# Overtone filtration techniques for music spectrum analysis (COMP4121)

There exists many tools for spectral analysis of audio using algorithms such as the Fast Fourier transform and the discrete cosine transform, yet they are not very accurate when used to transcribe notes from an audio input.

This paper explores two approachs to supplementing these spectral analytic methods, by improve their results. In particular, we explore methods involving markov chains and probabilistic filtering of overtones.

# Contents


**1 Previous work and acknowledgments** **3**
- 1.1 Attribution of source code . . . 3
- 1.2 DFT . . . 3
- 1.3 Converting frequencies into notes . . . 3
- 1.4 Window functions . . . 4

**2 Problems with naive DCT based transcriptions** **4**
- 2.1 Harmonic and Inharmonic interference . . . 4
- 2.2 Inaccuracy when multiple notes are played . . . 5

**3 A digression into why modelling as a stochastic process would be ineffective** **5**

**4 An iterative method for filtering out interfering overtones** **6**
- 4.1 Aim of method . . . 6
- 4.2 Overtone Distributions . . . 7
- 4.3 How do we find $O$? . . . 7
- 4.4 Noise . . . 8
- 4.5 Initial distribution . . . 9
- 4.6 Approximators . . . 9
  - 4.6.1 Echo approximator . . . 9
  - 4.6.2 Average of history . . . 9

**5 Results and further improvement** **9**
- 5.1 Results . . . 9
- 5.2 Ideas for further improvement . . . 9

**Appendices** **10**

**A Transcriptions** **10**
- A.1 The Well-Tempered Clavier (pedal) . . . 10
  - A.1.1 With our augmentations . . . 10
  - A.1.2 Without our augmentations . . . 11
- A.2 The Well-Tempered Clavier (no-pedal) . . . 12
  - A.2.1 With our augmentations . . . 12
  - A.2.2 Without our augmentations . . . 14
- A.3 Polyphonic . . . 15
  - A.3.1 With our augmentations . . . 15
  - A.3.2 Without our augmentations . . . 16

**6 Bibliography** **17**

# 1 Previous work and acknowledgments

## 1.1 Attribution of source code

The work in this paper builds upon already existing work on the effectiveness of spectral analysis techniques for music transcription, specifically that of a previous taker of this course (Gregory Omelaenko). When constructing experiments to guage the effectiveness of the techniques described in this paper, source code originally written by the author of the reference paper, but heavily modified (ported to a different language, different execution environment and to use different libraries) for the purpose of this paper was reused.

The code that was reused includes the implementations of the DCT/FFT algorithms, conversion from frequencies to notes, the window functions, the overlapping of samples and the code that combines these into an output result. The rest of this section skims over the sections of said paper of which are particularly relevant to this one, for the sake of locality.

## 1.2 DFT

The Discrete Fourier Transform (henceforth referred to as the DFT, or FFT) is an algorithm that decomposes a discrete signal into a sum of sine waves, encoding their frequency, amplitude and phase. It can be calculated in $O(n\log n)$. The DFT $X_0 ... X_{N-1}$ of sequence $x_0 ... x_{N-1}$ is defined as follows:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi ikn}{N}}$$

The Discrete Cosine Tranform is a similar transform, but one that only uses real numbers. It can also be calculated in $O(n\log n)$. Whilst there are many variations of it, all further references to the DCT refer to the DCT-II, which is defined as follows:

$$C_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right)$$

## 1.3 Converting frequencies into notes

When a notes is played, the resultant sound is the summation of a sequence of sine waves, with coefficients and frequencies varying by the note's timbre. The lowest frequency that is always present when the note is played is called the *fundamental frequency*, and is considered the frequency of the note. Any higher frequency associated with the note is called an *overtone*, the overtones and the fundamental are together called the *partials* of the note. From this, we can associate every note in the western musical scale with a frequency by associating it with it's *fundamental frequency*.

