Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 56561, 15 pages
doi:10.1155/2007/56561

Research Article

Audio Key Finding: Considerations in System Design and Case Studies on Chopin's 24 Preludes

Ching-Hua Chuan[1] and Elaine Chew[2]

[1] *Integrated Media Systems Center, Department of Computer Science, USC Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089-0781, USA*
[2] *Integrated Media Systems Center, Epstein Department of Industrial and Systems Engineering, USC Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089-0193, USA*

Received 8 December 2005; Revised 31 May 2006; Accepted 22 June 2006

Recommended by George Tzanetakis

We systematically analyze audio key finding to determine factors important to system design, and the selection and evaluation of solutions. First, we present a basic system, fuzzy analysis spiral array center of effect generator algorithm, with three key determination policies: nearest-neighbor (NN), relative distance (RD), and average distance (AD). AD achieved a 79% accuracy rate in an evaluation on 410 classical pieces, more than 8% higher RD and NN. We show why audio key finding sometimes outperforms symbolic key finding. We next propose three extensions to the basic key finding system—the modified spiral array (mSA), fundamental frequency identification (F0), and post-weight balancing (PWB)—to improve performance, with evaluations using Chopin's Preludes (Romantic repertoire was the most challenging). F0 provided the greatest improvement in the first 8 seconds, while mSA gave the best performance after 8 seconds. Case studies examine when all systems were correct, or all incorrect.

1. INTRODUCTION

Our goal in this paper is to present a systematic analysis of audio key finding in order to determine the factors important to system design, and to explore the strategies for selecting and evaluating solutions. In this paper we present a basic audio key-finding system, the fuzzy analysis technique with the spiral array center of effect generator (CEG) algorithm [1, 2], also known as FACEG, first proposed in [3]. We propose three different policies, the nearest-neighbor (NN), the relative distance (RD), and the average distance (AD) policies, for key determination. Based on the evaluation of the basic system (FACEG), we provide three extensions at different stages of the system, the modified spiral array (mSA) model, fundamental frequency identification (F0), and post-weight balancing (PWB). Each extension is designed to improve the system from different aspects. Specifically, the modified spiral array model is built with the frequency features of audio, the fundamental frequency identification scheme emphasizes the bass line of the piece, and the post-weight balancing uses the knowledge of music theory to adjust the pitch-class distribution. In particular, we consider several alternatives for determining pitch classes, for representing pitches and keys, and for extracting key information. The alternative systems are evaluated not only statistically, using average results on large datasets, but also through case studies of score-based analyses.

The problem of key finding, that of determining the most stable pitch in a sequence of pitches, has been studied for more than two decades [2, 4–6]. In contrast, audio key finding, determining the key from audio information, has gained interest only in recent years. Audio key finding is far from simply the application of key-finding techniques to audio information with some signal processing. When the problem of key finding was first posed in the literature, key finding was performed on fully disclosed pitch data. Audio key finding presents several challenges that differ from the original problem: in audio key finding, the system does not determine key based on deterministic pitch information, but some audio features such as the frequency distribution; furthermore, full transcription of audio data to score may not necessarily result in better key-finding performance.

We aim to present a more nuanced analysis of an audio key-finding system. Previous approaches to evaluation have

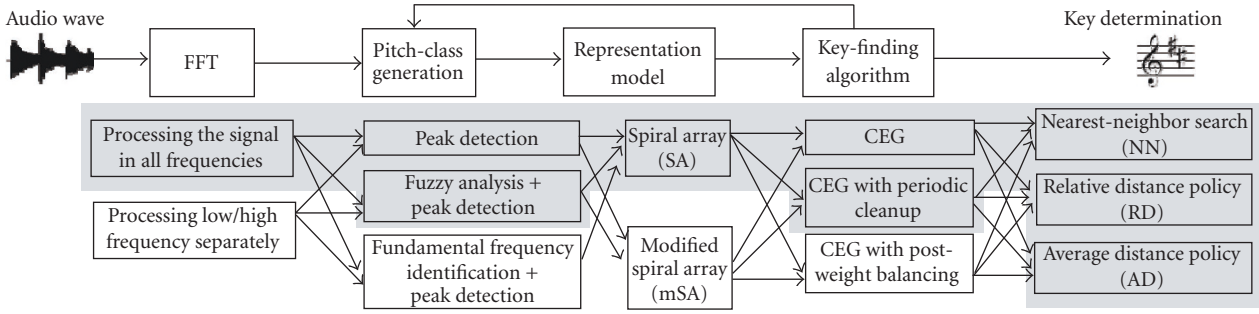


Figure 1: Audio key-finding system (fundamental + extensions).

simply reported one overall statistic for key-finding performance [3, 7–9], which fails to fully address the importance of the various components in the system, or the actual musical content, to system performance. We represent a solution to audio key finding as a system consisting of several alternative parts in various stages. By careful analysis of system performance with respect to choice of components in each stage, we attempt to give a clearer picture of the importance of each component, as well as the choice of music data for testing, to key finding. Our approach draws inspiration from multiple domains: from music theory to audio signal processing. The system components we introduce aim to solve the problem from different viewpoints. The modular design allows us explore the strengths and weaknesses of each alternative option, so that the change in system performance due to each choice can be made clear.

The rest of the paper is organized as follows. Section 1.1 provides a literature review of related work in audio key finding. Section 2 describes the overall system diagram, with new alternatives and extensions. The basic system, the FACEG system, and the three key determination policies, the nearest-neighbor (NN), relative distance (RD), and average distance (AD) policies, are introduced in Section 3. The evaluation of the FACEG system with the three key determination policies follows in Section 4. Two case studies based on the musical score are examined to illustrate situations in which audio key finding performs better than symbolic key finding. Section 5 describes three extensions of the system: the modified spiral array (mSA) approach, fundamental frequency identification (F0), and post-weight balancing (PWB). Qualitative and quantitative analyses and evaluations of the three extensions are presented in Section 6. Section 7 concludes the paper.

1.1. Related work

Various state-of-the-art Audio key-finding systems were presented in the audio key-finding contest for MIREX [10]. Six groups participated in the contest, including Chuan and Chew [11], Gómez [12], İzmirli [13], Pauws [14], Purwins and Blankertz [15], and Zhu (listed alphabetically) [16]. Analysis of the six systems reveals that they share a similar structure, consisting of some signal processing method, audio characteristic analysis, key template construction, query formation, key-finding method, and key determination criteria. The major differences between the systems occur in the audio characteristic analysis, key template construction, and key determination criteria. In Gómez's system, the key templates are precomputed, and are generated from the Krumhansl-Schmuckler pitch-class profiles [5], with alterations to incorporate harmonics characteristic of audio signals. Two systems employing different key determination strategies are submitted by Gómez: one using only the start of a piece, and the other taking the entire piece into account. In İzmirli's system, he constructs key templates from monophonic instruments samples, weighted by a combination of the K-S and Temperley's modified pitch-class profiles. İzmirli's system tracks the confidence value for each key answer, and the global key is then selected as the one having the highest sum of confidence values over the length of the piece. The key templates in Pauws' and Purwins-Blankertz systems are completely data-driven. The parameters are learned from training data. In their systems, the key is determined based on some statistical measure, or maximum correlation. In contrast, Zhu builds a rule-based key-finding system; the rules are learned from the MIDI training data. Further details of our comparative analysis of the systems can be found in [11].

2. SYSTEM DESCRIPTION

Consider a typical audio key-finding system as shown schematically in the top part of Figure 1. The audio key-finding system consists of four main stages: processing of the audio signal to determine the frequencies present, determination of the pitch-class description, application of a key-finding algorithm, and key answer determination. Results from the key-finding algorithm can give feedback to the pitch-class generation stage to help to constrain the pitch-class description to a reasonable set. In this paper, we will consider several possible alternative methods at each stage.

For example, as the basis for comparison, we construct a basic system that first processes the audio signal using the fast Fourier transform (FFT) on the all-frequency signal, then generates pitch-class information using a fuzzy analysis (FA) technique, calculates key results using the CEG algorithm with a periodic cleanup procedure, and applies key determination policy to output the final answer. This basic system, shown in the gray area in Figure 1, is described in detail in

Section 3, followed by an evaluation of the system using 410 classical pieces in Section 4. In Section 5, we present the details of several alternative options for the different stages of the audio key-finding system. In the audio processing stage, the two alternatives we consider are performing the FFT on the all-frequency signal, or separating the signal into low and high frequencies for individual processing. In the pitch-class generation stage, the options are to use the peak detection method with fuzzy analysis, to use peak detection with fundamental frequency identification, or to determine pitch classes using sound sample templates. In the key determination stage, we consider the direct application of the spiral array CEG Algorithm [1, 2], the CEG method with feedback to reduce noise in the pitch-class information, and the CEG method with post-weight balancing. The lower part of Figure 1 shows the various combinations possible, with the alternate modules proposed, in assembling a key-finding system. The in-depth evaluation and qualitative analysis of all approaches are given in Section 6.

