Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Andrew McLeod	Date

Automatic Transcription of Polyphonic Musical Signals with Linear Matching Pursuit

By

Andrew McLeod
Master of Science

Mathematics/Computer Science

Li Xiong
Advisor

Shun Yan Cheung
Committee Member

Ken Mandelberg
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Automatic Transcription of Polyphonic Musical Signals with Linear Matching Pursuit

By

Andrew McLeod

Advisor: Li Xiong, Ph.D.

An abstract of
A thesis submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science
in Mathematics/Computer Science
2013

Abstract

Automatic Transcription of Polyphonic Musical Signals with Linear Matching Pursuit
By Andrew McLeod

The Harmonic Matching Pursuit (HMP) algorithm has offered promising results in the automatic transcription of audio signals. It works by decomposing the given signal into a set of harmonic atoms, and then grouping those atoms into individual notes. HMP has shown very promising results, but more research has been needed for one case: when multiple notes with rational frequency relation are played simultaneously. This situation is called the overlapping partial problem, and it is very common in music, occurring in intervals such as major thirds, perfect fourths, and perfect fifths. A few solutions have been proposed to handle this overlapping partial problem by performing post-processing on the output of HMP (notably HMP with Spectral Smoothness (HMP SS) [3]). In this paper, I propose an algorithm called Linear Matching Pursuit (LMP) to solve the overlapping partial problem of automatic note detection, which uses new heuristics to solve the problem with no post-processing required. LMP's runtime is independent of the number of notes present in a given audio signal, unlike HMP. My experiments show that LMP offers an improvement upon the accuracy of the HMP algorithm, though not to the extent of HMP SS, and is very robust in runtime with respect to polyphony.

Automatic Transcription of Polyphonic Musical Signals with Linear Matching Pursuit

By

Andrew McLeod

Advisor: Li Xiong, Ph.D.

A thesis submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science
in Mathematics/Computer Science
2013

Acknowledgments

Firstly, I would like to thank my thesis advisor, Li Xiong, for her support and direction throughout the process of completing this thesis project. She was the first professor who challenged and inspired me to conduct research in the field of computer science. She allowed me the freedom to explore the vast field on my own, but still provided me with just the right amount of structure so that I would not get lost along the way.

I would also like to Ken Mandelberg, my undergraduate advisor, for his guidance throughout my years here at Emory. He has made it easy for me to find my way through college to graduation. His guidance and advice has allowed me to concentrate fully on my research during my time here.

I would like to thank my committee members, Shun Yan Cheung, along with Li Xiong and Ken Mandelberg, for the time they have taken to evaluate me and my thesis.

Thanks to Steve Nowicki, who inspired me to travel down this road to academia.

Thanks also to David Goodwin, who allowed my love of music to grow and eventually inspire this thesis.

Finally, I would like to thank my family for their never ending love and support in my pursuit of this complete thesis project.

Contents

List of Figures

List of Tables

1 Introduction

The automatic transcription of musical signals is a widely researched topic in the field of acoustics and signal processing. The human ear has a remarkably good ability to pick out a tune from an audio signal, but computers have a much more difficult time with this task. The most common method used to perform automatic music transcription is to decompose a given signal into the elementary waveforms of which it is made. This signal decomposition has implications in many fields of study outside of music, notably speech recognition, as used by Lilly and Paliwal [10] as well as Wang and Young [13].

The most commonly used algorithm for performing music transcription is the Matching Pursuit (MP) algorithm, proposed by Mallot and Zhang [11]. MP is a greedy, dictionary-based signal decomposition algorithm which works by picking the waveform from the given dictionary at each step with the highest correlation to the given signal until that signal's energy drops below a certain stopping threshold. It has two main drawbacks, as noted by Carabias-Orti et al. [4]: 1) When the signal contains certain correlated waveforms, MP tends to have a low accuracy; and 2) It is very sensitive to its stopping threshold.

In regards to MP's first drawback, most musical instruments produce sound in a harmonic nature that produces many of these correlated waveforms. The Harmonic Matching Pursuit (HMP) algorithm, proposed by Gribonval and Bacry [6] attempts to account for this property of instruments by altering MP slightly to work with a dictionary of harmonic atoms. HMP has been shown to works much better than MP for the decomposition of music containing strong harmonic content, but still performs poorly for the decomposition of music signals where multiple related notes are played simultaneously (the overlapping partial problem). HMP with Spectral Smoothness (HMP SS) was proposed by Canadas-Quesada et al. [3] to deal with this problem by performing some post-processing on the outputs from the standard HMP algorithm.

The method proposed in this paper, called Linear Matching Pursuit (LMP), offers two main advantages over the existing methods: 1) Its runtime is bounded only by the length of the input signal and the size of the dictionary, rather than the number of notes present in that signal, as in the existing approaches; and 2) It no longer relies on a global stopping threshold to decompose a signal. Instead, LMP tests each note in the dictionary once, making a decision on whether that note is present based on other heuristics. Some of these heuristics are still based on the energies of each individual note, but the fact that there is no global stopping threshold should, make it more robust when dealing with sound signals containing non-periodic sounds (such as drums). Since LMP checks

each note in the dictionary at each timeslice, its runtime is very consistent and scales well when decomposing musical signals with many notes present simultaneously.

This paper is laid out as follows: Section 2 contains useful definitions and a table of symbols used along with an in depth look at related work and a formal problem statement. The proposed LMP algorithm is presented in Section 3. Section 4 contains experimental settings and results. Conclusion and ideas for future work can be found in Section 5. Section 6 contains relevant reference tables.

2 Background and Related Works

2.1 Terms

Musical signals are essentially functions of time and can be divided into two categories based on their **polyphony** level (the number of notes which occur concurrently in a given signal). **Monophonic Signals** are those which consist of only one note being played at a time. These signals do not contain overlapping partials, and as such, are not relevant to this paper. A **Polyphonic Signal**, on the other hand, is one which contains two or more notes being played simultaneously. This could be either multiple instruments playing any number of notes each, or a single instrument playing more than one note. This paper will deal with polyphonic signals consisting of a single instrument, specifically a piano, playing multiple notes at once.

A **Note-event** is an event in a musical signal corresponding to a played note in that signal. Each note-event can be described by many parameters, but for music transcription, three in particular are critical as noted by Bello et al. [1]: pitch, onset time, and duration. **Pitch** is an attribute of sound which corresponds to its frequency (measured in Hz). This assumes that the given sound is nearly periodic, as are most sounds produced by a musical instrument. A note-event's **onset time** is the time which can describe its starting point in a musical signal [7]. This is essentially the time at which a given note is played on an instrument. The **duration** of a note-event is the length of time for which it is present in a musical signal.

In most music transcription algorithms based on MP, the given signal is decomposed into atoms. An **atom** is an elementary waveform [11] and atoms are essentially the building blocks of musical signals. A redundant set of atoms which spans the entire time-frequency plane is called a **dictionary** [11]. There are many different types of atoms, but the two relevant to this paper are Gabor atoms and harmonic atoms.