The association works as follows, we number each note $j$ by the number of semitones above A0 it is, $\#n(j)$. Then the frequency $f_j$ associated with note $j$ is:

$$f_j = 440 \cdot 2^{\frac{\#n(j)}{12} - 4}$$

We can use this to extract notes from our DCT/DFT. This can be further enhanced by associating a 'bucket' of frequencies with each note, by partitioning the frequency spectrum by their (logarithmically) closest associated frequency, allowing the entire FFT/DCT output to be used when classifying notes.

### 1.4 Window functions

Since the signal is usually not periodic in the number of samples in the DFT window, spectral leakage occurs. A simple way to combat this is via the application of a window function, which gives higher weighting to samples in the middle of the window. A window function is any function satisfying:

$$\begin{aligned} w &: \mathbb{Z} \times \mathbb{N}_{>0} \to [-1, 1] \\ i &\notin [0, n) \Rightarrow w(i, n) = 0 \end{aligned}$$

Application of a window function is simple: to apply the window function to a sample $x_i$ in a window of size $n$, you simply multiply $x_i$ by $w(i, n)$. The window function used in the reference analyser, and also used here is the Blackman-Harris window:

$$w_{\mathrm{BH}}(i, n) = \begin{cases} 0.422323 - 0.49755 \cos\left(\frac{2\pi i}{n}\right) + 0.7922 \cos\left(\frac{4\pi i}{n}\right) & \text{if } i \in [0, n) \\ 0 & \text{otherwise} \end{cases}$$

Overlapping samples are also employed to lower the chance of missing a note due to it being shorter than the DFT window, which is a problem made only worse by the use of a window function since some sections of the audio would be given very little weighting.

## 2 Problems with naive DCT based transcriptions

### 2.1 Harmonic and Inharmonic interference

One of the major issues observed in the operation of the spectral analysis transcriber is that it has considerable difficulty distinguishing notes from their octave harmonics. Whenever a note is played, it's fundamental frequency is sounded, but typically the harmonics are also sounded as well. As a result, not all tones that appear in the FFT/DCT correspond to an actual note being played, so these overtones must be removed before note analysis is performed. However, some of these harmonic tones may also be the fundamental frequencies of other notes that are being played! This leaves us with the quite difficult task of figuring out how much of the volume of each tone is caused by a note below it.

The author of the reference transcriber mistakenly assumes that a note can only interfere with, or be interfered by, a note whose fundamental frequency differs from it by a power of two. As a result of this, their strategy to reduce interference only worries about notes that differ by an integer number of octaves from each other, which could cause problems where the interference is on some non power of two harmonic. As an example, the third harmonic of a note might not be recognized as a harmonic of that note. If this tone has sufficient volume, a *just-perfect* fifth chord could be detected, where none occured.

However, not even accounting for all harmonic frequencies would sufficient to solve this problem. Not only can a note create harmonic overtones, it can also, in some cases, cause inharmonic overtones. Certain ways of playing instruments, and some instruments in general cause inharmonic tones, eg a string instrument being plucked, or most percussion instruments being played. This means that any solution that operates by filtering out overtones would have to account for possible interference between notes of any two frequencies, to be generally applicable.

### 2.2 Inaccuracy when multiple notes are played

Whilst the methods used by the reference transcriber are somewhat effective at identifying up to two notes, having more than that will cause entire notes to be missed. This is probably due to the excessive filtering needed to extract notes from simpler samples, which would result in correct notes being discarded as noise in more complex pieces.

## 3 A digression into why modelling as a stochastic process would be ineffective

A seemingly-simple method that comes quickly to mind when encountering this problem is to use information about how notes map to tones and what sequences of notes are expected to be common in the audio sample to create a Hidden Markov Model, and apply the viterbi algorithm to augment the results.

The biggest reason why this would not work is state-space explosion. The case where we have a very simple melody does not really require any augmentation, since the methods used by the spectral analyser are by and large effective enough for these cases. What we really care about are cases with multiple simultaneous notes being played. If we model the markov chain as states are the individual notes being played, we cannot really use it on samples where multiple notes are being played.