3. BASIC SYSTEM

We first construct our basic audio key-finding system as the main reference for comparison. This system, shown in the shaded portions of Figure 1, consists of first an FFT on the audio sample. Then, we use the peak detection method described in Section 3.1 and fuzzy analysis technique proposed in Section 3.2 to generate a pitch-class description of the audio signal. Finally, we map the pitch classes to the spiral array model [1] and apply the CEG algorithm [2] to determine the key. Distinct from our earlier approach, we explore here three key determination policies: nearest-neighbor (NN), relative distance (RD), and average distance (AD). Each method is described in the subsections below. We provide an evaluation of the system in Section 4.

3.1. Peak detection

We use the standard short-term FFT to extract frequency information for pitch identification. Music consists of streams of notes; each note has the properties pitch and duration. Pitch refers to the perceived fundamental frequency of the note. The peak values on the frequency spectrum correspond to the fundamental frequencies of the pitches present, and their harmonics. We use the frequency at the peak value to identify the pitch height, and map the peak spectral magnitude to the pitch weight. Pitches are defined on the logarithmic scale in frequency. A range of frequencies, bounded by the midpoints between the reference frequencies, is deemed acceptable for the recognition of each pitch. We focus our attention on the pitches in the range between C_1 (32 Hz) and B_6 (1975 Hz), which covers most of the common pitches in our music corpus.

We synthesize audio wave files from MIDI at 44.1 kHz and with 16-bit precision. We process audio signal using FFT with nonoverlapped Hanning windows. The window size is set at 0.37 second, corresponding to $N = 2^{14}$ samples. Other sample sizes were tested in the range of 2^{10} to 2^{15} (i.e., window size of 0.0232 to 0.74 second), but these did not perform as well. Let $x(n)$ be the input signal, where $n = 0, \ldots, N-1$. The power spectrum is obtained using the equation

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) W_N^{kn}, \tag{1}$$

where $W_N = e^{-j2\pi/n}$, and $k = 0, 1, \ldots, N-1$. We then calculate the magnitude from the power spectrum as follows:

$$M(k) = ||X(k)|| = \sqrt{(X(k)^2_{\text{real}} + X(k)^2_{\text{img}})}. \tag{2}$$

We set the reference fundamental frequency of A_4 at 440 Hz. Let $h(\text{p})$ be the number of half steps between a pitch p and the pitch A_4. For example, $h(p) = -9$ when $p = C_4$. The reference fundamental frequency of pitch p is then given by

$$F0_{\text{ref}}(p) = 440 \times 2^{h(p)/12}. \tag{3}$$

We employ a local maximum selection (LMS) method [7] to determine the presence of pitches and their relative weights. The midpoint between two adjacent reference fundamental frequencies forms a boundary. We examine $M(k)$ in the frequency band between two such adjacent boundaries surrounding each pitch p. The LMS method is based on two assumptions: (1) a peak value should be larger than the average to its left and to its right in the given frequency band; and (2) only one (the largest) peak value should be chosen in each frequency band. The value $M(k)$ satisfying the above conditions for the frequency band around p, $M^*(p)$, is chosen as the weight of that pitch. This method allows us to consider each pitch equally, so that the system is unaffected by the logarithmic scale of pitch frequencies.

We apply the FFT to the audio signals with two different setups. Under the first option, we process the signal as a whole, with a window size of 0.37 second, to generate the frequency magnitude for each pitch. In the second option, we partition the signals into two subbands, one for higher pitches (frequencies higher than 261 Hz, i.e., pitches higher than C_4), and one for lower ones. We use the same window size to process the higher-pitch signals, and use a larger and overlapped window size for the lower-pitch signals. The window size is relatively large compared to the ones typically used in transcription systems. We give two main reasons for our choice of window size. First, a larger window captures the lower pitches more accurately, which provide the more valuable pitch information in key finding. Second, a larger window smoothes the pitch information, allowing the method to be more robust to pitch variations less important to key identification such as grace notes, passing tones, non-chord tones, and chromatic embellishments.

3.2. Fuzzy analysis technique

The peak detection method described above generates pitch-class distributions with limited accuracy. We design the fuzzy analysis technique to clarify the frequency magnitudes obtained from the FFT, in order to generate more accurate pitch-class distributions for key finding. The main idea behind the fuzzy analysis technique is that one can verify the

existence of a pitch using its overtone series. Hence, we can emphasize the weight of a pitch that has been validated by its overtone series, and reduce the weight of a pitch that has been excluded due to the absence of its strongest overtones.

The problems stem from the fact that mapping of the frequency magnitude directly to pitch weight as input to a key-finding algorithm results in unbalanced pitch-class distributions that are not immediately consistent with existing key templates. We have identified several sources of errors (see [3]) that include uneven loudness of pitches in an audio sample, insufficient resolution of lower-frequency pitches, tuning problems, and harmonic effects. In spite of the unbalanced pitch-class distributions, the key answer generally stays within the ballpark of the correct one, that is, the answer given is typically a closely related key. Some examples of closely related keys are the dominant major/minor, the relative minor/major, and the parallel major/minor keys.

The fuzzy analysis technique consists of three steps. The first step uses information on the overtone series to clarify the existence of the pitches in the lower frequencies. The second step, which we term adaptive level weighting, scales (multiplies) the frequency magnitudes by the relative signal density in a predefined range, so as to focus on frequency ranges containing most information. After the frequency magnitudes have been folded into twelve pitch classes, we apply the third step to refine the pitch-class distribution. The third step sets all normalized pitch class values 0.2 and below to zero, and all values 0.8 and above to one. Details of each step are given below. After the three-part fuzzy analysis technique, we introduce the periodic cleanup procedure for preventing the accumulation of low-level noise over time.

Clarifying lower frequencies

In the first step, we use the overtone series to confirm the presence of pitches below 261 Hz (C_4). Because of the logarithmic scale of pitch frequencies, lower pitches are more closely located on the linear frequency scale than higher ones. The mapping of lower frequencies to their corresponding pitch number is noisy and error prone, especially when using discrete frequency boundaries. There exists greater separation between the reference frequencies of higher pitches, and the mapping of higher frequencies to their corresponding pitches is a more accurate process. For lower pitches, we use the first overtone to confirm their presence and refine their weights.

We use the idea of the membership value in fuzzy logic to represent the likelihood that a pitch has been sounded. Assume that $P_{i,j}$ represents the pitch of class j at register i, for example, middle C (i.e., C_4) is $P_{4,0}$. We consider the pitch range $i = 2, 3, 4, 5, 6$, and $j = 1, \ldots, 12$, which includes pitches ranging from C_2 (65 Hz) to B_6 (2000 Hz). The membership value of $P_{i,j}$ is defined as

$$\operatorname{mem}(P_{i,j}) = \frac{M^*(P_{i,j})}{\max_p \{M^*(p)\}}. \tag{4}$$

Next, we define the *membership negation value* for lower pitches, a quantity that represents the fuzzy likelihood that a pitch is not sounded. Let the membership negation value be

$$\sim \operatorname{mem}(P_{i,j}) = \max\{\operatorname{mem}(P_{i,j+1}), \operatorname{mem}(P_{i+1,j}), \operatorname{mem}(P_{i+1,j+1})\}, \tag{5}$$

where $i = 2, 3$ and $j = 1, \ldots, 12$, because we consider only the lower-frequency pitches, pitches below C_4. This value is the maximum of the membership values of the pitch one half-step above ($P_{i,j+1}$), and the first overtones of the pitch itself ($P_{i+1,j}$), and that of the pitch one half-step above the first overtone. The membership value of a lower-frequency pitch is set to zero if its membership negation value is larger than its membership value:

$$\operatorname{mem}(Pi_j) = \begin{cases} 0 & \text{if } \sim \operatorname{mem}(P_{i,j}) > \operatorname{mem}(Pi_j), \\ \operatorname{mem}(Pi_j) & \text{if } \sim \operatorname{mem}(Pi_j) \le \operatorname{mem}(Pi_j), \end{cases} \tag{6}$$

where $i = 2, 3$ and $j = 1, \ldots, 12$. This step is based on the idea that if the existence of the pitch a half-step above, as indicated by $\operatorname{mem}(P_{i,j+1})$ and $\operatorname{mem}(P_{i+1,j+1})$, is stronger than that of the pitch itself, then the pitch itself is unlikely to have been sounded. And if the signal for the existence of the pitch is stronger in the upper registers, then we can ignore the membership value of the present pitch.

Adaptive level weighting

The adaptive level weight for a given range, a scaling factor, is the relative density of signal in that range. We scale the weight of each pitch class by this adaptive level weight in order to focus on the regions with the greatest amount of pitch information. For example, the adaptive level weight for register i (which includes pitches C_i through B_i), Lw_i, is defined as

$$Lw_i = \frac{\sum_{j=1}^{12} M(P_{i,j})}{\sum_{k=2}^{6} \sum_{j=1}^{12} M(P_{k,j})}, \tag{7}$$

where $i = 2, \ldots, 6$. We generate the weight for each pitch class, $\operatorname{mem}_C(C_j)$, by summing the membership values of that pitch over all registers, and multiplying the result by the corresponding adaptive level weight:

$$\operatorname{mem}_C(C_j) = \sum_{i=2}^{6} Lw_i^* \operatorname{mem}(P_{i,j}), \tag{8}$$

where $j = 1, \ldots, 12$.

