Real-Time Pitch Spelling Using the Spiral Array

Elaine Chew and Yun-Ching Chen

University of Southern California Viterbi School of Engineering
Epstein Department of Industrial and Systems Engineering
Integrated Media Systems Center
3715 McClintock Avenue GER 240 MC:0193, Los Angeles, CA 90089-0193, USA
echew@usc.edu, rs6@ycchen.net

This paper describes and presents a real-time bootstrapping algorithm for pitch spelling based on the Spiral Array model (Chew 2000). Pitch spelling is the process of assigning appropriate pitch names that are consistent with the key context to numeric representations of pitch, such as, MIDI or pitch class numbers. The Spiral Array model is a spatial model for representing pitch relations in the tonal system. It has been shown to be an effective tool for tracking evolving key contexts (see Chew 2001, 2002). Our pitch spelling method derives primarily from a two-part process consisting of the determining of context-defining windows and pitch name assignment using the Spiral Array. The method assigns the appropriate pitch names without having to first ascertain the key. The Spiral Array model clusters closely related pitches and summarizes note content by spatial points in the interior of the structure. These interior points, called *centers of effect* (CE's), approximate and track the key context for the purpose of pitch spelling. The appropriate letter name is assigned to each pitch through a nearest neighbor search in the Spiral Array space. The algorithms utilize windows of varying sizes for determining local and long-term tonal contexts using the Spiral Array model.

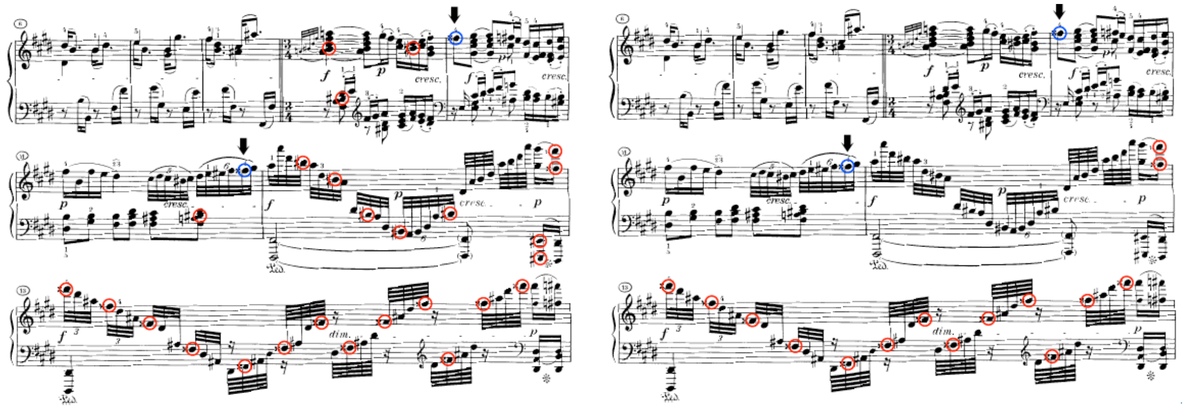
Why Spell?

The problem of pitch spelling is an artefact of equal temperament tuning in western tonal music whereby several pitches are approximated by the same frequency so as to ensure that music in different keys can be played using a reasonably sized pitch set. Pitches of the same frequency but denoted by different pitch names are said to be enharmonically equivalent. In MIDI format and many other numeric music representations, enharmonically equivalent pitches are represented by the same number, indicating the note’s ostensible frequency and not its letter name. Hence, each number can map to more than one letter name. For example, MIDI number 56 can be spelt as A♭ or G#. The spelling of a given pitch number is primarily dependent on the key context, and to a lesser extent, the voice leading tendencies in the music.

One might argue that enharmonic spellings are part of a man-made system that may not entirely be necessary for the representation and analysis of music. However, as noted by Cambouropoulos (2003), "[enharmonic spelling] *allows higher level musical knowledge to be represented and manipulated in a more precise and parsimonious manner*" and thus "*may facilitate other musical tasks such as harmonic analysis, melodic pattern matching, and motivic analysis.*" Meredith (2004) adds that “*a correctly notated score … is a structural description of the passage that is supposed to represent certain aspects of the way that the passage is intended to be interpreted by an expert listener... [and] serves a similar function to, say, a Schenkerian analysis.”* We proceed to make a case for the necessity for better spelling algorithms.