A **Gabor atom** is a specific type of atom which is obtained by scaling (s), translating (u), and

modulating (ξ) a mother window ($w(t)$) as follows:

$$g_{s,u,\xi}(t) := \frac{1}{\sqrt{s}} w(\frac{t-u}{s}) e^{i2\pi\xi(t-u)}$$

Each Gabor atom is located around time u with duration on the order of s. Its Fourier Transform $\hat{g}_{s,u,\xi}(w)$ is centered at frequency ξ with a dispersion on the order of $1/s$. In this function, the **mother window** $w(t)$ represents the general shape of the atom and is real-valued, positive, and of unit norm (i.e. its energy is 1). In general, the **energy** of a function is denoted $|f(t)|$ and defined as follows:

$$|f(t)| = \int_{-\infty}^{\infty} f(t)^2 dt.$$

A **harmonic atom** is an atom obtained by summing an ordered set of K Gabor atoms together as follows:

$$h(t) := \sum_{k=1}^{K} \alpha_k g_{s,u,\xi_k}(t)$$

Each $\alpha_k g_{s,u,\xi_k}(t)$ is referred to as a **partial** and, in general, a partial is referred to ordinally as the ith partial for $k = i$. The signal generated by the first partial of an atom is called the **tone** of that atom, while the signals of the remaining $K - 1$ partials are all called **overtones**. These atoms get their name from the harmonicity of the frequency modulation factor ξ_k of each partial. In general, $\xi_k = k\xi_0$, where ξ_0 is referred to as a given atom's **fundamental frequency**, and is the one frequency which most accurately represents a given note [8]. (See Table 4 for a list of the fundamental frequencies of each note on a piano). Each of the K partials of a harmonic atom have the same s and u. The Fourier Transform of a harmonic atom has K peaks, each centered around a different ξ_k with dispersion on the order of $1/s$.

An **overlapping partial** occurs when two harmonic atoms have a rationally related fundamental frequency. For example, if the ratio of note n's fundamental frequency ξ_{0_n} to note m's fundamental frequency ξ_{0_m} is 3:2, every third partial of note m has essentially the same frequency as—and therefore overlaps with—every second partial of note n. This situation actually occurs very frequently in music (see Table 2). For example, consider the notes A4 and E5, which are a perfect fifth apart, and have fundamental frequencies of 440 Hz and 659.255 Hz respectively. Note that for small enough k, the frequency of the $3k$th partial of the note A4 is essentially equal to the frequency of the $2k$th partial of the note E5.

2.2 Related Works

2.2.1 Matching Pursuit

The Matching Pursuit (MP) algorithm was first introduced by Mallot and Zhang in 1993 [11]. MP is a greedy, iterative algorithm that decomposes a signal $f(t)$ into a set of atoms chosen from a redundant dictionary D, with each decomposed atom corresponding to a different note-event in the signal. Mallot and Zhang used a dictionary of Gabor atoms $g_{s,u,\xi}(t)$, each of which are normalized so that their energy is 1. The general MP algorithm has been reproduced here for completeness:

Input: Signal $f(t)$, Dictionary D, Stopping Threshold T_{stop}
Output: List of coefficients and atoms $(a_n, g_{s_n,u_n,\xi_n}(t))$
$R_1(t) \leftarrow f(t)$
$n \leftarrow 1$
while $|R_n(t)| > T_{stop}$ **do**
  $g_{s_n,u_n,\xi_n}(t) \leftarrow g_{s,u,\xi}(t) \in D$ s.t. the inner product $\langle g_{s,u,\xi}(t), R_n \rangle$ is maximized
  $a_n \leftarrow \langle g_{s_n,u_n,\xi_n}(t), R_n \rangle$
  $R_{n+1}(t) \leftarrow R_n(t) - a_n g_{s_n,u_n,\xi_n}(t)$
  $n \leftarrow n + 1$
end while

In general, MP decomposes a signal in discrete timesteps. That is, it only looks at a very small portion of the given signal at a time, the length of which can be set by a parameter. The algorithm as shown only decomposes a single timestep of the signal, but can be called iteratively to decompose any longer signal. MP decomposes its input signal $f(t)$ into atoms from the dictionary until the energy of the residual signal (R_n for the residual signal after removing $n-1$ atoms) falls below a given threshold (T_{stop}).

To choose which atom $g_{s,u,\xi}(t)$ to remove from the signal next, the inner product between the signal and every atom in the dictionary is computed and the atom with the maximum inner product is chosen. In general, the inner product of two functions $f(t)$ and $g(t)$ is denoted as $\langle f(t), g(t) \rangle$ and is defined as $\int_{-\infty}^{\infty} f(t)g(t)dt$. In general, the greater the value of the inner product of two functions is, the more similar they are. Its coefficient a_n is set to that inner product, and then the product $a_n g_{s,u,\xi}(t)$ is subtracted from the current residual. There are two important things to note here:

1. Since the Gabor atoms are all normalized such that their energies are all 1, the inner product

between an atom $g_{s,u,\xi}(t)$ and a signal $f(t)$ is exactly the coefficient a_n that will maximize the inner product between that signal $f(t)$ and the product $a_n g_{s,u,\xi}(t)$.

2. Maximizing the inner product between $f(t)$ and $a_n g_{s,u,\xi}(t)$ is equivalent to minimizing the energy of the residual $R = f(t) - a_n g_{s,u,\xi}(t)$, which is exactly what this algorithm attempts to accomplish.

MP has been proven to be a convergent algorithm, and while it is not guaranteed to find the optimal solution, its greedy approach to decomposing a musical signal performs well when decomposing signals without correlated waveforms. As noted above, however, most musical instruments produce sounds which are harmonic in nature. Since MP uses a non-harmonic dictionary, it tends to decompose every partial of a note as a different atom.

2.2.2 Harmonic Matching Pursuit

In 2004, Gribonval and Bacry [6] proposed a modification to the original Matching Pursuit algorithm called Harmonic Matching Pursuit (HMP). HMP extends MP to use a dictionary of harmonic atoms $h(t)$, where each of the K (set by a parameter) partials of every atom has been normalized so that its energy is 1. This harmonic dictionary tends to work better when decomposing the signals of most real-world instruments because of the harmonic nature of the sound they produce.