Flatten high and low values

To reduce minor differences in the membership values of important pitch classes, and to eliminate low-level noise, we introduce the last step in this section. We set the pitch-class membership values to one if they are greater than 0.8, and zero if they are less than 0.2 (constants determined from held-out data). This flat output for high membership values prevents louder pitches from dominating the weight.

Periodic cleanup procedure

Based on our observations, errors tend to accumulate over time. To counter this effect, we implemented a periodic cleanup procedure that takes place every 2.5 seconds. In this cleanup step, we sort the pitch classes in ascending order and isolate the four pitches with the smallest membership values. We set the two smallest values to zero, a reasonable choice since most scales consist of only seven pitch classes. For the pitch classes with the third and fourth smallest membership values, we consult the current key assigned by the CEG algorithm; if the pitch class does not belong to the key, we set the membership value to zero as well.

3.3. Spiral array model and the center of effect algorithm

The spiral array model, proposed by Chew in [1], is a three-dimensional model that represents pitches, and any pitch-based objects that can be described by a collection of pitches, such as intervals, chords, and keys, in the same three-dimensional space for easy comparison. On the spiral array, pitches are represented as points on a helix, and adjacent pitches are related by intervals of perfect fifths, while vertical neighbors are related by major thirds. The pitch spiral is shown on Figure 2(a). Central to the spiral array is the idea of the center of effect (CE), the representing of pitch-based objects as the weighted sum of their lower-level components. The CE of a key is shown on Figure 2(b). Further details for the construction of the spiral array model are given in [1, 2].

In the CEG algorithm, key selection is performed by a nearest-neighbor search in the spiral array space. We will call this the nearest-neighbor (NN) policy for key determination. The pitch classes in a given segment of music is mapped to their corresponding positions in the spiral array, and their CE generated by a linear weighting of these pitch positions. The algorithm identifies the most likely key by searching for the key representation closest to the CE. The evolving CE creates a path that traces its dynamically changing relationships to the chord and key structures represented in the model [17].

Previous applications of the CEG algorithm have used the relative pitch durations as the CE weights, either directly [2] or through a linear filter [17]. Here, in audio key finding, we use the normalized pitch-class distribution derived from the frequency weights to generate the CE.

One more step remains to map any numeric representation of pitch to its letter name for key analysis using the spiral array. The pitch spelling algorithm, described in [18, 19], is applied to assign letter names to the pitches so that they can be mapped to their corresponding representations in the spiral array for key finding. The pitch spelling algorithm uses the current CE, generated by the past five seconds of music, as a proxy for the key context, and assigns pitch names through a nearest-neighbor search for the closest pitch-class representation. To initialize the process, all pitches in the first time chunk are spelt closest to the pitch class D in the spiral array, then the CE of these pitches is generated, and they are respelt using this CE.

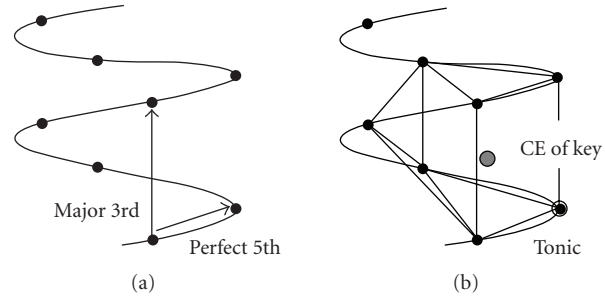


Figure 2: (a) Pitch spiral in the spiral array model, and (b) the generating of a CE to represent the key.

If $|d_{j,t} - d_{k,t}| < d$,
  If $\overline{d_{i,t}} < \overline{d_{k,t}}$,
    choose key i as the answer;
  Else, choose key k as the answer;
Else, choose key i as the answer.

Algorithm 1: Related distance policy.

3.4. Key determination: relative distance policy

In the audio key-finding systems under consideration, we generate an answer for the key using the cumulative pitch-class information (from time 0 until the present) at every analysis window, which eventually evolves into an answer for the global key for the whole duration of the music example. Directly reporting the key with the shortest distance to CE as the answer at each analysis window, that is, the NN policy, does not fully reflect the extent of the tonal analysis information provided by the spiral array model. For example, at certain times, the CE can be practically equidistant from two different keys, showing strong ambiguity in key determination. Sometimes the first key answer (the one with the shortest distance to CE) may result from a local chord change, cadence, or tonicization, and the second answer is actually the correct global key. The next two key determination policies seek to address this problem.

We first introduce the relative distance key determination policy with distance threshold d, notated (RD, d). In the RD policy, we examine the first two keys with the shortest distances to the CE. If the distance difference between the first two keys is larger then the threshold d, we report the first key as the answer. Otherwise, we compare the average distances of the two keys from the beginning to the current time chunk. The one with shorter average distance is reported as the answer.

Formally, let $d_{i,j}$ be the distance from the CE to key i at time j, where $i = 1, \ldots, 24$. At time t, assume that keys i and k are the closest keys to the CE with distances $d_{j,t}$ and $d_{k,t}$, respectively. Algorithm 1 describes the (RD,d) policy in pseudocode.

The RD policy attempts to correct for tonal ambiguities introduced by local changes. The basic assumption underlying this method is that the NN policy is generally correct.

In cases of ambiguity, which are identified as moments in time when the first and second closest keys are less than the threshold distance apart from each other, then we use the average distance policy to determine which among the two most likely candidates is the best choice. The next section describes the average distance policy in greater detail.

In this paper, we test two values of d. The choice of d depends on the distance between keys in the spiral array. Assume d_1 denotes the shortest, and d_2 the second shortest, distance between any two keys in the spiral array model. Then we constrain the value of d to the range

$$\alpha d_1 \leq d \leq \beta d_2, \tag{9}$$

where $0 < \alpha, \beta \leq 0.5$. In this paper we set both α and β equal to 0.25. Intuitively, this means that the CE should lie in the center half of the line segment connecting two very close keys, if there is ambiguity between the two keys.

3.5. Key determination: average distance policy

The average distance key determination policy (AD) is inspired by the method used by İzmirli in his winning submission to the MIREX 2005 audio key-finding competition [13, 20], where only the global key answer was evaluated. İzmirli's system tracks the confidence value for each key answer, a number based on the correlation coefficient between the query and key template. The global key was then selected as the one having the highest sum of confidence values over the length of the piece.

In the spiral array, the distance from each key to the current CE can serve as a confidence indicator for that key. In the AD policy, we use the average distance of the key to the CE at all time chunks to choose one key as the answer for the whole testing duration of the piece.

Formally, at time t, if

$$\overline{d_{j,t}} = \underset{i=1,\ldots,24}{\text{MIN}} \left(\overline{d_{i,t}}\right), \quad \text{choose key } j \text{ as the answer.} \tag{10}$$

We explore the advantages and the disadvantages of the (RD, d) and (AD) policies in the rest of the paper.

4. EVALUATION OF THE BASIC SYSTEM

In this paper we test the systems in two stages. In the first stage, we use 410 classical music pieces to test the basic systems described in Section 3, that is, the audio key-finding system using fuzzy analysis and the CEG algorithm, with the three key determination policies, (NN), (RD, d), and (AD). Both the local key answer (the result at each unit time) and the global key answer (one answer for each sample piece) are considered for the evaluation. The results are analyzed and classified by key relationships, as well as stylistic periods. At the second stage of the evaluation, we use audio recordings of 24 Chopin Preludes to test the extensions of the audio key-finding system.

We choose excerpts from 410 classical music pieces by various composers across different time and stylistic periods, ranging from Baroque to Contemporary, to evaluate the methods. Table 1 shows the distribution of pieces across the various classical genres. Most of the chosen pieces are concertos, preludes, and symphonies, which consist of polyphonic sounds from a variety of instruments. We regard the key of each piece stated explicitly by the composer in the title as the ground truth for the evaluation. We use only the first fifteen seconds of the first movement so that the test samples are highly likely to remain in the stated key for the entire duration of the sample.

Table 1: Results analysis of global key answers across periods obtained from fuzzy analysis technique and CEG algorithm.

Categories	Baro*	Class	Early roman.	Roman	Late roman.	Con.
CORR **	80	95.7	72.4	76	72.9	82.8
DOM	16.8	0	25.3	8	5.9	0
SUBD	0	0.9	0	4	0	1
REL	0	0.9	0	6	5	3
PAR	2.1	1.7	0	2	1	1
Others	0	0.9	2.3	4	1	0
Num.	95	115	87	50	34	29

* Baro = baroque, Class = classical, Roman = romantic, Con. = contemporary.
** CORR = correct, DOM = dominant, SUBD = subdominant, REL = relative, PAR = parallel, Other = other.

In order to facilitate comparison of audio key finding from symbolic and audio data, we collected MIDI samples from http://www.classicalarchieves.com, and used the Winamp software with 44.1 kHz sampling rate to render MIDI files into audio (wave format). We concurrently tested four different systems on the same pieces. The first system applied the CEG algorithm with the nearest-neighbor policy, CEG(NN) to MIDI files, the second applied the CEG algorithm with the nearest-neighbor policy and fuzzy analysis technique, FACEG(NN), and the third and the fourth are similar to the second with the exception that they employ the relative distance policy in key determination, FACEG(RD, d), with different distance thresholds. The last system, FACEG(AD), applies the relative distance policy with average distances instead.