Pitch spelling is the prelude to all automated transcription systems. The spelling of each pitch directly affects its notation. For example, the choice of A♭ or G# for MIDI number 56 affects its notation in a score. There is much room for improvement in spelling technology as demonstrated by the performance of leading notation software in transcribing MIDI to score. We show in Figure 1 examples of pitch spelling errors in an excerpt from the beginning of Beethoven's Piano Sonata Op.109. The notation shown is the correct spelling, and the circles indicate places in the music where a misspelling (not shown) occurred. The spelling errors made by two leading notation software are circled in the two examples. In each case, the default settings were used. The process typically involved the reading of the key from the file followed by pitch names assignments based on the given key. The majority of errors in these two cases were due to an inability to detect the changes in the local tonal context. On the other hand, the arrows indicate the spelling errors made by our algorithm.

Figure 1. Spelling errors by two leading notation software programs (locations circled, misspellings not shown) compared to those made by our proposed algorithm (arrowed).



FILE: Compare_109_Finale_2.eps FILE: Compare_109_Sibelius-2.eps

Pitch spelling is also critical for building accurate computer systems for music analysis. Accurate spelling will result in more robust key finding algorithms. For example, G# is an important pitch (the leading tone) in the key of A major, whereas A♭ is not a member of the same key and would hardly be used at all except in passing. Pitch spellings also affect chord recognition algorithms and the interpretation of chord function. Consider the problem of determining the chord label and function of the MIDI numbers { 60, 64, 68 }. The pitches { C, E, G# } form an augmented triad with C as its root while the pitches { A♭, C, E } form an augmented triad with A♭ as its root with very different implications for the ensuing harmonic motion.

The local key context determines the pitch spelling; in turn, the name of the pitch determines the notation and serves as a clue to the key context. When creating and notating a new piece, the composer's choice of musical notation, such as pitch spelling, stems from his or her concept of the local context (key) and voice leading. The notated score then serves as a guide for the expert listener or interpreter to the composer's intended musical structure (contextual delineations) for the piece. This inter-dependence between notation (such as pitch spelling) and structure (for example, context) creates a potentially problematic cycle in the design of an accurate algorithm for pitch spelling, the classic 'chicken and egg' problem.

We de-couple the problems of pitch spelling and key-finding by creating an algorithm that does not require explicit knowledge of the key context to determine the appropriate spelling. Using the Spiral Array, any musical fragment that has been correctly spelled will generate a center of effect (CE), a summary point in the model's interior, that is closest to, and hence can act as a proxy for, the key. Using this evolving CE, the algorithm maps each numeric representation to its plausible pitch names on the Spiral Array and selects the best match through a nearest-neighbor search. Our algorithm computes in real-time, requiring only present and past information, thus facilitating its integration into real-time music systems.

In this paper, we test the algorithms using MIDI rather than audio data to isolate the pitch spelling process for more accurate assessment. A system that takes audio as input, even audio generated from MIDI data, would introduce additional error through the audio to fundamental frequencies transcription. By starting with MIDI information, we directly assess the accuracy of the proposed pitch spelling algorithms. Note that the algorithms apply to any numeric input that is related to frequency and not just MIDI data. Examples of such representations include, and are not limited to, audio input that has been transcribed to frequency numbers, pitch class sets and piano rolls.

We present and analyze the computational results of three variants of our real-time bootstrapping pitch spelling algorithm when applied to examples from two different cultures. The first set consists of selected movements from Beethoven's Piano Sonatas: the third movement of Op.79 and the first movement of Op.109. The second is a set of variations for piano and violin on a popular Taiwanese folksong by composer Youdi Huang with a primarily pentatonic melody. The late Beethoven (Op.109) Sonata is a complex example, a mature work by the composer containing many key changes, both sudden and subtle. By careful analysis of the pitch spelling results using a piece of such complexity, we gain insight into the challenges in pitch spelling so as to guide the algorithm design and parameter selection. We found few examples of music of harmonic complexity comparable to the late Beethoven sonata with pitch spelling information in the public domain. In our tests, all musical examples were optically scanned and manually checked for errors, then converted to MIDI from score as input for the program. It is important to point out that it is not the aim of this current paper to provide a comprehensive or comparative evaluation of the algorithm; such a study will be an important step in future research.