A way to get around this would be to consider combinations of notes as our states. This gives us a state space of size $|S| = 2^N$, where $N$ is the number of notes we account for (which in practice would be at least 88, which is the number of keys on a piano). This is already infeasible, since Viterbi requires

$O(|S|^2 + |S|O)$ space, and $O(|S|^2 \times T)$ time to run. Even if we were to limit this by only allowing up to $k$ notes to be pressed at a time, we would still end up with $|S| = C_k^n$, which still grows very fast in $k$.

This problem becomes even worse when polyphonic music is considered. Monophonic music is music that only has one 'voice', in which case there are very strong structural ties between consecutive notes and chords in, which case Viterbi could be somewhat useful. Polyphonic music has multiple voices, and the Viterbi algorithm is incapable of leveraging knowledge about individual voices to reason about their composition.

Modelling a piece of music as a stochastic process would require some way of partitioning notes into their corresponding voices to succeed, which is incredibly non-trivial, as the notes of two different voices can stay very close to each other. If this problem were to be solved, the problem of state space explosion would be just-barely avoidable, since each voice could realistically produce 5 or less notes simultaneously.

# 4 An iterative method for filtering out interfering overtones

## 4.1 Aim of method

We propose a radical simplification to our model to make it more tractable, one that is somewhat justifiable, but not necessarily always true. We shall assume that all notes of an instrument are approximately the same, just frequency shifted by the ratios of their fundamental frequencies. We shall also assume that their is only one instrument in the playback (though this is a much simpler assumption).

This implies that all notes have the same overtone distribution, which greatly simplifies the process of filtering out the overtones of a note. In practice, this is not true, the distribution is different for different notes of the same instrument, and even different instances of the same note. There are a lot of reasons behind this, most of them related to psycho-acoustics, some of them related to constructive and destructive interference, how the human ear responds to different frequencies, and the compensation methods employed by the instruments to counteract these effects. The assumption is still somewhat justifiable though, since the distribution of these overtones partially determines the timbre of the instrument, and since instruments typically aim to sound uniform in timbre across their musical ranges.

We propose an iterative method for approximating the distribution of overtones for an instrument given a series of observed volumes of tones, and to use it to better isolate notes from the output of the DCT/FFT. This method takes advantage of the fact that notes can only interfere with notes above them, not notes below, and attempts to figures out the overtone distribution from bottom-up.

## 4.2 Overtone Distributions

We assume, like the spectral analyser does, that there are only 137 different pitches to distinguish, and operate on a vector of 137 non-negative real numbers representing the volumes of each tone as detected in the sample. We define this vector as:

$$\begin{aligned} &V \in (\mathbb{R}_0^+)^{137} \\ &V_i = \textit{Volume of note i semitones above A0} \end{aligned}$$

We then define a vector of overtone ratios $O$, henceforth also referred to as an overtone distribution, and a vector of determined notes $N$ as follows:

$$\begin{aligned} O &\in (\mathbb{R}_0^+)^{136} \\ N &\in (\mathbb{R}_0^+)^{137} \\ O_i &= \textit{ratio of volume of Fundamental Frequency} \\ &\quad\ \textit{to volume of overtone i+1 semitones above} \\ V_i &= N_i + \sum_{k=0}^{i-1} N_k O_{i-1-k} \end{aligned}$$

Whilst negative overtone ratios are possible, due to destructive interference, we assume that they are anomalous, and model $O$ as only containing non-negative values. From this, it follows that:

$$N_i = V_i - \sum_{k=0}^{i-1} N_k O_{i-1-k}$$

It becomes apparent that $V_0 = N_0$, and that we can calculate the value of $N$ by iteratively solving for each successive element in $N$, and subtracting the overtone distribution for that term. Which gives us the following iterative algorithm for finding $N$ given $O$ and $V$ (NOTE: in the pseudocode below, sequences such as 1..2 include the start but not the end):

```
for i in 0 .. 137:
  N[i] := V[i]
  for j in 0 .. 136-i:
    V[i+j+1] := V[i+j+1] - N[i] * O[j]
```

## 4.3 How do we find $O$?

We need to actually calculate $O$ before we can use it to remove overtones.