Two types of results are shown in the following sections. Section 5.1 presents the averages of the results of all periods over time for the four systems. Each system reported a key answer every 0.37 second, and the answers are classified into five categories: correct, dominant, relative, parallel, and others. Two score-based analyses are given to demonstrate the examples in which audio key-finding system outperforms the MIDI key-finding system that takes explicit note information as input. In Section 5.2, the global key results given by the audio key-finding system with fuzzy analysis technique and CEG algorithm are shown for each stylistic period.

4.1. Overall results over time

Figure 3(a) shows the average correct rates of the five systems over time on 410 classical music pieces. We can observe that

in the second half of the testing period, from 8 to 15 seconds, four of the systems, all except FACEG(AD), achieve almost the same results by the percentage correct measure.

The relative distance key determination policy using average distance FACEG(AD) performed best. Its correct percentage is almost 10% higher than the other systems from 8 to 15 seconds. Notice that the improved correct rate of FACEG(AD) is mainly due to the reduction of dominant and relative errors shown in Figures 3(b) and 3(c). The relative distance policy using threshold distance (RD, d) slightly outperforms the systems with only the nearest-neighbor (NN) policy in audio key finding. The results of the systems with the RD and AD policies maintain the same correct rates from 5 seconds to the end. The longer-term stability of the results points to the advantage of the RD and AD policies for choosing the global key.

The CEG(NN) system outperforms all four audio systems in the first five seconds. The RD policy even lowers the correct rate of the FACEG(NN) audio key-finding system. The results show that audio key-finding system requires more time at the beginning to develop a clearer pitch-class distribution. The RD policy may change correct answers to the incorrect ones at the beginning if the pitch-class information at the first few seconds is ambiguous.

Figures 3(b) to 3(e) illustrate the results in dominant, relative, parallel, and others categories. Most differences between the CEG(NN) system and the FACEG audio key-finding systems can be explained in the dominant and parallel errors, shown in Figures 3(b) and 3(d). We can use music-theoretic counterpoint rules to explain the errors. In a composition, doubling of a root or the fifth of a chord is preferred over doubling the third. The third is the distinguishing pitch between major and minor chords. When this chord is the tonic, the reduced presence of thirds may cause a higher incidence of parallel major/minor key errors in the first four seconds. For audio examples, the third becomes even weaker because the harmonics of the root and the fifth are more closely aligned, which explains why audio key-finding systems have more parallel errors than the MIDI key-finding system CEG(NN). The ambiguity between parallel major and minor keys subsides once the system gathers more pitch-class information.

In the relative and other error categories, shown in Figures 3(c) and 3(e), the audio key-finding systems perform slightly better than the MIDI key-finding system. We present two examples with score analysis in Figures 4 and 5 to demonstrate how the audio key-finding systems—FACEG (NN), FACEG (RD, 0.1), FACEG (RD, 0.17), FACEG (AD)—outperform the MIDI key-finding system.

4.2. When audio outperforms symbolic key finding

Figure 4 shows the first four measures of Bach's *Double Concerto in D minor* for two violins, BWV1043. For the whole duration of the four measures, all audio systems give the correct key answer, D minor. In contrast, the MIDI key-finding system returns the answer F major in the first two measures,

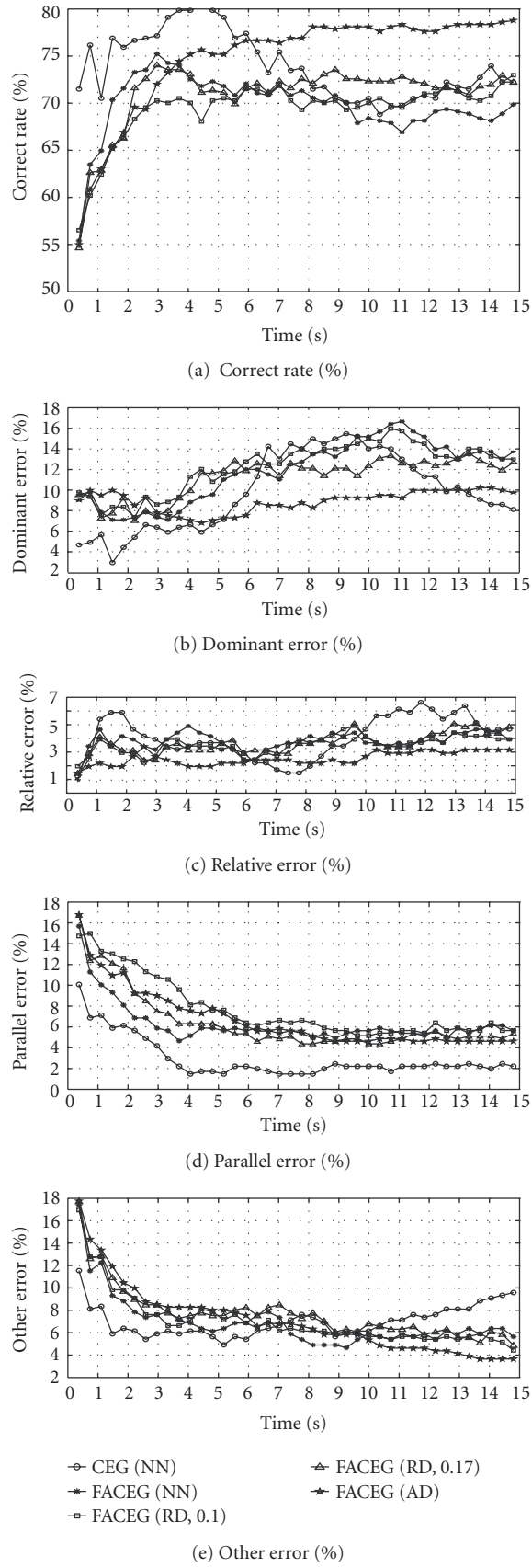


Figure 3: Results of first fifteen seconds of 410 classical pieces, classified into five categories.

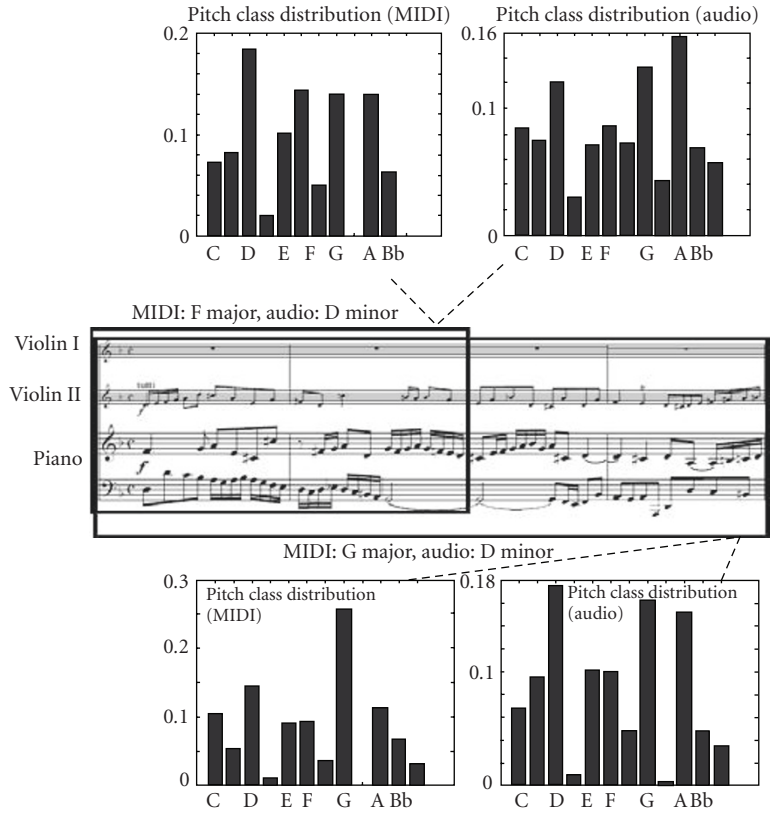


Figure 4: Pitch-class distribution of Bach concertos in D minor.

then changes the answer to G major at the end. We can explain the results by studying the pitch-class distributions for both the MIDI and audio systems at the end of the second and fourth measures.

The pitch-class distribution of the MIDI system at the second measure does not provide sufficiently significant differences between the pitch sets belonging to F major and D minor; however, the high weight on pitch class A, the second harmonic of the pitch D, in the corresponding distribution derived from audio helps to break the tie to result in the answer, D minor. At the end of the second measure and the beginning of the third, there are two half-note G's in the bass line of the piano part. These relatively long notes bias the answer towards G major in the MIDI key-finding system. The audio key-finding systems are not affected by these long notes because the effect of the overlapping harmonics results in a strong D, and a not-as-high weight on G in the pitch-class distribution.