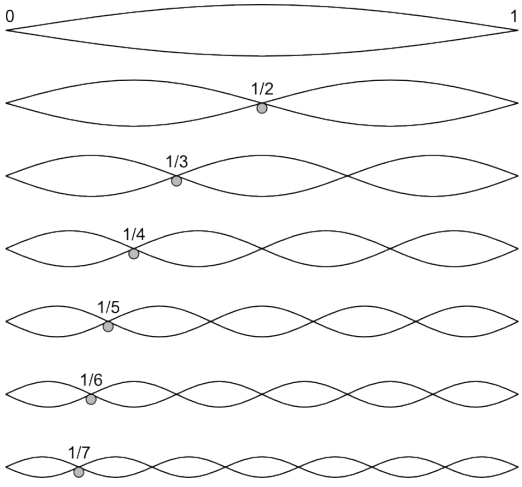

Figure 1: The first 7 harmonics vibrating on a string

When a string or an air-column vibrates, it vibrates at many different frequencies at once, not just at its fundamental frequency. Since a string of length L is fixed at both ends, it can only vibrate with a wavelength λ such that the length of the string is a multiple of half of the wavelength. That is,

$L = \frac{n}{2}\lambda$ for any natural number n (the partial number). Since the frequency ξ of a wave is $O(1/\lambda)$, the frequency of the nth partial of a harmonic atom can be derived with the following formula:

$$\xi_n = n\xi_0 \text{ for } n \geq 1$$

In general, each successive partial vibrates at a slightly lower amplitude than the previous partial. See Figure 1 for an illustration of the first seven harmonic partials vibrating on a string. In wind and brass instruments, a similar harmonic series of partials is produced, with the fundamental frequency depending instead on the length of the pipe [2].

The HMP algorithm is reproduced here for completeness:

Input: Signal $f(t)$, Dictionary D, Stopping threshold T_{stop}
Output: List of n coefficients and atoms $(\alpha_{n,1}, \alpha_{n,2}, ..., \alpha_{n,K}, h_n(t))$
$R_1(t) \leftarrow f(t)$
$n \leftarrow 1$
while $|R_n(t)| > T_{stop}$ **do**
  for each $h_m(t) \in D$ **do**
    $temp_m(t) \leftarrow R_n(t)$
    for $i = 1 \rightarrow K$ **do**
      $\alpha'_{m,i} \leftarrow \langle g_{s_m,u_m,\xi_{m_i}}(t), temp(t) \rangle$
      $temp_m(t) \leftarrow temp_m(t) - \alpha'_{m,i} g_{s_m,u_m,\xi_{m_i}}(t)$
    end for
  end for
  Each $\alpha_{n,i} \leftarrow \alpha_{m,i}$ s.t. $\sum_{i=1}^{K} \alpha_{m,i}$ is maximized
  $h_n(t) \leftarrow$ corresponding $h_m(t)$
  $R_{n+1}(t) \leftarrow$ corresponding $temp_m(t)$
  $n \leftarrow n + 1$
end while

The HMP algorithm itself is very similar to the original MP algorithm. The only differences are due to the dictionary used. At each iteration of the algorithm, an inner product must be calculated between the current residual signal and each *partial* of each atom in the dictionary, rather than just the inner product between the current residual and each *atom* in the dictionary, as in MP. Similarly, each individual partial is assigned its own coefficient rather than an atom as a whole

getting a coefficient, as in MP. The atom selected is the atom which maximizes the sum of its partial's coefficients. Note that this algorithm is equivalent to MP if K is set to 1.

HMP offers a good improvement over MP when decomposing music created by real-world instruments. However, it still struggles when attempting to decompose signals containing overlapping partials. In general, HMP will tend to decompose a musical signal containing two notes with overlapping partials into an atom whose fundamental frequency is the greatest common factor of the fundamental frequencies of the notes which are actually present. For example, HMP will most likely decompose a musical signal containing the notes A4 ($\xi_0 = 440$ Hz) and E5 (ξ_0=659.225 Hz) into the atom corresponding to the note A3 (ξ_0=220 Hz). While the sum of the coefficients of the first K partials of the atoms corresponding to A4 and E5 would be $\sum_{k=1}^{K} b_{A4,k}$ and $\sum_{k=1}^{K} b_{E5,k}$ respectively, the sum of the first K partials of the atom corresponding to A3 would be $\sum_{k=1}^{\lfloor K/2 \rfloor} b_{A4,k} + \sum_{k=1}^{\lfloor K/3 \rfloor} b_{E5,k}$. Since, in general, each subsequent partial has a lower amplitude than the previous partial, this third sum would likely be greater than both the second sum and the first sum, and the atom corresponding to A3 would incorrectly be selected as the one with the maximal sum in line 15 above.

2.2.3 HMP with Spectral Smoothness

In 2008, Canadas-Quesada et al. [3] proposed an algorithm to solve the overlapping partial problem using spectral smoothing techniques. This algorithm is run as post-processing on the coefficients (referred to here as $\alpha_{n,k}$, where n is the note number and k is the partial number) from the standard HMP algorithm. The SS post-processing steps have been reproduced below for completeness. First, the coefficients are constrained to be strictly decreasing from their max:

$$\tilde{b}_{n,k} = \begin{cases} \alpha_{n,k_{n_{max}}} & : k = k_{n_{max}} \\ min(\tilde{b}_{n,(k-1)}, \alpha_{n,k}) & : k > k_{n_{max}} \\ min(\tilde{b}_{n,(k+1)}, \alpha_{n,k}) & : k < k_{n_{max}} \end{cases}$$

Consider again a musical signal containing the notes A4 and E5 as decomposed by HMP. The fifth partial of the decomposed atom corresponding to A3 would have a coefficient of 0, since neither A4 nor E5 contain the corresponding partial. After this first step of the SS post-processing, all subsequent coefficients would be set to 0 as well.

In the next step, those coefficients are smoothed by averaging each $\tilde{b}_{n,k}$ with the coefficients of the partials of note n whose frequency is within an octave of that kth partial as follows:

$$b_{n,k} = \begin{cases} \tilde{b}_{n,k_{n_{max}}} & : k = k_{n_{max}} \\ min(\tilde{b}_{n,k}, \frac{\sum_{j=\lceil k/2 \rceil}^{2k-1} \tilde{b}_{n,j}}{(2k-1)-\lceil k/2 \rceil+1}) & : k \neq k_{n_{max}} \end{cases}$$

These new coefficients are then run through another algorithm, which has been reproduced below for completeness:

Inputs: Coefficients $\alpha_{n,k}$ from HMP, Stopping thresholds D_{th1} and D_{th2}
Outputs: Set of present notes $Fundamentals$
Compute $b_{n_c,k}$ and D_{n_c} for $1 \leq n_c \leq K$
for each candidate n_c with $D_{n_c} > D_{th1}$ **do**
    Begin a *way* w from the current n_c
    $D_w \leftarrow 0$
    $r_{w,n_c,k} \leftarrow \alpha_{n,n_c k}$ for $1 \leq k \leq \lfloor \frac{K}{n_c} \rfloor$
    $Fundamentals_w \leftarrow \{\}$
    while $D_{n_c} > D_{th2}$ **do**
        $Fundamentals_w \leftarrow Fundamentals_w \cup \{n_c\}$
        $D_w \leftarrow D_w + D_{n_c}$
        $r_{w,n_c,k} \leftarrow r_{w,n_c,k} - b_{n_c,k}$ for $1 \leq k \leq \lfloor \frac{K}{n_c} \rfloor$
        Compute $b_{n_c,k}$ and D_{n_c} from $r_{w,n_c,k}$ for $1 \leq n \leq K$
        $D_{n_c} \leftarrow \max(D_{n_c})$
    end while
end for
return $Fundamentals_w$ with $\max(D_w)$