One way of doing so is having a single known note played a couple times at the beginning, so as to get a rough estimate of the instrument's overtone distribution.

We can use feedback from $N$ to iteratively calculate better approximations of $O$. We have to filter out the noise from $N$ to figure out which notes were actually present, and which notes were noise. We set the notes that were determined to be noise to 0 in $N$. We then calculate $O'$ to minimize the difference between the $V$ and $V'$, where $V'$ is calculated from $O'$ and the filtered $N$.

Calculating $O'$ can be expensive, however we note that we care a lot more about smaller overtone frequency ratios than larger ones, since their volume ratios tend to be higher. Hence, we can make our calculation of $O'$ greedy without too much loss in accuracy, by trying to maximise smaller frequency ratios first. This allows the algorithm to be more efficient.

We then feed $O'$ into whatever algorithm we are using to approximate $O$, which will give us the next approximation.

Our new algorithm for determining $N$, given $V$ and an approximation of $O$ is:

```
O := approximater.get()
V_t := V.clone()
for i in 0 .. 137:
  if V_t[i] < noise_threshold:
    N[i] := 0
  else:
    N[i] := V_t[i]
    for j in 0 .. 136-i:
      V[i+j+1] := max(0, V[i+j+1] - N[i] * O[j])
O_t := new double[136]
V_t := V.clone()
for i in 0 .. 137:
  V_t[i] := V_t[i] - N[i]
for i in 0 .. 136:
  O_t[i] = 0
  j := 0
  while j+i+1 < 137 && (N[j] == 0 || N[i+j+1] == 0):
    j := j + 1
  if j+i+1 != 137:
    O_t[i] = V_t[j+i+1]/N[j]
    for j in 0 .. 137-i-1:
      V_t[i+j+1] := max(0, V_t[i+j+1] - N[j] * O_t[i])
approximator.feed(O_t)
```

### 4.4 Noise

For the purpose of detecting noise, we shall reuse the noise filtration system used in the Spectral analyser. We simply consider the volume of the loudest notes of this sample $v$ and the threshold of the previous sample $t'$ and set the noise threshold $t = \max(sv, t'd)$ for some constants $0 \leq s, d \leq 1$. For our experiments, we set $d = 0.7, s = 0.4$

## 4.5 Initial distribution

We now have to choose an initial distribution. Whilst simply choosing zero would probably be sufficient, we make the observation that typically, the overtones of a note are its harmonics. With this in mind, we choose an initial vector $O$ that is zero for all elements $O_i$ except where $2^{i/12}$ is the closest value to some integer $1 < k < 7$, in which case $O_i = 1/(k * k)$.

## 4.6 Approximators

Now we get on to the subject of how we design our approximators. An approximator is some opaque object that can be queried for an overtone vector determined by the overtone vectors that were fed into it so far.

### 4.6.1 Echo approximator

The simplest approximator possible would be the Echo approximator. It basically returns the last vector that was fed in to it. For obvious reasons, this would be an awful approximation.

### 4.6.2 Average of history

A better approximator would be one that returns a mean of some number of previous overtone vectors. In particular, it would average the last $n$ vectors, which are initially just the initial vector $n$ times. This is the approximator we use.

# 5 Results and further improvement

## 5.1 Results

In practice, the techniques shown in this paper do not greatly increase the quality of the transcription. There are a few areas where this algorithm creates a slightly more accurate, and it is never noticably worse than the baseline, so it can tentatively be called an improvement over the naive spectral analyser. Comparisons between transcriptions with and without this algorithm can be found in the appendix.