We give another example in Figure 5, which shows the first eight measures of Brahms' *Symphony No. 4 in E minor*, Op.98. Both the MIDI and audio key-finding systems report correct answers for the first six measures. At measures 6 through 8, the chords progress from vi (pitches C, E, G) to III (pitches G, B, D) to VII (pitches D, $F^{\#}$, A) in E minor, which correspond to the IV, I, and V chords in G major. After these two measures the answer of the MIDI key-finding system becomes G major. This example shows that having explicit information of only the fundamental pitches present makes the MIDI key-finding system more sensitive to the local tonal changes.

4.3. Results of global key across periods

We use the average of the distances between the CE and the key over all time chunks to determine the global key. The one which has the shortest average distance is chosen to be the answer. Table 1 lists the results of global key answers, broken down by stylistic periods, obtained from the audio key-finding, FACEG(AD), systems. The period classifications are as follows: Baroque (Bach and Vivaldi), Classical (Haydn and Mozart), Early Romantic (Beethoven and Schubert), Romantic (Chopin, Mendelssohn, and Schumann), Late Romantic (Brahms and Tchaikovsky), and Contemporary (Copland, Gershwin, and Shostakovich). The results themselves are separated into six categories as well: Correct, Dominant, Subdominant, Relative, Parallel, and Other (in percentages).

Notice that in Table 1, the results vary significantly from one period to another. The best results are those of the Classical period, which attains the highest correct percentage rate of 95.7% on 115 pieces. The worst results are those of pieces from the Early Romantic period, having many more

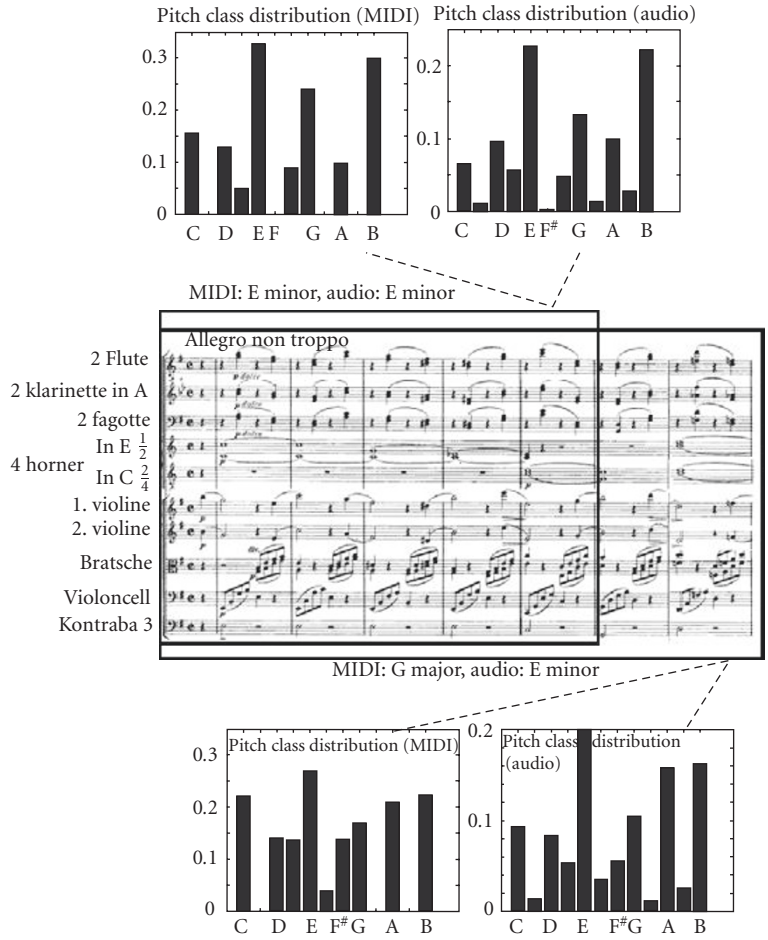


Figure 5: Pitch-class distributions of Brahms symphony number 4 in E minor.

errors on the dominant and others categories. The variances in Table 1 show clearly the dependency between the system performance and the music style. Lower correct rates could be interpreted as an index of the difficulty of the test data.

5. SYSTEM EXTENSIONS

In this section, we propose three new alternatives for the pitch-class generation and the key-finding stages to improve audio key finding as was first presented in the system outline given in Figure 1. These methods include modifying the spiral array model using sampled piano audio signals, fundamental frequency identification, and post-weight balancing. The three approaches affect different stages in the prototypical system, and use different domains of knowledge. In the first alternative, we modify the spiral array model so that the positions of the tonal entities reflect the frequency features of audio signals. The second alternative affects pitch-class generation; we use the information from the harmonic series to identify the fundamental frequencies. The third method of post-weight balancing is applied after the key-finding algorithm; it uses the key-finding answer to refine the pitch-class distribution. Each of the three approaches is described in the subsections to follow.

5.1. Modified spiral array with piano signals

Since the pitch-class distribution for each audio sample is constructed using the frequency magnitudes derived from the FFT, in order to compare the CE of this distribution to an object of the same type, we propose to prepare the spiral array to also generate tonal representations based on audio-signal frequency features. In this section, we describe how we modify the major and minor key spirals so that the positions of key spirals are constructed according to the frequency features of the audio signals. The advantages of the proposed modification are that the modified spiral array can manage the diversity of the frequency features of audio signals, and tolerate the errors from pitch detection method. A similar idea is proposed by İzmirli to modify the Krumhansl-Schmuckler key-finding method to address audio signals in [13].

Figure 6 shows the sequence of steps for remapping the spiral array representations for audio. The mapping uses the

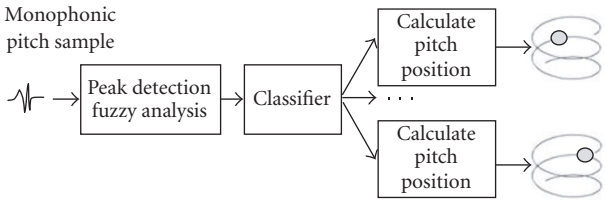


Figure 6: System diagram of reconstructing pitches in spiral array model.

frequency distribution of monophonic pitch samples to first classify pitches into subclasses based on their harmonic profile, then calculates the new position of each pitch for each subclass. The monophonic pitch samples, piano sounds from Bb_0 to C_8, are obtained from the University of Iowa Musical Instrument Samples online [21]. The classification step is essential because tone samples from different registers exhibit different harmonic characteristics. Hence, the representations are regenerated for each subclass.

Formally, for each monophonic pitch sample, we apply the peak detection method and fuzzy analysis technique to generate a pitch-class distribution for that pitch, $\text{mem}(C_j)$, $i = 1, 2, \ldots, 12$. Each pitch then is classified into several subclasses according to the pitch-class distribution. The classification can be done by any existing classifiers, such as k nearest neighbors. The classification must satisfy the constraint that each class consists of pitches that are close to one another. This constraint is based on the assumption that pitches in the same range are likely to have similar pitch-class distributions. For the purposes of the tests in this paper, we classify the pitches into five classes manually.

The new position of the pitch representation in the spiral array, for each subclass, is recomputed using these weights. Assume $\mathbf{P}_i$ represents the original position of pitch class i in the spiral array model. The new position of pitch class i, $\mathbf{P}_i'$, is defined as

$$Pi' = \frac{1}{n} \sum_{j=1}^{12} \text{mem}\,(C_j) \times \mathbf{p}_j, \tag{11}$$

where $j = 1, \ldots, 12$ and n is the size of the subclass. Figure 7 shows conceptually the generating of the new position for pitch class C.

Once we obtain the new position of pitches, we can calculate the new position of keys for each subclass by a weighted linear combination of the positions of the triads. The composite key spirals are generated in real time as the audio sample is being analyzed. We weight the key representation from each subclass in a way similar to that for the level weights method described in Section 3.2. That is to say, the level weight for a given subclass is given by the relative density of pitches from that subclass. The position of each key in a key spiral is the sum of the corresponding key representations for each subclass, multiplied by its respective level weight. Assume $\mathbf{T}_i$ is the original position of key i in the spiral array, the new position of key i, $\mathbf{T}_i'$, is calculated by

$$T_i' = Lw_i \times T_j'', \tag{12}$$

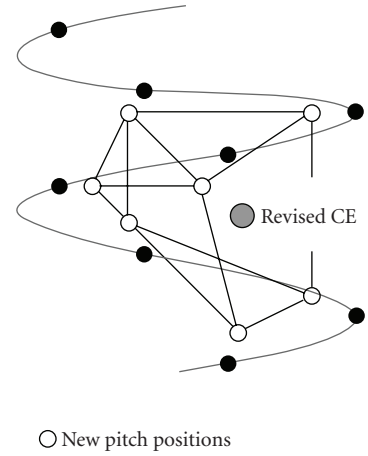


Figure 7: Recalculating pitch position using pitch-class distribution.