This algorithm is run once for each note detected by HMP. It begins by calculating a smoothed distance metric D_{n_c} for each candidate note $1 \leq n_c \leq K$ as follows:

$$D_{n_c} = \sum_{k=1}^{\lfloor \frac{K}{n_c} \rfloor} |b_{n,n_c k}|$$

This smoothed distance is essentially the sum of the coefficients corresponding to each potential fundamental frequency of an atom. These potential fundamental frequencies are the frequencies corresponding to each individual partial of the original atom. D_{th1} is used here only to speed up the computation time of the algorithm; it is not necessary for its functionality. Each n_c where $D_{n_c} > D_{th1}$ is initialized as a new *way*. A *way* refers to a set of harmonic atoms. The goal is

for the *way* chosen to consist of the actual notes present in the original audio signal. For each of these smoothed distances D_{n_c}, residual coefficients $r_{w,n_c,k}$ are computed based on the differences between the original coefficients and the new coefficients, computed as shown above. New distances are then computed based on these residual coefficients. This process is repeated based on the new distances, adding the corresponding note to the current *way*, until the maximum D_{n_c} is less than a second threshold D_{th2}. After all possible *ways* are computed, the *way* with the maximal sum of the distances of its corresponding notes is chosen to be correct.

2.3 Problem

The music transcription problem as is solved by the proposed LMP algorithm is essentially a supervised classification problem. The goal of LMP is to correctly identify all of the musical notes that are present in each timeslice $f_i(t)$ of a given audio signal $f(t)$. Specifically, for each timeslice $f_i(t)$ of the input signal, LMP classifies each note n corresponding to an atom in the dictionary D as either present or not present.

3 Proposed Solution

I have found it useful to think of the set of possible classifications of a musical signal as a tree structure as in Figure 2. This tree contains every possible subset of n notes. The root of the tree represents an audio signal with no notes present, and in general, all of the nodes at level n of the tree represent a solution that is a combination of n notes. This tree is highly left skewed, so as to remove duplicates. A node representing a set of notes S with maximum note m will have $n - m$ children: one for the union of S with each note in the range $m+1$ to n, inclusive. Once this tree has been formed, the music transcription problem as proposed can be thought of as a search problem through this tree.

3.1 Linear Matching Pursuit

The LMP algorithm traverses the proposed search tree in the following manner, beginning with the head node (which is considered a candidate solution):

- If the current node is a candidate solution, continue searching with its first child. If the current node has no children, it is chosen as the solution.

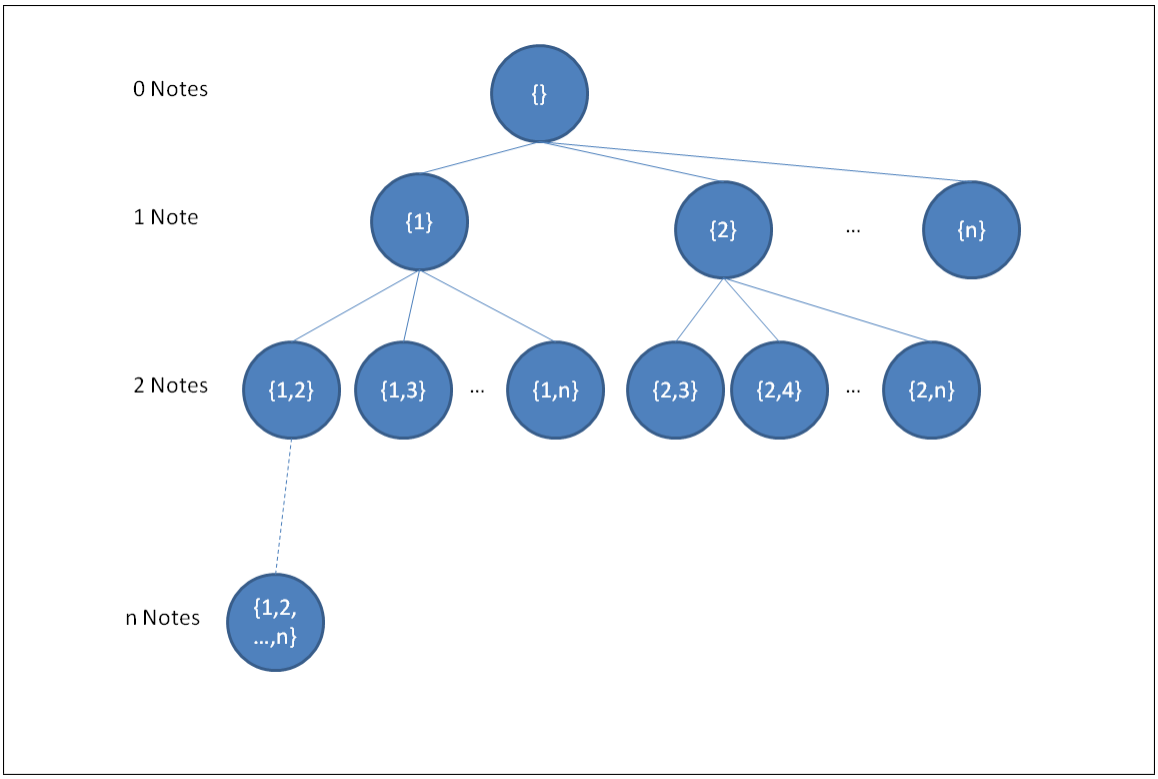

Figure 2: The proposed search tree containing every possible solution to the music transcription problem with an upper bound of n possible notes

- If the current node is not a candidate solution, continue searching with its next sibling. If the current node has no subsequent siblings, its parent is chosen as the solution.

In general, a node is considered a candidate solution if all of the notes present in its note-set are classified as present in the musical signal. A note is classified as present if the coefficients of the corresponding atom from its dictionary meet certain requirements. This behaves like a linear search through the dictionary, with each atom classified in a binary manner as either present or not present. LMP is able to make a decision on each note in such linear fashion by ordering the atoms in the dictionary by their fundamental frequencies, from low to high. This intelligent dictionary ordering ensures that the first partial of each atom tested cannot possibly overlap with any partial of any subsequent atom, and can therefore be assigned entirely to the atom being tested, assuming that all previous classifications were performed correctly. It is this linear search that accounts for the main contribution of LMP: it reduces the computation time from $O(tnm)$ to $O(tm)$, where t is the length of the input signal, m is the number of atoms in the dictionary, and n is the number of notes present in the input signal. The general LMP algorithm used to identify the notes present in a single

timeslice of a musical signal is shown here:

Inputs: Signal $f(t)$, Dictionary D

Outputs: An array of length m, *present*, referring to whether a given note m is present in $f(t)$