It is, however, noteworthy that when the outputs of the analysis were fed back into a synthesizer (synth.cc in the source code), the songs that were transcribed were mostly recognizable, and in some sections fairly correct. This implies that whilst it is not sufficient for automated audio transcription, it might be still be useful as a guide for manual transcription.

## 5.2 Ideas for further improvement

Using linear programming to calculate the optimal $O'$, instead of greedily calculating a lower-term biased $O'$, would probably improve the results in heavily pedaled pieces. I believe in retrospect that the greedy calculation of $O'$, whilst favouring the frequencies that are closer to the fundamental, also greatly favours the lower frequency of notes in general. This causes the base of any octave chord to dominate the higher part, which causes innaccuracies for certain sequences of notes (eg. arpeggios being played with pedal).

Another interesting idea for further improvement is using maximum likelihood estimation (or some analogue of it) for choosing an approximation for $O$ based on past $O'$s, instead of simpy averaging.

One final, if not very well fleshed out potential avenue for advancement lies again in the Viterbi algorithm. Were it possible to collapse the state space heavily, or find some way of effectively partitioning the observed tones into their respective voices, Viterbi might be feasible.

# Appendices

## A Transcriptions

Note: The numbers represent the time elapsed in $\frac{1}{32}$-seconds.

### A.1 The Well-Tempered Clavier (pedal)

This is an excerpt from the attempted transcription of the beginning of a pedaled recording of Bach's 'The Well-Tempered Clavier'.

#### A.1.1 With our augmentations

```
109: C4 down B3 down C#4 down
110: B3 up C#4 up
128: C4 up E4 up
136: C5 down
149: C5 up
156: C5 down
169: C4 down
170: C5 up
172: C5 down
177: E4 down C5 up
179: E5 down C5 down
180: C5 up E5 up
182: C4 up E4 up
190: C5 down
201: C5 up
210: C5 down
216: C5 up
223: C4 down
510: G4 down C4 up
517: C5 down
528: C5 up
556: C4 down C5 down
560: G4 up C5 up
571: A4 down G4 down
576: A4 up
578: D#5 up C4 up E4 up
585: A5 down E5 up
591: A4 down
597: A4 up
612: C4 down C5 down
617: C5 up
624: A5 up
625: A4 down
629: A4 up
632: C4 up E4 up
639: A5 down
```

245: D4 up C4 up
259: A4 down C4 down
261: F5 down C4 up
264: D4 up A4 up F5 up
279: C4 down C5 down
284: C5 up
287: C4 up
292: C4 down
299: D4 up C4 up
310: C4 down
314: A4 down
319: C#4 down
320: D5 down C4 up C#4 up D4 up A4 up F5 up
334: C4 down
335: B3 down C4 up D5 up F5 up
353: B3 up
355: D5 down B3 down D4 up
356: B3 up
365: B3 down
368: B3 up D5 up F5 up
388: B3 down
410: B3 up D4 up
412: B3 down
420: B3 up D4 up F5 up
442: C4 down B3 down
443: B3 up
455: C4 up E4 up E5 up
463: C5 down
474: C5 up
483: C5 down
492: C5 up
496: C4 down
646: A4 down C4 down
650: C4 up A4 up
653: A5 up
660: A5 down
667: C4 down
669: E5 up A5 up
675: C4 up
678: C4 down
680: C4 up
722: C4 down
727: D4 down C#4 down
728: C#4 up
734: C4 up
740: C4 down
741: C4 up
745: C4 down
747: C4 up D4 up
775: B3 down
789: G4 down B3 up D4 up
794: B3 down
795: D5 down B3 up
802: B3 down
807: B3 up
828: B3 down G4 up
848: B3 up
855: B3 down
860: B3 up D5 up
882: B3 down
888: C4 down
897: C4 up
902: B3 up
907: B3 down