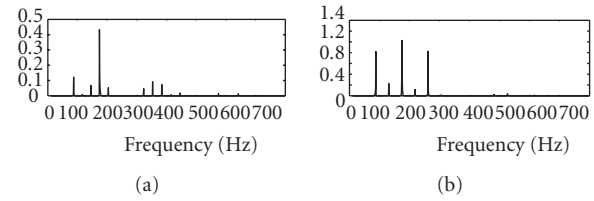


Figure 8: Frequency responses of pitches (a) Bb_0 and (b) F_1 using FFT.

where Lw_i is the level weight for subclass i and $\mathbf{T}_j''$ is the composite position for key j in subclass i, $j = 1, \ldots, 24$ for 24 possible keys.

As the final step, we perform the usual nearest-neighbor search between the CE generated by the pitch-class distribution of the audio sample and the key representations to determine the key.

5.2. Fundamental frequency identification

Audio signals from music differ from speech signals in three main aspects: the frequency range, the location of the fundamental frequency, and the characteristic of the harmonic series. Compared to human voices, instruments can sound in a much wider range of frequencies. Furthermore, the lower pitches are typically organized in such a way as to highline the tonal structure of the music sample, while the higher pitches are less important structurally, and may contain many superfluous accidentals. However, the structurally more important lower pitches cannot always be detected using signal processing methods such as the FFT. Also, several lower pitches may generate similar distributions in the frequency spectrum. Missing information in the lower registers seriously compromises the results of key finding. Figure 8 shows the FFT output for pitches Bb_0 and F_1. It is important to note that these two pitches have similar frequency distributions, yet neither of their fundamental frequencies appear in

Table 2: Frequency and pitch relations of seven harmonics.

Frequency ratio	1	2	3	4	5	6	7
Pitch relation***	1	8va	8va + P5	16va	16va + M3	16va + P5	16va + m7
Semitone distance	0	12	19	24	28	31	34

***: P5: perfect fifth, M3: major third, m7: minor seventh.

the FFT. In the case of pitch Bb_0, none of the pitches in the pitch class Bb is presented. This example reveals a key consideration as to why audio key finding frequently suffers from dominant errors. The audio signals of each individual pitch are collected from the piano recordings on the Iowa University website [21].

Many systems for automatic transcription that use fundamental frequency to identify pitch have been proposed recently [22, 23]. The transcription problem requires the extraction of multiple fundamental frequencies of simultaneously sounding pitches. We, instead, are concerned with finding only the lowest pitch in the bass. We use the first seven harmonics to identify each fundamental frequency. The frequency ratio (multiple of the fundamental frequency) and the pitch relation of the harmonic structure are given in Table 2. We use this harmonic structure as a template for locating the fundamental frequencies as follows. Given an audio signal, first we extract the frequencies with the largest and second largest frequency magnitudes. Then we move the harmonic template so as to find all possible ways to cover the two frequencies, and calculate the total number of frequencies that are both in the harmonic template and the extracted frequency spectrum. The highest scoring option gives the location of the fundamental frequency. We employ a heuristic to break ties. Ties happen because not all the harmonics appear for a tone of a given fundamental pitch. When an octave pair is encountered, it is unclear if this represents the fundamental pitch class, or the fifth above the fundamental. The heuristic is based on our observations, and prefers the interpretation of the fifth when finding an octave pair in the lower registers.

Using a window size that is three times larger for low frequencies than higher ones (0.37 second), we tested the above method on monophonic piano samples of pitches ranging from Bb_0 to B_3 obtained from the Iowa University website [21]. For the 38 samples, we successfully identified the fundamental frequencies of 33 pitches, with 4 octave errors and 1 perfect fifth error. The octave error does not affect key finding.

5.3. Post-weight balancing

In audio key finding, unbalanced pitch-class distribution is often obtained using the frequency spectrum derived from the FFT. One particularly problematic example occurs when the weight of a certain pitch class is much higher than the others. The pitch class dominates the weight distribution so much so that the CE is strongly biased by that pitch class, and cannot fairly represent the presence of the other pitch classes.

Let $\mathbf{K_3}$ be the set of three closest keys and $\mathbf{K_2}$ is any subset with two keys of $\mathbf{K_3}$.

(1) If $\mathbf{K_2}$ contains a *relative* major/minor pair, then the tonic of $\mathbf{K_3} \setminus \mathbf{K_2}$ is labeled as overweighted.

(2) If $\mathbf{K_2}$ contains a *parallel* major/minor pair, then the tonic of $\mathbf{K_2}$ is labeled as overweighted.

(3) Overweighted pitch class is assigned the average weight of pitches in $\mathbf{K_3}$.

Algorithm 2: Post-weight balancing.

The relative distance policy in key determination, in which the system compares the distance difference between the first two keys, cannot solve this unbalanced distribution problem. Similarly, one cannot readily eliminate the problem by simply examining the low-level features such as the frequency of the audio signal.

To solve the problem of unbalanced weight distributions, we design a post-weight balancing mechanism. We use high-level knowledge of the relations between keys to determine which pitch class has been weighted too heavily. The post-weight balancing mechanism is based on two principles: (1) if the three closest keys contain a *relative* major/minor pair, then the tonic of the other key is likely overweighted; (2) if the three closest keys contain a *parallel* major/minor pair, then the tonic of the pair is likely overweighted.

Once the overweighted pitch class is identified, we reduce its weight in the pitch-class distribution, and reapply the CEG algorithm to generate a new answer with the adjusted pitch-class distribution. The new answer is then verified again using the post-weight balancing mechanism to specifically disambiguate the relative or parallel major/minor answers. To differentiate between relative major/minor keys, we compare the weights of the respective tonic pitch classes. The one with larger weight is chosen as the answer. To differentiate between parallel major/minor keys, we examine the weights of the nondiatonic pitches in each candidate key. The post-weight balancing algorithm is summarized in Algorithm 2.

6. IN-DEPTH ANALYSIS OF RESULTS

In order to explore the possible approaches for improving audio key-finding system outlined in Section 5, we test five systems on Evgeny Kissin's CD recordings of Chopin's *Twenty-Four Preludes* for piano (ASIN: B00002DE5F). We chose this test set for three reasons: (1) it represents one of

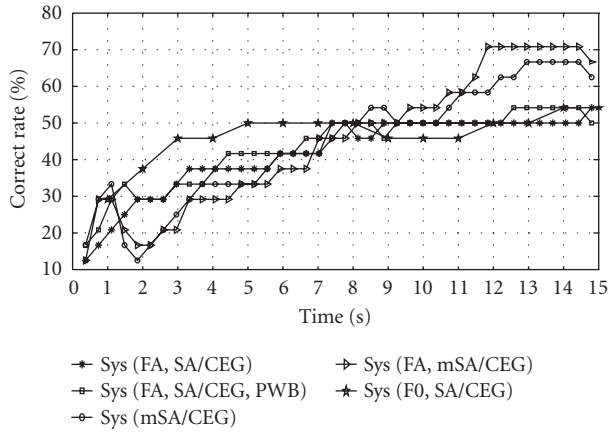


Figure 9: Correct rate of five extended systems on Chopin's 24 Preludes.

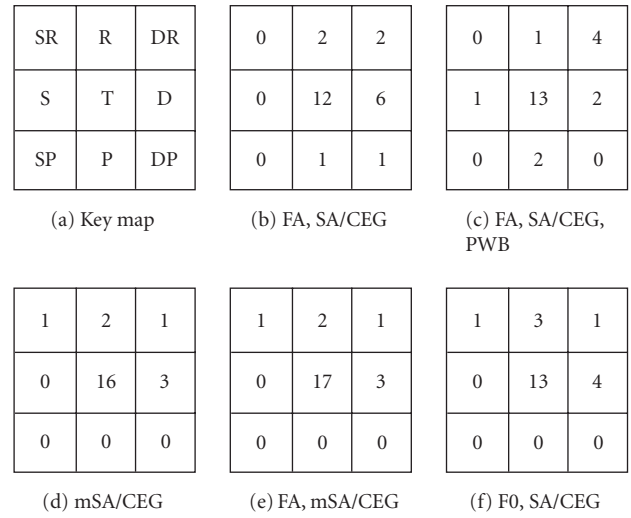


Figure 10: Results of five systems on Chopin's 24 Preludes T: tonic, D: dominant, S: subdominant, R: relative, P: parallel.

the most challenging key-finding datasets we have tested to date—in a previous study (see [3]), we discovered that the results for this test set was farthest from that for MIDI; (2) the audio recording created using an acoustic piano allows us to test the systems' robustness in the presence of some timbral effects; and (3) a minor but aesthetic point, all 24 keys are represented in the collection.

The five systems selected for testing consist of combinations of the approaches described in Sections 3 and 5, with a focus on the three new alternatives introduced in Section 5. We introduce a notation for representing the systems. The five systems are as follows:

(a) the basic system, sys(FA, SA/CEG);
(b) FACEG with post-weight balancing, sys(FA, SA/CEG, PWB);
(c) the modified spiral array with CEG, sys(mSA/CEG);
(d) FACEG with modified spiral array, sys(FA, mSA/CEG);
(e) fundamental frequency identification with CEG, sys(F0, SA/CEG).

We have ascertained in Section 4.1 that the best key determination policy was the AD, average distance policy, which is now employed in all five systems.