$R(t) \leftarrow f(t)$

for $m \leq$ the number of atoms $\in D$ **do**

  $present_m \leftarrow true$

  $max \leftarrow 0$

  $R_{tmp}(t) \leftarrow R(t)$

  $\alpha_{m,1} \leftarrow \langle R_{tmp}(t) | g_{s,u,\xi_1}(t) \rangle$

  if $\alpha_{m,1} < T_{p1}$ **then** ▷ New Heuristic 1

    $present_m \leftarrow false$

    continue

  end if

  $R_{tmp}(t) \leftarrow R_{tmp}(t) - \alpha_{m,1} * g_{s,u,\xi_1}(t)$

  for $2 \leq k \leq K$ **do**

    $\alpha_{m,k} \leftarrow \langle R_{tmp}(t), g_{s,u,\xi_k}(t) \rangle$

    if $max = 0$ and $\alpha_{m,k} < \alpha_{m,k-1}$ **then** ▷ New Heuristic 2

      $max \leftarrow k - 1$

      if $\alpha_{m,k-1} < minMax$ **then** ▷ New Heuristic 3

        $present_m \leftarrow false$

      end if

    else

      if $max \neq 0$ **then**

        $\alpha_{m,k} \leftarrow \min(\alpha_{m,k}, \alpha_{m,k-1})$

      end if

    end if

    if $\alpha_{m,k} = 0$ **then**

      if $k < firstZero$ **then** ▷ New Heuristic 4

        $present_m \leftarrow false$

      end if

```
            break
        end if
        R_tmp(t) ← R_tmp(t) − α_{m,k} * g_{s,u,ξ_k}(t)
    end for
    if Σ_{i=1}^{K} α_{m,i} < minTotal then                    ▷ New Heuristic 5
        present_m ← false
    end if
    if present_m then
        R(t) ← R_tmp(t)
    end if
end for
```

The new heuristics are as follows:

1. Heuristic 1 is a minimum requirement for the coefficient of the first partial of an atom. When a note with fundamental frequency ξ_0 is present in a musical signal, it can contribute only frequencies which are whole multiples of ξ_0; that is, frequencies $\geq \xi_0$. Therefore, if we go through the atoms in order of their fundamental frequencies and find an atom with a loud first partial, we can assume that the corresponding note is present in the signal, as no subsequent atoms will be able to account for that particular frequency. To account for background noise in the signal or imperfectly classified previous atoms, T_{p1} is used. If the coefficients of the first partial of an atom is $\leq T_{p1}$, then it is assumed that the given atom is not present in the signal, and we can safely stop computation on the current atom.

2. Heuristic 2 is the requirement that the coefficients of the partials of a given atom be strictly decreasing away from the max, as in HMP SS. However, some slight tweaking has been done to allow this step to be computed during the decomposition process as opposed to as a post-processing step. The maximum value is picked greedily. That is, the first local maximum found is assumed to be the global maximum. This should not be a problem, since the energy of the partials in a given note normally decrease down from the first partial [2]. No smoothing is performed on the coefficients since that would require some knowledge of the future coefficients to compute.

3. Heuristic 3 is a minimum coefficient requirement for the loudest partial of an atom. That is,

the coefficient of the partial with the greatest corresponding coefficient in a given atom must be $\geq minMax$ for the atom to be classified as present. Since the maximum partial is chosen greedily, if a max does not reach this requirement, it is assumed that the given atom is not present in the signal, and we can safely stop computation on the current atom.

4. Heuristic 4 is the requirement that, for an atom to be present, it must have at least $firstZero$ non-zero-coefficient partials. That is, at least $firstZero$ partials of an atom must be present in a given signal for that atom to be classified as present. Note that due to heuristic 2, all partials following the first zero-coefficient partial will also have a coefficient of 0.

5. Heuristic 5 is the requirement that for an atom to be present, the sum of its partials' coefficients ($\sum_{i=1}^{K} \alpha_{n,i}$ for note n) must be $\geq minTotal$. This is essentially a loudness threshold for an individual atom.

If an atom satisfies all five of these heuristics, it is assumed to be present in the given signal. That atom's signal is then subtracted from the current residual signal, and decomposition continues with the next atom in the dictionary.

4 Experiments

4.1 Metrics

There are a few things to note when considering what metrics to use to evaluate the performance of a music transcription algorithm. First, let us consider the confusion matrix. A true positive (TP) is a note classified correctly as present in a given timestep. A false positive (FP) is a note incorrectly classified as present in a given timestep. A true negative (TN) is a note correctly classified as not present in a given timestep. A false negative (FN) is a note incorrectly classified as not present in a given timestep. Given the nature of music transcription, and the fact that many more notes are not present in a given timestep than are present, it follows that there will be a disproportionately large number of true negatives during classification. Given this, it is likely that any metric involving true negatives will be skewed. To solve this, Dixon [5] proposed a new accuracy metric Acc, defined as follows:

$$Acc = \frac{TP}{FP+FN+TP}$$

This accuracy metric has been used to evaluate many of the previously existing solutions to the automatic music transcription problem. It is bounded by 0 and 1, with a perfect transcription

having an accuracy value of 1.

In 2004, the National Institute of Standards and Technology (NIST) [15] proposed another metric for the evaluation of music transcription algorithms. It consists of a single error score E_{tot}, which can be broken down into three parts: substitution error E_{sub}, which refers to the case in which there is both a false positive and a false negative in a timestep; miss error E_{miss}, which refers to the case in which there is only a false negative in a timestep; and false alarm error E_{fa}, which refers to the case in which there is only a false positive in a timestep. To formally define each of these values as functions, it is useful to define three other values: $N_{ref}(t)$, the number of notes present at timestep t; $N_{sys}(t)$, the number of notes detected at timestep t; and $N_{corr}(t)$, the number of correctly classified notes at timestep t. Thus, the error values can be defined as follows, where T is the total number of timesteps in the input signal:

$$E_{sub} = \frac{\sum_{t=1}^{T}[min(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)]}{\sum_{t=1}^{T} N_{ref}(t)}$$

$$E_{miss} = \frac{\sum_{t=1}^{T} max(0, N_{ref}(t) - N_{sys}(t))}{\sum_{t=1}^{T} N_{ref}(t)}$$

$$E_{fa} = \frac{\sum_{t=1}^{T} max(0, N_{sys}(t) - N_{ref}(t))}{\sum_{t=1}^{T} N_{ref}(t)}$$

Note that these error values have no probabilistic implications in that they are not bounded by 1. Rather, they are percent error values. Specifically, E_{sub} is the number of substitution errors made reported as a percentage of the total number of notes present in a signal, E_{miss} is the number miss errors made reported as a percentage of the total number of notes present, and E_{fa} is the number of false alarm errors made reported as a percentage of the total number of notes present. Note that in the formulas, care is taken to ensure that no errors are counted twice (i.e. if an error is counted as a substitution, it will not be counted again as a miss or a false alarm). Since each of these error values are percent errors, E_{tot} can be computed as a simple sum:

$$E_{tot} = E_{sub} + E_{miss} + E_{fa}$$

This sum will still have a meaningful value. Specifically, E_{tot} refers to the total number of errors found as a percent of the total number of notes present in a given audio signal.