### A.1.2 Without our augmentations

109: C4 down B3 down C#4 down
110: C#4 up B3 up
111: C5 down
117: C5 up
121: E4 down
122: C4 up
124: C4 down E5 down
127: E5 up
128: E4 up C4 up
129: G4 down
136: C5 down
300: D5 down
307: F5 down
310: C4 down D4 down
315: A4 down
318: A4 up
320: C4 up D4 up F5 up
328: F5 down
335: B3 down D5 up F5 up
342: D4 down
349: G4 down
351: D4 up

142: E5 down
149: E5 up C5 up
156: C5 down
169: C4 down
172: G4 up
177: E4 down
179: E5 down
181: E5 up
182: C4 up E4 up C5 up
183: G4 down
190: C5 down
196: E5 down
201: E5 up C5 up
211: C5 down
213: C5 up
219: E5 down
224: C4 down
228: G4 up
229: C5 down
230: C5 up E5 up
231: D4 down
234: C4 up
241: C4 down
243: C4 up
245: D4 up
246: D5 down
253: F5 down
254: F5 up
257: D4 down
259: A4 down
261: F5 down
263: F5 up
264: A4 up D4 up
279: C4 down C5 down D5 up
285: C5 up
286: D4 down
287: C4 up
292: C4 down
299: D4 up C4 up
353: B3 up
355: D5 down
363: F5 down
366: B3 down
367: D5 up
368: B3 up F5 up
376: D5 down
382: F5 down
384: F5 up
386: D5 up
388: B3 down
390: G4 up
394: D4 down
397: D5 down
398: D5 up
402: G4 down
404: B3 up
405: D4 up
406: B3 down
407: D4 down
408: D4 up
409: D5 down
410: B3 up
415: B3 down
419: D5 up
420: B3 up
429: D5 down
435: F5 down
439: F5 up
440: D5 up
442: C4 down B3 down
443: B3 up
445: C5 down
447: C5 up G4 up
448: E4 down
452: E5 down
455: C4 up E4 up E5 up
456: G4 down
463: C5 down

## A.2 The Well-Tempered Clavier (no-pedal)

This is an excerpt from the attempted transcription of the beginning of a non-pedaled recording of Bach's 'The Well-Tempered Clavier'.

### A.2.1 With our augmentations

27: C4 down
29: G5 down
30: G5 up
34: E4 down D#4 down
35: C4 up D#4 up
43: G4 down
51: C5 down
56: G4 up
57: E4 up C5 up
58: E5 down
60: E5 up
63: E4 down
66: G4 down E4 up
71: E4 down G4 up
72: E4 up
74: C5 down
81: C5 up
82: E5 down
90: C4 down G5 down E5 up
93: G5 up
97: E4 down D#4 down C4 up
98: D#4 up
106: G4 down
114: C5 down G4 up
116: C5 up
120: E4 up
121: E5 down
123: E5 up
124: E4 down
128: E4 up
129: G4 down
132: G4 up
134: E4 down
135: E4 up
136: C5 down
143: C5 up
144: E5 down
150: E5 up
151: C4 down C#4 down C5 down G5 down
152: C#4 up C5 up G5 up
154: G5 down
155: G5 up
160: D4 down C#4 down
161: C4 up C#4 up
168: A4 down
170: D4 up
186: F5 up
190: A4 down A#4 down
191: A#4 up
195: A4 up
199: D5 down
207: F5 down D5 up
212: F5 up
214: C4 down C#4 down G5 down C5 down
215: C#4 up C5 up
218: G5 up
222: C4 up
223: D4 down
231: A4 down
238: D5 down
240: A4 up
244: C4 down
245: D5 up C4 up
246: F5 down D4 up
249: F5 up
251: D4 down
252: D4 up
253: A4 down
257: A4 up
261: D5 down
268: F5 down
269: D5 up
273: F5 up
276: B3 down A#3 down
278: A#3 up
285: D4 down B3 up
288: B3 down
292: G4 down
300: D5 down
302: D4 up
305: D4 down G4 up
307: F5 down
309: D5 up
310: D4 up
311: F5 up
313: D4 down
314: D4 up
315: G4 down B3 up
320: G4 up
322: D5 down
331: F5 down
332: D5 up