The first system, sys(FA, SA/CEG), serves as a reference, a basis for comparison. The second system, sys(FA, SA/CEG, PWB), tests the effectiveness of the post-weight balancing scheme applied to the basic system. The fourth system, sys(FA, mSA/CEG), tests the effectiveness of the modifications to the spiral array based on audio samples, in comparison to the basic system. To further test the power of the modified spiral array, we take away the fuzzy analysis layer from (d) to arrive at the third system, sys(mSA/CEG). The fifth system, sys(F0, SA/CEG), tests the effectiveness of the fundamental frequency identification scheme relative to the basic system.

The overall results for all five extended systems are shown in Figure 9. The system employing fundamental frequency identification with the CEG algorithm, sys(F0, CEG), outperforms the others in the first 8 seconds. The systems using the modified spiral array model, sys(mSA/CEG) and sys(FA, mSA/CEG), achieve the best correct rates after 8 seconds, and significantly higher correct rates from 12 to 15 seconds. In comparison to the original system, sys(FA, CEG), the post-weight balancing used in sys(FA, CEG, PWB) improves the results slightly.

Table 3: Summary of the results of five systems categorizing the pieces by the number of the correct answers.

no. of sys w. corr. ans.	5	4	3	2	1	0
Piece	6, 8, 17,	14, 15,	2, 4,	10,	1, 3,	5, 9,
Prelude no.	19, 20, 24	21, 23	7, 12	22	11, 16	13, 18

Figure 10 shows the cross-section of results at fifteen seconds for the five systems. Each result is represented in a key map, where the tonic (the ground truth) is in the center, and the dominant, subdominant, relative, parallel are to the right and left, and above, and below the tonic, respectively. The number inside each grid is the number of the answers that fall in that category. Answers that are out of the range of the key map are not shown.

Figure 10(b) shows the results of the original system, the fuzzy analysis technique with spiral array CEG algorithm. The results of system extensions are given in Figures 10(c) through 10(f). Notice that the FACEG system with modified spiral array model, Figure 10(e), gets the most correct answers, which implies that including the frequency features within the model could be the most effective way to improve the system. This result is confirmed by Figure 9.

In Table 3, we categorize the pieces by the number of systems that report correct answers. By analyzing pieces with the most or least number of correct answers, we can better understand which kinds of musical patterns pose the greatest difficulty for all systems, as well as the advantages of one system over the others for certain pieces. The results summarized in Table 3 suggest that the ease of arriving at a correct key solution is independent of the tempo of the piece.

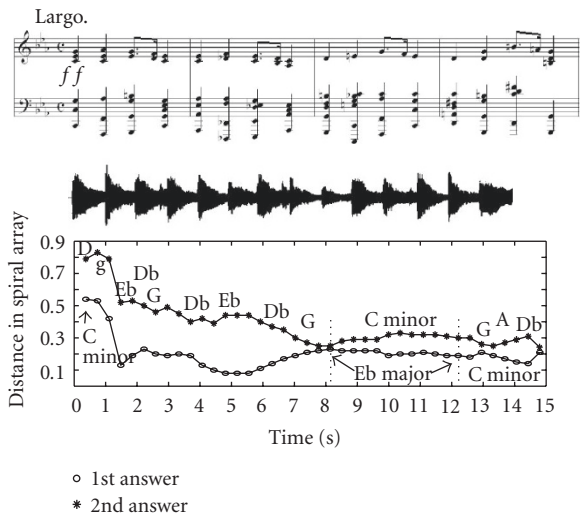


Figure 11: Key answers over time of sys(FA, CEG) on Chopin's Prelude number 20 in C minor.

We now focus on the analysis of some best- and worst-case answers for audio key finding. Chopin's *Prelude No. 20 in C minor* is one of the pieces for which all of the systems gave a correct answer, even though it could be interpreted as being in four different keys in each of the first four bars. Figure 11 shows the first part of the score corresponding to the audio sample analyzed, the audio wave of the first fifteen seconds, and the top two key answers over time, and the distances of these key representations to the CE of the audio sample. In the spiral array model, shorter distances correspond to more certain answers. The results over time presented in Figure 11 show the two key representations closest to the current cumulative CE at each time sample. The closest answer is C minor for most of the time, except from 8.14 to 13.32 seconds, when the closest answer is Eb major, while C minor is relegated to the second closest answer. The graph shows the chord sequence's brief tonicization in Eb major at the end of measure three, before reaching G major at the end of measure four, which acts as the dominant to the key of C minor.

Chopin's *Prelude No. 9 in E major* is an example in which all five systems reported B major as the key, the dominant of E. Figure 12 shows the score for the first part of the piece, the audio wave of the first fifteen seconds, and the top two key answers over time for the four audio systems. From Figures 12(a) to 12(d), we observe the following two common behaviors in all graphs: (1) for most of the time during the fifteen seconds, the top-ranked answer is B major, while the second answer traverses through related keys; and (2) the first and second answers are almost equidistant from the CE in the middle part, between 7 and 8 seconds.

The test duration (first fifteen seconds) corresponds to the first two and half measures of the piece. From the score, the chord progression of the first measure is I-V-I-IV in E major, which unequivocally sets up the key as E major. The dominant errors of all systems can be attributed to the audio attributes of Evgeny Kissin's expressive performance. In the recording, Kissin emphasizes the pitch B_3 at each beat

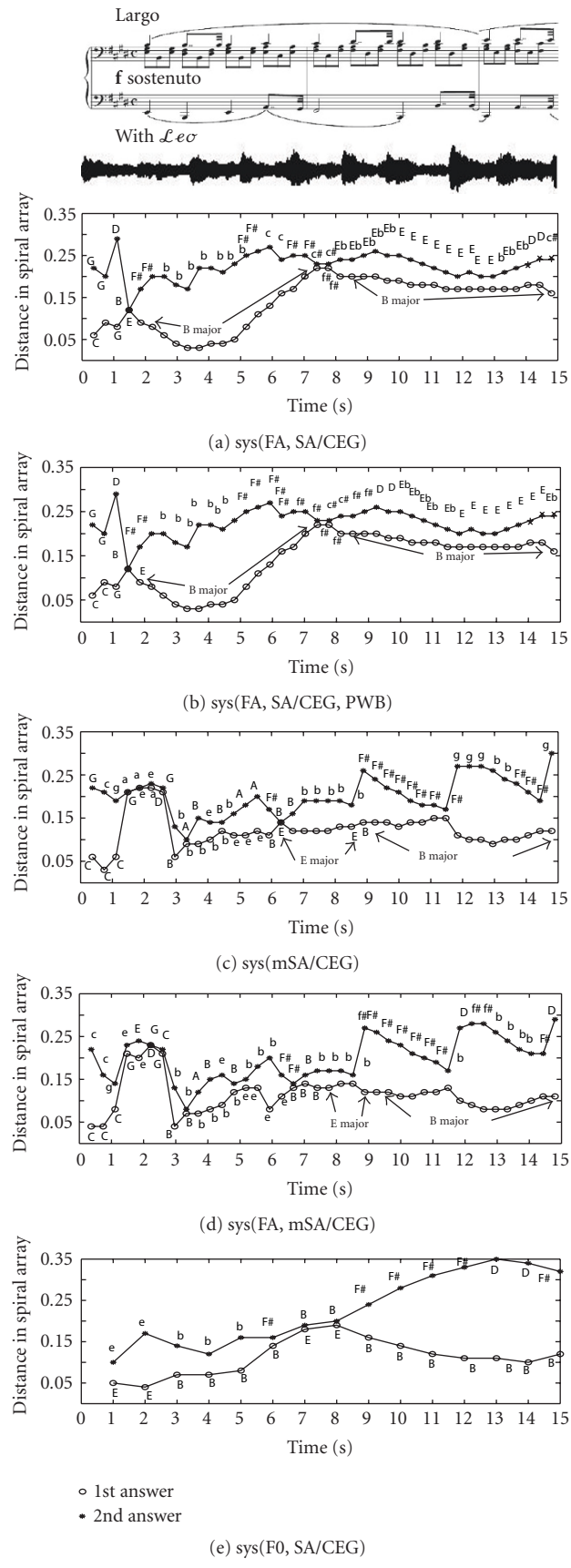


Figure 12: Key answers over time on Chopin's Prelude number 9 in E major.

as it is an important pitch that is repeated frequently in the melodic line. The emphasis on the pitch B_3, and later on the lowest note B_1, results in the larger weight of B in the pitch-class distribution. Even though the notes in the score and the chords as early as the first bar clearly show E major as the key, the prominence of the pitch B in the performance overshadows the answer so that B major and B minor appear most frequently as top choices for this piece, even though the key of E major is ranked first or second at least once in the first two seconds for all systems.

The results in Figures 12(a) and 12(b) show that post-weight balancing does not improve the answers in this case, where the pitches emphasized in the expressive performance dominate the results. The PWB method may be more effective if the parameters for the weight adjustments are optimized through learning. Compared with the other three systems, the F0 method employed in system (e) appears to have stabilized the results, and to react most sensitively to changing pitches in the lower registers. Sys(F0, SA/CEG) gets the correct answer within the first second, which is quicker than the others. In (c) and (d), sys (mSA/CEG) and sys(FA, mSA/CEG), the answers based on the modified spiral array model, appear more random than the other systems, and are difficult to explain directly from the score. Using the frequency distribution of the pitches instead of the apparent duration does not appear to have been very helpful in this case.