Since one of the main contributions of the LMP algorithm is its low time complexity, its speed will also be measured with respect to a change in polyphony. In theory, this speed should be stay relatively steady. The measured time will be the number of seconds that it takes to decompose one second of an audio signal. The algorithm has been coded in MATLAB R2012b and was run on an Intel Core 2 Duo CPU at 2.40 GHz with 6.0 GB of RAM and Windows Vista.

4.2 Dataset

LMP was tested using piano samples from the university of Iowa database [14]. Test cases were created with polyphony levels ranging from two to six (1000 test cases for each polyphony level). Each note was cut to 100 ms in length and normalized such that its energy was one before the creation of the test cases. The notes used were sampled as in [3]: uniformly at random from between C3 and B6 inclusive (see Section 6) for a total sample space of four octaves' worth of notes. Care was taken to ensure that a single note can occur only once in each test case. That is, within a single test case, the notes were sampled without replacement, but they were replaced between different test cases. To combine multiple notes, the sum of each individual note's signal is used.

4.3 Parameters

The LMP algorithm has many parameters to set. Through testing, the following parameters were found to be optimal (these values are reproduced in the Appendix in Table 3):

- $windowTime$—25 ms—The length of the timesteps used in decomposition. The longer this window, the more likely there will be a note that either starts or ends in the middle of our window. On the other hand, the shorter this window, the longer the computation time of the algorithm. Thus, this value has been set to the greatest value at which there is no dropoff in accuracy.

- $w(t)$—1—The mother window used for the Gabor atoms in the dictionary. As noted previously, this dictates the general shape of the atom. In general, the longer the $windowTime$, the more this function affects the resulting decomposition. Musical signals have properties called attack and sustain, which refer to how quickly the note becomes audible and how quickly the dies off, respectively. With a very short $windowTime$, however, there is no need to worry about any change in amplitude of a partial signal within that window. Setting this value to 1 also improves the computation time of the algorithm.

- $numPartials$—8—The number of partials to test for each atom. The lower this value, the faster the computation of the algorithm. However, the larger this value, the more accurate it is. If this value is set too low, there may be audible partials whose signals are not removed from the input signal. That may cause interference with other notes in the same timestep. Thus, this value has been set to the smallest value at which there is no dropoff in accuracy.

- T_{p1}—.002—The coefficient of the first partial of an atom must be greater than this value in order for that atom to be classified as present. The greater this value, the faster the computation of the algorithm, since computation may be stopped on any atom whose first partial does not reach this threshold. However, if it is set too high, there may be atoms present in a given timestep that are not identified (false negatives). Therefore, T_{p1} has been set to the greatest value at which there is no dropoff in accuracy.

- $firstZero$—2—The first partial in an atom that may have a zero coefficient for that atom to be classified as present. $firstZero$ is a whole number in the interval $[2, numPartials + 1]$. A setting of 2 results in this parameter having no effect, while a setting of $numPartials + 1$ makes it necessary for all of an atom's partials to be present in a timestep for that atom to be classified as present. If this value is too high, there will be a large number of false negatives. On the other hand, if it is too low, there will be a great number of false positives. This value was left at two instead of the heuristic simply being removed because it may have implications in future work when using a different set of heuristics.

- $minMax$—.0082—The maximum coefficient of an atom must be greater than this value for that atom to be classified as present. Setting $minMax$ too high will result in a large number of false negatives, while setting it too low will result in a large number of false positives.

- $minTotal$—0—The sum of the coefficients of an atom must be greater than this value for that atom to be classified as present. Setting $minTotal$ too high will result in a large number of false negatives, while setting it too low will result in a large number of false positives. This value has been left at zero instead of the heuristic simply being removed because it may have implications in future work when using a different set of heuristics.

4.4 Results

In testing, LMP's total error value was lower than that of HMP, but not quite as low as that of HMP SS for every polyphony level. In general, LMP's total error increases slightly for every increase in polyphony, whereas HMP's total error decreases and HMP SS's total error remains fairly steady. These results suggest that more heuristics might by useful to make LMP more robust in regards to polyphony. Taking a closer look at the three components of this error value will help to discover exactly where such improvements might be made. (See Figure 4).

LMP's substitution error level lies between that of HMP and HMP SS for all polyphony levels.

In general, this value increases across polyphony levels in a manner that is similar to the increase of E_{sub} in both HMP and HMP SS, though slightly slower than the substitution error increase of HMP. This suggests that LMP's new heuristics have improved upon the HMP algorithm, though not to the level of improvement that HMP SS accomplishes. This is expected, since LMP applies heuristics during runtime instead of in post-processing. However, the improvements suggest that perhaps more heuristics could be found which can also be applied during runtime that might improve upon this result even further.

LMP's miss error level is greater than that of both HMP and HMP SS for all polyphony levels; however, its false alarm error level is less than that of both HMP and HMP SS for all polyphony levels. The combination of these two differences suggest that LMP excludes more notes than either HMP or HMP SS due to its heuristics. All of the heuristics used in LMP are exclusive. That is, each one decides whether a given atom is *not* present in a given signal. Perhaps this miss error level could be improved if some heuristics can be found which are able to make the opposite decision: that an atom is definitely present in a given signal.

The *Acc* value of LMP lies between that of HMP and HMP SS (see Figure 3) for all polyphony levels. In general, LMP's accuracy decreases slightly with each increase in polyphony, as does HMP's accuracy. HMP SS's, on the other hand, remains relatively steady with each polyphony increase.

The speed of LMP scales very well with an increase of polyphony. Its speed in computation time per signal time varies between 1 and 2 seconds for polyphony levels of one through six. In general, a slight slowdown is seen with every increase in polyphony, with a linear increase on the order of $polyphony/10$. This most likely occurs because, even though the time complexity does not depend on the polyphony level of a musical signal, more computation must still be done during the decomposition. Since more notes are present, there are less opportunities to use the heuristics of LMP to make a quick decision on each atom, and the computation time increases.

