	96 C7	2093		
41 F2	87.3	69 A4	440	97 C#7	2217.5		
42 F#2	92.5	70 A#4	466.2	98 D7	2349.3		
43 G2	98	71 B4	493.9	99 D#7	2489		
44 G#2	103.8	72 C5	523.3	100 E7	2637		
45 A2	110	73 C#5	554.4	101 F7	2793.8		
46 A#2	116.5	74 D5	587.3	102 F#7	2960		
47 B2	123.5	75 D#5	622.3	103 G7	3136		
48 C3	130.8	76 E5	659.3	104 G#7	3322.4		

Figure 2. Frequency/MIDI mapping chart

6.2 Inner Voices Algorithm

Once a melody and bass line have been detected, there is still the issue of determining inner voices. As discussed, polyphonic pitch detection is limited, so we decided to approach the problem from a machine learning perspective. Since four part harmony has rules as to which chords and voicings can follow a certain chord, we needed a way to encode these rules so they can be applied to the melody and bass we have already determined. In order to solve this problem, we created a graph with weighted, directed edges. Each node in the graph represents a possible combination of SATB and each outgoing edge leads to a potential next combination. The weight of each edge represents the probability of that edge being used. In order to solve the inner voices, we simply need to traverse the graph, accepting paths that satisfy the predetermined Soprano and Bass lines. If there are multiple paths, we can select the path with the largest sum of weights as the most likely inner voices.

6.2.1 Create_graph.py

In order to train our graph, we used midi files of chorales from BWV 1-197 [20]. The algorithm for generating our graph is as follows:

```
for each chorale:
  transpose to C major or A minor
  get sequence of chords
  for each chord in sequence:
    id = s + a*127 + t*127*127 + b*127*127*127
    if id not in graph:
      create node(id)
  for each chord in sequence:
    if edge from curr to next chord does not exist:
      create edge
```

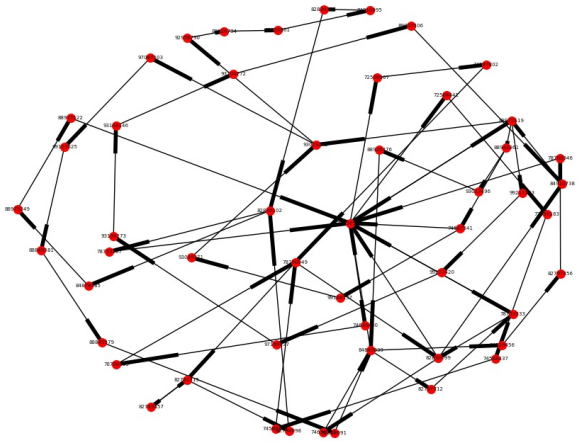

Figure 3. Graph created from one chorale as training data

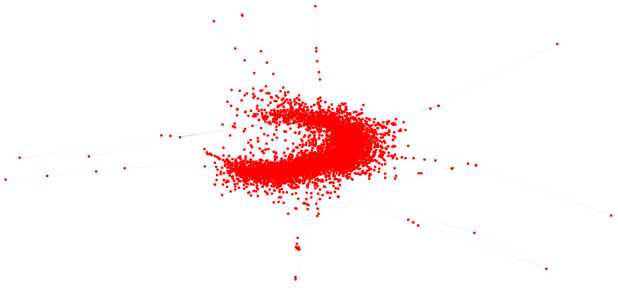

Figure 4. Graph from 347 chorales used as training data

One issue that arose was how to give each node a unique id. The solution we used was to use a number of base 127 converted to an integer. Base 127 is used because there are 127 midi values. This way we can easily determine the four notes present by converting the id back to base 127 and seeing the values at each place. Although the range of each voice is far less than all possible 127 midi values, we kept with 127 because, $127^4 - 1$is still small enough to be represented by a python integer and if this algorithm were to be put to use with other instruments there would be no need to convert ranges of instruments.

$B = C4, T = E4, A = G4, S = C5$
$As\ midi: [48, 52, 55, 60]$
$id = 60 * 127^0 + 55 * 127^1 + 52 * 127^2 + 48 * 127^3$

Figure 5. Converting chord to id using base 127

$id = 99168137$
$99168137\ /127 = 780851\ R\ 60 \qquad S = 60$
$780851/127 = 6148\ R\ 55 \qquad T = 55$
$6148/127 = 48\ R\ 52 \qquad A = 52$
$48/127 = 0\ R\ 48 \qquad S = 67$

Figure 6. Getting SATB midi values from id

2.2.2 Gen_inner.py

Now that we have a graph that when traversed follows the rules of four part harmony, we need to discover a path

that obeys a given midi file containing a soprano and bass line. The traversal algorithm is as follows:

```
traverse(chord, c, i, g):
  queue = [(chord, [[chord.id, 0]])]
    while queue:
      (node, path) = queue.pop(0)
      for n in node.edges:
        if len(path)<len(c):
          if n.s == c[len(path)].s and n.b= = c[len(path)].b
            if len(path)+1 == len(c):
              yield path + [next]
            else:
              queue.append((n, path + [next]))
```

curr_c is current node, c is an array containing the provided soprano and bass lines, i is the index of curr_c in relation to c, g is the graph. p is an array of the ids of potential next nodes and l is the list of paths to return.