7. CONCLUSIONS AND DISCUSSION

We have presented a fundamental audio key-finding system, FACEG, with three key determination policies of (NN), (RD), and (AD). We evaluated the basic system by comparing the results between the audio key-finding systems, with the three different key determination policies, as well as a symbolic key-finding system. We showed that the fundamental audio key-finding system with the average distance key determination policy, FACEG (AD), is superior as it achieves generally 10% higher correct rates than the other systems. We showed that the stylistic period of the classical pieces could be used as an indicator for the difficulty of key finding for that test set.

We also presented three possible extensions to the basic system: the modified spiral array (mSA), fundamental frequency identification (F0), and post-weight balancing (PWB) scheme. We evaluated the three methods by constructing five audio key-finding systems with different combinations of possible extensions, and provided qualitative as well as quantitative analyses of their key-finding performance on recordings of Chopin's Preludes for piano. We observed that the fuzzy analysis system with the modified spiral array performs best, by the average correct rate metric. The overall performance of the five systems matched our expectations that identifying the fundamental frequencies is helpful in the first 8 seconds when fewer notes have been sounded; and the systems employing the modified spiral array, which incorporated audio frequency features in its weights, become the most effective after 8 seconds, when more notes have been played.

We further provided detailed analyses on the case when all five audio systems gave the correct answer and on another case when all five systems failed. The result of the case studies presented some evidence against the summary statistics as a metric for key finding, and demonstrated the specific advantages and disadvantages of the system extensions.

Each of the improvements proposed to the specific system addressed in this paper can also be employed in other audio key-finding systems. For example, modifying an existing model with instrument-specific signals (as in the modified spiral array method of Section 5.1) can be applied to any other key matching algorithm, such as the Krumhansl-Schmuckler method; fundamental frequency identification (Section 5.2) can be applied to the transcription of monophonic bass instruments, such as cello, with appropriate harmonic templates; post-weight balancing (Section 5.3) can be directly plugged into any audio key-finding system for refining pitch-class distributions based on the key answer.

In this paper, our test data comprised of audio synthesized from MIDI (mostly symphonies and concertos), and of audio from acoustic piano. Methods such as the modified spiral array (Section 5.1), and post-weight balancing (Section 5.3), and analyses of when audio outperforms MIDI key finding (Section 4.2) can be highly dependent on the instrumental timbre of the training and test data. Timbre is a variable that requires systematic study in the future.

The constant Q transform is another option for extracting frequency information from audio signal. The constant Q transform is closely related to the FFT, except that the frequency-resolution ratio remains constant. This constant ratio confers two major advantages to the constant Q: first, with a proper choice of center frequency, the output of the constant Q transform corresponds to musical notes; second, the constant Q transform uses higher time resolutions for higher frequencies, which better models the human auditory system. One of the design choices for the present audio key-finding system was to concentrate on lower frequencies, because bass notes present strong and stable cues for the key; thus, our decision was to use the proposed fuzzy analysis technique to confirm lower frequencies using the overtones. In the future, we plan to explore other techniques using the constant Q transform for audio key finding.

REFERENCES

[1] E. Chew, "Towards a mathematical model of tonality," Doctoral dissertation, Department of Operations Research, Massachusetts Institute of Technology, Cambridge, Mass, USA, 2000.

[2] E. Chew, "Modeling tonality: applications to music cognition," in *Proceedings of the 23rd Annual Meeting of the Cognitive Science Society (CogSci '01)*, pp. 206–211, Edinburgh, Scotland, UK, August 2001.

[3] C.-H. Chuan and E. Chew, "Fuzzy analysis in pitch-class determination for polyphonic audio key finding," in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR '05)*, pp. 296–303, London, UK, September 2005.

[4] H. C. Longuet-Higgins and M. J. Steedman, "On interpreting bach," in *Machine Intelligence*, vol. 6, pp. 221–241, Edinburgh University Press, Edinburgh, Scotland, UK, 1971.

[5] C. L. Krumhansl, "Quantifying tonal hierarchies and key distances," in *Cognitive Foundations of Musical Pitch*, chapter 2, pp. 16–49, Oxford University Press, New York, NY, USA, 1990.

[6] D. Temperley, "What's key for key? the Krumhansl-Schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[7] C.-H. Chuan and E. Chew, "Polyphonic audio key finding using the spiral array CEG algorithm," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME '05)*, pp. 21–24, Amsterdam, The Netherlands, July 2005.

[8] E. Gómez and P. Herrera, "Estimating the tonality of polyphonic audio files: cognitive versus machine learning modelling strategies," in *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR '04)*, pp. 92–95, Barcelona, Spain, October 2004.

[9] S. Pauws, "Musical key extraction from audio," in *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR '04)*, pp. 96–99, Barcelona, Spain, October 2004.

[10] 1st Annual Music Information Retrieval Evaluation eXchange, MIREX 2005, http://www.music-ir.org/index.php/Main_Page.

[11] C.-H. Chuan and E. Chew, "Audio key finding using FACEG: fuzzy analysis with the CEG algorithm," in *Abstract of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, London, UK, September 2005.

[12] E. Gómez, "Key estimation from polyphonic audio," in *Abstract of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, London, UK, September 2005.

[13] Ö. İzmirli, "An algorithm for audio key finding," in *Abstract of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, London, UK, September 2005.

[14] S. Pauws, "KEYEX: audio key extraction," in *Abstract of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, London, UK, September 2005.

[15] H. Purwins and B. Blankertz, "Key finding in audio," in *Abstract of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, London, UK, September 2005.

[16] Y. Zhu, "An audio key finding algorithm," in *Abstract of the 1st Annual Music Information Retrieval Evaluation eXchange (MIREX '05)*, London, UK, September 2005.

[17] E. Chew and A. R. J. François, "Interactive multi-scale visualizations of tonal evolution in MuSA.RT Opus 2," *Computers in Entertainment*, vol. 3, no. 4, pp. 1–16, 2005, special issue on Music Visualization.

[18] E. Chew and Y.-C. Chen, "Mapping MIDI to the spiral array: disambiguating pitch spellings," in *Proceedings of the 8th INFORMS Computing Society Conference (ICS '03)*, pp. 259–275, Chandler, Ariz, USA, January 2003.

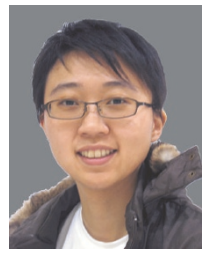
[19] E. Chew and Y.-C. Chen, "Real-time pitch spelling using the spiral array," *Computer Music Journal*, vol. 29, no. 2, pp. 61–76, 2005.

[20] Ö. İzmirli, "Template based key finding from audio," in *Proceedings of the International Computer Music Conference (ICMC '05)*, Barcelona, Spain, September 2005.

[21] Electronic Music Studios in the University of Iowa, http://theremin.music.uiowa.edu/MIS.html.

[22] A. P. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 804–816, 2003.

[23] A. Klapuri, "A perceptually motivated multiple-F0 estimation method," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, October 2005.



Ching-Hua Chuan received B.S. and M.S. degrees in electrical engineering from National Taiwan University in 1999 and 2001, respectively. She is currently pursuing a Ph.D. degree in computer science at Viterbi School of Engineering, University of Southern California. Since 2004, she has worked as a Graduate Research Assistant in the Music Computation and Cognition (MuCoaCo) Laboratory at the Integrated Media Systems Center (IMSC), a National Science Foundation Engineering Research Center. In 2005, she organized the audio key finding competition with Mardirossian and Chew in the First Annual Music Information Retrieval Evaluation Exchange (MIREX). Her research interests include audio signal processing, music content analysis, expressive performance study, and artificial intelligence. She has presented and/or published papers at national and international conferences such as the International Conference on Music Information Retrieval (ISMIR), International Conference on Multimedia and Expo (ICME), and the INFORMS Computing Society's biennial meeting. As a guitarist, she has performed in several all-female rock bands.

Elaine Chew is currently the Viterbi Early Career Chair Assistant Professor at the Epstein Department of Industrial and Systems Engineering in the University of Southern California, Viterbi School of Engineering, and the Founder and Director of the Music Computation and Cognition Laboratory at the Integrated Media Systems Center. She is a Recipient of the prestigious Presidential Early Career Award in Science and Engineering for her research on performer-centered approaches to computer-assisted music making, and her efforts to integrate research and education at the intersection of music and engineering, following a Career Award from the National Science Foundation. She received her Ph.D. and S.M. degrees in operations research from the Massachusetts Institute of Technology. She received her B.A.S. degree in music performance (distinction) and mathematical and computational sciences (honors) from Stanford University, and the Fellowship and Licentiate Diplomas from Trinity College, London, (FTCL and LTCL) in piano performance. Since her Ph.D. dissertation on mathematical modeling of tonality, she has published numerous refereed articles in journals and at conferences. She is on the founding editorial boards of ACM Computers in Entertainment, the Journal of Mathematics and Music, and the Journal of Music and Meaning, and on editors' panel for Computing in Musicology.