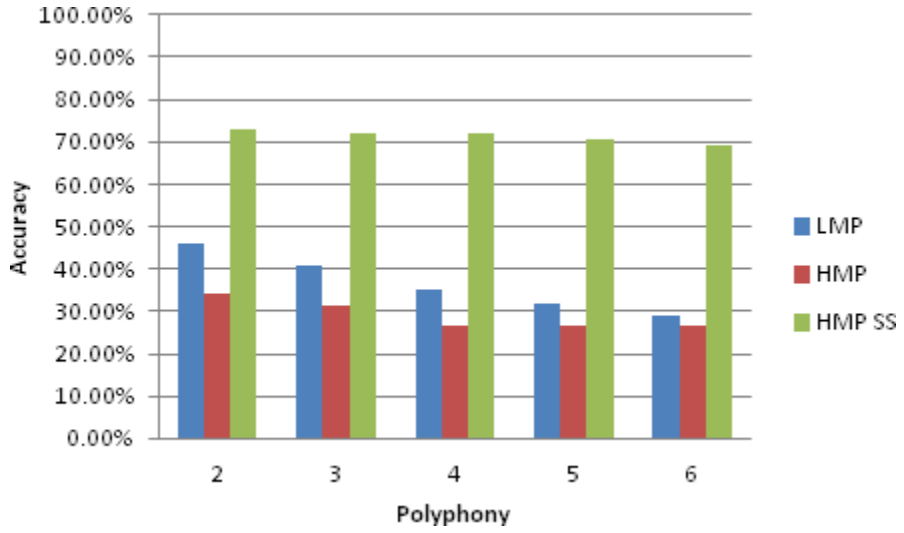

Figure 3: Accuracy values of LMP, compared with those reported in [3] for HMP and HMP SS. The accuracy value varies by polyphony level, from two to six.

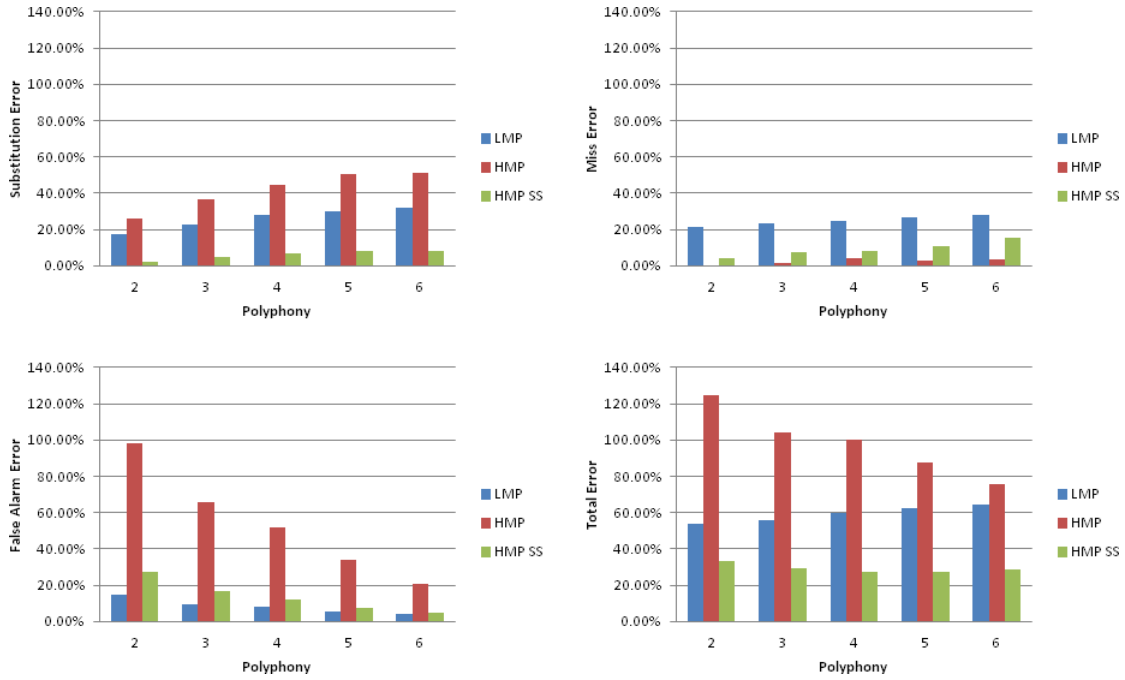

Figure 4: Error values for LMP, compared with those reported in [3] for HMP and HMP SS. The top-left chart compares E_{sub} values; the top-right chart compares E_{miss} errors; the bottom-left chart compares E_{fa} values; the bottom-right chart compares E_{tot} values. Each chart shows those values varied by polyphony level, from two to six.

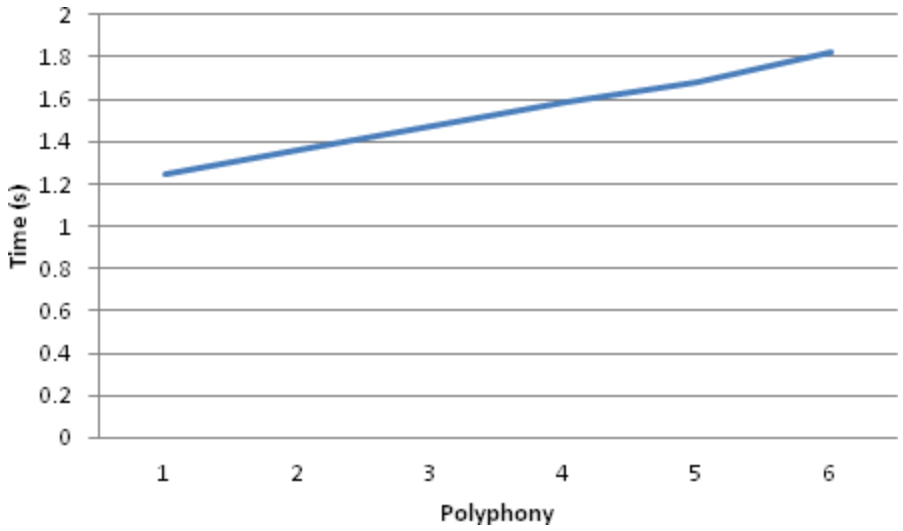

Figure 5: Speed values of LMP, reported by polyphony level, from one to six. The vertical axis gives the length of time in seconds it takes the algorithm to decompose one second of a signal at each level of polyphony.

5 Conclusion

While LMP shows promising results, there is still room for improvement. The algorithm does improve upon HMP's error and accuracy rates, but not to the level that HMP SS does. If LMP's error and accuracy rates can be improved upon further (i.e. with additional or more precise heuristics) there would be considerable real-world applications for the algorithm due to its low time complexity. Additionally, developing functions to tune some of its parameters dynamically would make the algorithm much more robust in many different environments. It is also possible that the additional use of probabilities to predict the note sequence of a musical signal or the use of data mining techniques on either the coefficients found through LMP or the original signal itself could offer a significant improvement over LMP's current accuracy and error rates. It may also be useful to look into solving the music transcription algorithm using a non Matching Pursuit based algorithm. Possibilities include he use of Fourier or Laplace transforms to decompose a signal. Future work should also be done to expand LMP's functionality. Some possibilities include the development of algorithms that are able to detect the tempo of a given musical signal or the style in which that music is played. With the addition of such algorithms, the creation of a successful music transcription program with real-world applications may not be too far off.