7. RESULTS

7.1 Melody and Bass Transcription

The accuracy of melody and bass transcription still needs improvement, however we were able to produce score output of melody and bass lines. Figure 7 shows the expected output for a chorale, and Figure 9 shows the output from our transcription.

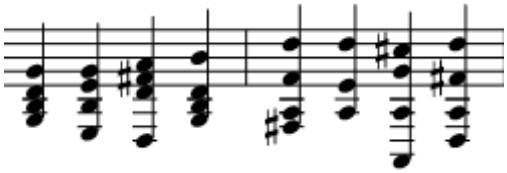

Figure 7. Desired output of J.S. Bach 371 Four Part Chorales Volume I - Number 9

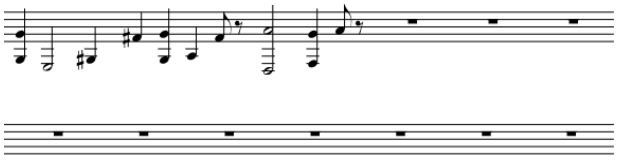

Figure 8. Our melody and bass lines output of Chorale Number 9

In addition, we were able to output the melody and bass lines into a midi file to be used for the inner voice algorithm. We used the music21 function, music21.translate.streamToMidiFile(), to convert the music stream into a midi file. Figure 9 shows an example of this conversion.

```
<MidiEvent DeltaTime, t=0, track=1, channel=1>
<MidiEvent SEQUENCE_TRACK_NAME, t=0, track=1, channel=1, data=''>
<MidiEvent DeltaTime, t=0, track=1, channel=1>
<MidiEvent PITCH_BEND, t=0, track=1, channel=1, _parameter1=0, _parameter2=64>
<MidiEvent DeltaTime, t=1024, track=1, channel=1>
<MidiEvent NOTE_ON, t=0, track=1, channel=1, pitch=68, velocity=90>
<MidiEvent DeltaTime, t=512, track=1, channel=1>
<MidiEvent NOTE_OFF, t=0, track=1, channel=1, pitch=68, velocity=0>
<MidiEvent DeltaTime, t=0, track=1, channel=1>
<MidiEvent NOTE_ON, t=0, track=1, channel=1, pitch=68, velocity=90>
<MidiEvent DeltaTime, t=512, track=1, channel=1>
<MidiEvent NOTE_OFF, t=0, track=1, channel=1, pitch=68, velocity=0>
<MidiEvent DeltaTime, t=0, track=1, channel=1>
<MidiEvent NOTE_ON, t=0, track=1, channel=1, pitch=68, velocity=90>
```

Figure 9. The printed MIDI output in Python

7.2 Inner Voices Algorithm

The first set of tests conducted used 347 chorales from music21's built in corpus.chorales.Iterator() to train the graph. 10 chorales from these 347 were then used to test the algorithm. Only the first 25 chords were considered as the run time was very long. We felt 25 chords was long enough to distinguish a chorale from others. Upon completion, the inner voices produced were compared to the chorales correct in voices. The results are as follows:

Chorale	Possible paths	Inner voices correct?
0	2	yes
1	1	yes
2	1	yes
3	7	no
4	2	yes
5	2	no
6	2	no
7	10	no
8	1	yes
9	1	yes

Table 1. Results from 10 chorales in training data

The algorithm produced the correct inner voices 60% of the time, however, the correct voices were produced only 33% of the time multiple paths were possible. This suggests that given more data, the performance would suffer.

For the next set of tests, 10 chorales were removed from the original 347. The graph was then trained on the remaining 337 chorales and tested using the 10 removed chorales. The results are as follows:

Chorale	Possible paths	Inner voices correct?
0	0	-
1	0	-
2	1	no
3	0	-
4	0	-
5	0	-
6	0	-
7	10	no
8	0	-
9	1	no

Table 2. Results from 10 chorales outside training data

As expected, the performance is significantly worse than the first set of tests. Only 30% of tests produced inner voices, all incorrect when compared to the truth. However, the inner voices from the three tests that produced results are acoustically pleasing and clearly obey the rules of harmony. This suggests that with more chorales used to train the graph, more paths would be produced. Even though there is a large chance the inner voices would be incorrect, there would still be some value in producing harmonically correct results.

8. FUTURE WORK

8.1 Improvements to Inner Voices Algorithm

As indicated before, a larger dataset would improve the results of our algorithm when testing data is not included in training data. A possible method to further improve the selection of a correct path would be to use chord detection. That was the path would be selected based not only on matching soprano and bass voices, but also on harmony. Also, nonchord tones are the most complicated part of Bach chorales. If a method of filtering out nonchord tones was implemented before the training of the graph, the graph would be simpler and more closely match the rules of harmony. This could potentially yield better results.