176: D5 down A4 up
178: D4 down
181: A4 down
183: F5 down D4 up A4 up D5 up
338: F5 up
339: B3 down
347: B3 up
348: D4 down

### A.2.2 Without our augmentations

27: C4 down
34: E4 down C4 up
43: G4 down
51: C5 down
53: C5 up
54: G4 up
55: C5 down
56: C5 up
57: E4 up
58: E5 down
60: E5 up
64: E4 down
65: E4 up
66: G4 down
69: G4 up
74: C5 down
81: C5 up
82: E5 down
89: E5 up
90: C4 down G5 down
92: G5 up
96: C4 up
97: E4 down
106: G4 down
114: G4 up
120: E4 up
121: E5 down
123: E5 up
125: E4 down
128: E4 up
129: G4 down
132: G4 up
136: C5 down
138: C5 up
140: C5 down
143: C5 up
144: E5 down
150: E5 up
151: C4 down C#4 down C5 down G5 down
183: F5 down D5 up
186: F5 up
190: A4 down A#4 down
191: A#4 up
194: A4 up
199: D5 down
206: D5 up
207: F5 down
212: F5 up
214: C4 down C#4 down C5 down G5 down
215: C#4 up
216: C5 up
218: G5 up
222: C4 up
223: D4 down
231: A4 down
233: D4 up
238: D5 down
240: D4 down A4 up
245: D5 up D4 up
246: F5 down
249: F5 up
253: A4 down
257: A4 up
261: D5 down
267: D5 up
268: F5 down
273: F5 up
276: B3 down
285: D4 down B3 up
290: B3 down
292: G4 down
297: B3 up
299: B3 down D4 up
300: D5 down
302: B3 up
304: B3 down
305: G4 up
307: F5 down

152: C#4 up G5 up
153: C5 up
154: G5 down
155: G5 up
160: D4 down C4 up
168: A4 down
170: D4 up
176: D5 down A4 up
178: D4 down
181: A4 down
182: D4 up A4 up
309: D5 up
311: F5 up
315: G4 down B3 up
320: G4 up
322: D5 down
331: F5 down
332: D5 up
338: F5 up
339: B3 down
347: B3 up
348: D4 down

## A.3 Polyphonic

This is an excerpt from the attempted transcription of a polyphonic piece. The piece is 'Bach-Brahms: Chaconne in D minor, performed by Krystian Zimerman'.

### A.3.1 With our augmentations

48: D0 down
49: G0 down
51: G0 up
52: C0 down
53: C0 up D0 up
55: C0 down
56: D0 down C0 up
58: C0 down
59: C0 up D0 up
60: F0 down
61: D0 down
62: D0 up F0 up
64: C0 down
65: C0 up
67: D0 down
68: D0 up
70: F0 down
72: F0 up
73: D0 down C0 down
74: C0 up D0 up
75: A3 down A4 down
76: C5 down
78: F4 down
79: C5 up
84: G#3 down
86: G#3 up
134: C0 down D0 up
135: F4 down C0 up
136: D0 down
137: D0 up F4 up
138: D4 up A4 up
139: C0 down
140: D0 down
141: C0 up D0 up
142: A3 down A4 down F4 down
144: C5 down
147: C5 up
156: A3 up
160: A3 down
162: F4 up A3 up
163: E4 down C0 down D#4 down
164: C0 up D#4 up A4 up
165: A#3 down A3 down
167: A3 up
179: D4 down
180: D4 up
182: D4 down
183: A#3 up D4 up
185: A#3 down D4 down
186: D4 up
188: D4 down
190: A4 down