6 Appendix

Table 1: Common symbols used in this paper

$g_{s,u,\xi}(t)$	The Gabor atom classified by s, u, and ξ as a function of time
$w(t)$	The mother window of a Gabor atom
s	The scale of a Gabor atom
u	The time around which a Gabor atom is centered
ξ	The modulating frequency of a Gabor atom
$h(t)$	A harmonic atom, as a function of time
ξ_0	The fundamental frequency of a harmonic atom. That is, the ξ of its first partial
α_k	The coefficient of the kth partial of a harmonic atom
K	The number of partials of a harmonic atom
$f(t)$	An audio signal as a function of time
$R_i(t)$	The residual signal in the ith loop of the Matching Pursuit algorithm, as a function of time
D	A dictionary of atoms
D_{n_c}	The smoothed distance of a *way* in HMP SS
Acc	The accuracy metric used in this paper
E_{sub}	A metric corresponding to the number of substitution errors made
E_{miss}	A metric corresponding to the number of miss errors made
E_{fa}	A metric corresponding to the number of false alarm errors made
E_{tot}	A metric corresponding to the number of total errors made

Table 2: Ratios of the fundamental frequencies for musical note intervals within an octave

Note Difference	ξ_0 **Ratio**	**Interval Name**
1	16:15	Minor second
2	9:8	Major second
3	6:5	Minor third
4	5:4	Major third
5	4:3	Perfect fourth
6	45:32	Augmented fourth
7	3:2	Perfect fifth
8	8:5	Minor sixth
9	5:3	Major sixth
10	16:9	Minor seventh
11	15:8	Major seventh
12	2:1	Perfect octave

Table 3: LMP parameter settings

$windowTime$	25
$w(t)$	1
$numPartials$	8
T_{p1}	.002
$firstZero$	2
$minMax$	.0082
$minTotal$	0

Table 4: The notes of a standard 88 key piano

Key	Name	ξ_0(Hz)	Key	Name	ξ_0(Hz)	Key	Name	ξ_0(Hz)
88	C8	4186.01	58	F♯5/G♭5	739.989	30	D3	146.832
87	B7	3951.07	57	F5	698.456	29	C♯3/D♭3	138.591
86	A♯7/B♭7	3729.31	56	E5	659.255	28	C3	130.813
85	A7	3520.00	55	D♯5/E♭5	622.254	27	B2	123.471
84	G♯7/A♭7	3322.44	54	D5	587.330	26	A♯2/B♭2	116.541
83	G7	3135.96	53	C♯5/D♭5	554.365	25	A2	110.000
82	F♯7/G♭7	2959.96	52	C5	523.251	24	G♯2/A♭2	103.826
81	F7	2793.83	51	B4	493.883	23	G2	97.9989
80	E7	2637.02	50	A♯4/B♭4	466.164	22	F♯2/G♭2	92.4986
79	D♯7/E♭7	2489.02	49	A4	440.000	21	F2	87.3071
78	D7	2349.32	48	G♯4/A♭4	415.305	20	E2	82.4069
77	C♯7/D♭7	2217.46	47	G4	391.995	19	D♯2/E♭2	77.7817
76	C7	2093.00	46	F♯4/G♭4	369.994	18	D2	73.4162
75	B6	1975.53	45	F4	349.228	17	C♯2/D♭2	69.2957
74	A♯6/B♭6	1864.66	44	E4	329.628	16	C2	65.4064
73	A6	1760.00	43	D♯4/E♭4	311.127	15	B1	61.7354
72	G♯6/A♭6	1661.22	42	D4	293.665	14	A♯1/B♭1	58.2705
71	G6	1567.98	41	C♯4/D♭4	277.183	13	A1	55.0000
70	F♯6/G♭6	1479.98	40	C4	261.626	12	G♯1/A♭1	51.9131
69	F6	1396.91	39	B3	246.942	11	G1	48.9994
68	E6	1318.51	38	A♯3/B♭3	233.082	10	F♯1/G♭1	46.2493
67	D♯6/E♭6	1244.51	37	A3	220.000	9	F1	43.6535
66	D6	1174.66	36	G♯3/A♭3	207.652	8	E1	41.2034
65	C♯6/D♭6	1108.73	35	G3	195.998	7	D♯1/E♭1	38.8909
64	C6	1046.50	34	F♯3/G♭3	184.997	6	D1	36.7081
63	B5	987.767	33	F3	174.614	5	C♯1/D♭1	34.6478
62	A♯5/B♭5	932.328	32	E3	164.814	4	C1	32.7032
61	A5	880.000	31	D♯3/E♭3	155.563	3	B0	30.8677
60	G♯5/A♭5	830.609				2	A♯0/B♭0	29.1352
59	G5	783.991				1	A0	27.5000

Print References

[1] J. Bello, G. Monti, and B David. Techniques for automatic music transcription. *International Symposium on Music Information Retrieval (ISMIR)*, 2000.

[2] A.H. Benade. *Fundamentals of Musical Acoustics*. Oxford University Press, 1976.

[3] F.J. Canadas-Quesada, P. Vera-Candeas, N. Ruiz-Reyes, R. Mata-Campos, and J.J. Carabias-Orti. Note-event detection in polyphonic musical signals based on harmonic matching pursuit and spectral smoothness. *Journal of New Music Research*, 37(3):167–183, 2008.

[4] J.J. Carabias-Orti, P. Vera-Candeas, F.J. Canadas-Quesada, and N. Ruiz-Reyes. Music scene-adaptive harmonic dictionary for unsupervised note-event detection. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3), March 2010.

[5] Simon Dixon. On the computer recognition of solo piano music. *Proceedings of Australasian Computer Music Conference*, 2000.

[6] R. Gribonval and E. Bacry. Harmonic decomposition of audio signals with matching pursuit. *IEEE Trans. Signal Processing*, 51(1):349–352, Mar 2004.

[7] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 6:3089–3092, 1999.

[8] A. Klapuri and M. Davy. *Signal Processing Methods for Music Transcription*. New York: Springer Science + Business Media LLC, 2004.

[9] Daniel Levitin. *This is Your Brain on Music*. Dutton Books, 2006.

[10] B.T. Lilly and Kuldip K. Paliwal. Robust speech recognition using singular value decomposition based speech enhancement. *TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, 1:257–260, 1997.

[11] S. Mallat and Z. Zhang. Matching pursuit with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41:3397–3415, Dec 1993.

[12] M. Plumbley, S. Abdallah, J. Bello, M. Davies, G. Monti, and M. Sandler. Automatic music transcription and audio source separation. *Cybernetics and Systems*, 33(6):603–627, 2002.

[13] M.Q. Wang and Stephen J. Young. Speech recognition using hidden markov model decomposition and a general background speech model. *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, 1:253–256, 1992.

Other References

[14] Lawrence Fritts. Music instrument samples, 2011. Available online at: `http://theremin.music.uiowa.edu/MIS.html`.

[15] National Institute of Standards and Technology. Rich transcription meeting recognition evaluation plan, 2004. Available online at: `http://www.itl.nist.gov/iad/mig//tests/rt/2004-spring/index.html`.