8.2 Improvements to Melody and Bass Transcription

To improve the melody and bass transcription, training the algorithm with the same data set that we used to train the inner voice algorithm may be an approach to consider. On the other hand, since Bach chorales are not completely homorhythmic, the rhythm Bayesian model also need modifications.

9. CONCLUSION

Our melody and bass transcription need a lot of improvement in order to get an acceptable result. A difficulty that we may face will be the separation the notes into treble and bass clef, so we can put soprano and alto voice on the treble and tenor and bass on the bass clef.

The inner voices algorithm described in this paper produced some promising results. Using a subset of training data as testing data produced correct results 60% of the time, while using testing data separate from training data found an incorrect, although suitable, solution 30% of the time. These results would be improved with the addition of chord detection and a larger set of chorales as training data.

Polyphonic transcription remains a difficult problem in music information retrieval. The approach outlined in this paper is as follows: (1) Transcribe Melody and Bass voices, (2) Use those voices and a graph trained with the rules of harmony to solve for inner voices. We believe that using the rules of harmony has great potential in improving future polyphonic transcription algorithms.

10. REFERENCES

[1] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff and A. Klapuri, "Automatic music transcription: challenges and future directions", *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407-434, 2013.

[2] E. Benetos and S. Dixon, "Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model", *The Journal of the Acoustical Society of America*, vol. 133, no. 3, pp. 1727-1741, 2013.

[3] D. Cazau, G. Revillon, J. Krywyk and O. Adam, "An investigation of prior knowledge in Automatic Music Transcription systems", *The Journal of the Acoustical Society of America*, vol. 138, no. 4, pp. 2561-2573, 2015.

[4] Y. E. H., Chunghsin, "Multiple fundamental frequency estimation of polyphonic recordings" Ph. D. dissertation, University Paris 6, 2008.

[5] V. Emiya, R. Badeau and B. David, "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643-1654, 2010.

[6] A. Pertusa,, and J.Inesta, "Multiple fundamental frequency estimation using Gaussian smoothness." *Acoustics, Speech and Signal Processing*, 2008.

[7] E. Vincent, N. Bertin and R. Badeau, "Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 528-537, 2010.

[8] J.S. Bach, Ein feste Burg ist unser Gott, BWV 80. Ed. Georg L. Sothilander. http://ks.imslp.net/files/imglnks/usimg/8/86/IMSLP464380-PMLP149581-Luther_Bach-

Ein_feste_Burg_ist_unser_Gott-Chorale-EG362-BWV80-2sys.pdf

[9] L. Su, T. Chuang, Y. Yang, "Exploiting Frequency, Periodicity and Harmonicity Using Advanced Time-Frequency Concentration Techniques for Multipitch Estimation of Choir and Symphony", *ISMIR,* pp. 393-399, 2016.

[10] Y. Ojima, E. Nakamura, K. Itoyama, K. Yoshii, "A Hierarchical Bayesian Model of Chords, Pitches, and Spectrograms for Multipitch Analysis", *ISMIR,* pp. 309-315, 2016.

[11] D. Hu, L. and K. Saul, "A Probabilistic Topic Model for Unsupervised Learning of Musical Key-Profiles," *ISMIR,* pp. 441-446, 2009.

[12] R. Nishikimi, E. Nakamura, K. Itoyama, K. Yoshii, "Musical Note Estimation for F0 Trajectories of Singing Voices Based on a Bayesian Semi-Beat-Synchronous HMM," *ISMIR*, pp.461-467, 2016

[13] *songle.jp.* [Online]. Available: http://songle.jp [Accessed 14-Oct-2017].

[14] *Midi.org*. [Online]. Available: http://www.midi.org. [Accessed: 12- Oct- 2017].

[15] M. S. Cuthbert, and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data." *11th International Society for Music Information Retrieval Conference*, pp. 637-642, 2010.

[16] *Scipy.org*, 2017. [Online]. Available: https://www.scipy.org. [Accessed: 14- Oct- 2017].

[17] *music21.corpus.chorales.* [Online]. Available: http://web.mit.edu/music21/doc/moduleReference/moduleCorpusChorales.html. [Accessed: 10- Dec - 2017]

[18] *LilyPond-Music notation for everyone*. [Online]. Available: http://lilypond.org [Accessed 14-Nov-2017].

[19] *ScoreCloud.com*. [Online]. Available: http://scorecloud.com [Accessed: 14- Oct- 2017].

[20] J.S. Bach, Chorales from Cantatas BWV 1-197, Jürgen Knuth, Werner Icking Music Collection. http://imslp.org/wiki/389_Choralges%C3%A4nge_(Bach%2C_Johann_Sebastian)#MIDI_files