92: E5 down
96: D4 down E5 up
98: D4 up F4 up
101: G#3 down F4 down
102: G#3 up
103: D4 down
105: F4 up
107: D3 down
108: G#3 down D3 up
109: C0 down
110: E5 down D3 down C0 up G#3 up
111: D3 up E5 up
112: F4 down
114: F4 up
115: C0 down
117: C0 up A4 up
118: F0 down
119: D0 down A3 up F0 up
120: D4 up
121: C0 down
122: D0 up
126: D0 down C0 up
129: D4 down
132: A4 down
191: A4 up
193: D0 down C0 down G3 down
194: C0 up
195: G3 up D0 up
197: G3 down
204: A#3 up
205: C0 down D0 down
206: C0 up D0 up E4 up
207: G3 up D4 up
208: C#2 down D0 down F0 down
209: F0 up D0 up C#2 up
210: C0 down
211: D0 down
212: C0 up D0 up
216: E4 down
223: A4 down
224: A4 up
226: A4 down
227: A4 up
229: A4 down
231: A4 up
238: A4 down
257: G3 down
260: A4 up

### A.3.2 Without our augmentations

48: D0 down
50: D0 up
52: D0 down C0 down
53: D0 up C0 up
57: D0 down
58: C0 down
59: C0 up D0 up
61: F0 down
62: F0 up
64: C0 down
65: C0 up
67: D0 down
68: D0 up
70: F0 down
71: F0 up
73: D0 down
74: D0 up
75: A4 down
76: A3 down
178: A#3 down
181: A#3 up
192: D4 down
193: A#3 down
194: D0 down A#3 up
195: D0 up
197: G3 down A#3 down
203: A#3 up
204: E4 up
205: C0 down D0 down
206: D4 up C0 up D0 up
207: G3 up
208: C#2 down D0 down F0 down
209: F0 up D0 up C#2 up
210: C0 down
211: D0 down
212: C0 up D0 up
216: E4 down
217: E5 down

82: F4 down
95: E5 down
96: E5 up
98: F4 up
106: D4 down
109: G#3 down C0 down
110: C0 up G#3 up
115: C0 down
116: A3 up A4 up
117: C0 up
119: D4 up
121: C0 down
126: C0 up
130: D4 down
133: A4 down
136: D0 down
137: D0 up
138: D4 up A4 up
139: C0 down
141: C0 up
142: A4 down A3 down
148: F4 down
152: F4 up
154: F4 down
155: A3 up
161: A3 down
162: F4 up A3 up
163: E4 down
164: A4 up
166: A#3 down
168: A#3 up
219: E5 up
250: A4 down
251: A4 up
259: G3 down
260: G3 up
263: G3 down
265: D0 down C0 down
266: E4 up G3 up D0 up
267: C0 up
271: C0 down
273: C0 up
280: E4 down
306: F4 down E4 up
335: F4 up
339: F4 down A#3 down
348: F4 up
349: D4 down D5 down A#3 up
352: A#3 down
354: A#3 up
360: A#3 down
361: A#3 up D5 up
368: D5 down
371: A#3 down A#4 down
372: A#3 up D5 up
374: A#3 down
375: F4 down
376: C0 down D0 down
377: C0 up D0 up
379: C0 down
381: D4 up A#3 up A#4 up C0 up
383: A#4 down

## 6 Bibliography


1. `https://en.wikipedia.org/wiki/Overtone`
2. `https://en.wikipedia.org/wiki/Maximum_likelihood_estimation`
3. `https://en.wikipedia.org/wiki/Loudness`
4. `https://en.wikipedia.org/wiki/Pitch_(music)`
5. `https://en.wikipedia.org/wiki/Harmonic`
6. `https://en.wikipedia.org/wiki/Transcription_(music)#Pitch_detection`
7. Greg's source code for his 4121 project.
8. Greg's report for his 4121 project.

9. https://en.wikipedia.org/wiki/Inharmonicity