Related Work

Pitch spelling has been a concern in computer music research since the early work of Longuet-Higgins (1976) where the goal of automated transcription was achieved by a combination of meter induction, pitch spelling and key induction. Although in his earlier work with Steedman (1971), Longuet-Higgins touches on the problem of pitch spelling, that work focussed on key-finding, and the authors noted near the end of the paper that *"after the key has been established then accidentals can be met with equanimity."* In Longuet-Higgins (1976), appropriate pitch spellings were selected such that the distance between each note and the tonic of the given key was less than six steps in the line of fifths, thus favoring diatonic intervals over chromatic ones. Longuet-Higgins defined diatonic intervals to be the distance between pitches separated by less than six steps on the line of fifths (P5/P4, M2/m7, M6/m3, M3/m6, M7/m2, where P = perfect, M = major, m = minor, d = diminished and A = augmented); the `diabolic' tritone interval has a separation of exactly six steps on the line of fifths; and, any interval separated by more than six steps is considered a chromatic one. The tonic is derived by assuming that the first note of the melody is either the tonic or dominant. As the melody unfolds, the algorithm re-visits the tonic choice by evaluating a small local set of notes to see if an alternative interpretation of the local key would result in diatonic intervals among the pitches. The program was applied to the English horn solo from the prelude to Act III of Wagner's *Tristan und Isolde*.

More recent pitch spelling algorithms include Temperley's line-of-fifths approach (2001), Cambouropoulos' interval optimization approach (2001, 2003) and Meredith's ps13 algorithm (2003, 2004). These pitch-spelling algorithms de-couple the problem of key finding and pitch spelling; they produce output that can be used as input for key finding programs. Admittedly, they all incorporate implicit knowledge of key structure in their design, for example, through the representations used, their interval preferences and/or the use of harmonic analysis.

Temperly's algorithm (2001), like Longuet-Higgins', uses the line of fifths pitch representation. Temperley's method maps pitches to the line of fifths in a way such that the spellings map to close clusters on the line. Each collection of spelt notes generates a center of gravity on the line of fifths. The best spelling is determined by seeking the nearest neighbor to the current center. In addition, the algorithm avoids chromatic semitones adjacent to each other through voice leading rules and prefers good harmonic representations through the harmonic feedback rule. The accuracy of the pitch-spelling algorithm depends on the reliability of the metrical and harmonic analysis. The method was implemented in the Melisma Harmonic

Analyzer (Sleator and Temperley 2001) and tested on the Kostka-Payne corpus. Temperley reports that the algorithm made 103 spelling errors out of a total of 8747 notes, giving a correct percentage of 98.82% (Temperley 2001, p.136).

Cambouropoulos' method (2001, 2003) focuses on interval relations, and selects pitch spellings based on intervals more likely to occur within the tonal scales. The fundamental principles of this algorithm are: (1) notational parsimony (minimizing the use of accidentals); and, (2) interval optimization (avoiding diminished and augmented intervals that are rare or do not appear at all in the tonal major and minor scales). By limiting the interval choice, Cambouropoulos essentially constrains the pitch-spelling search to a local region along the line of fifths. Cambouropoulos argues that the line-of-fifths approach can be viewed as a special case of the interval optimization algorithm. The algorithm was tested on ten complete sonatas by Mozart (K279-284, K330-K333) and on three waltzes by Chopin (Op.64 Nos.1-3). Because Chopin's harmonic language was more complex than Mozart's, the algorithm performed better on the Mozart examples. Cambouropoulos reports an overall correct spelling rate of 98.59%.

Meredith's ps13 algorithm can be broken down into two stages. The first stage votes for the most likely spelling while the second stage scans and corrects for local voice-leading errors. In stage 1, the algorithm counts the number of times each pitch occurs in a context surrounding the note to be spelt; then, these numbers, summed over all plausible harmonic chromatic scale tonics that would result in a given letter name, returns the votes for the given note having that letter name. The letter name with the highest number of votes wins.

Table 1. Percentage of intervals and notes spelt correctly for selected composers based on Meredith's 2004 comparison of pitch spelling algorithms.

Author(s)	Method	Percentage correct			
		Vivaldi *1678-1741*	Bach *1685-1750*	Mozart *1756-1791*	Beethoven *1770-1827*
		223678 notes	627083 notes	172097 notes	48659 notes

Meredith	ps13	99.23	99.37	98.49	98.54
Cambouropoulos	Interval Opt	99.13	97.92	98.58	98.63
Longuet-Higgins	Line-of-Fifths	98.48	98.49	96.26	97.46
Temperley	Line-of-Fifths	98.69	99.74	93.40	92.34
AVERAGE		**98.88**	**98.88**	**96.68**	**96.74**

Meredith compares the ps13 algorithm's performance with that of his implementations of Longuet-Higgins', Temperley's and Cambouropoulos' algorithms. In the 2003 pilot study, he tested the algorithms on all Preludes and Fugues in Book 1 of J. S. Bach's *Well-Tempered Clavier* (BMV 846 - 869) and reported the following correct percentage rates: ps13 (99.81%), Cambouropoulos (93.74%), Longuet-Higgins (99.36%) and Temperley (99.71%). In an updated study using a large corpus of tonal music containing a total of 1.73 million notes, Meredith reports the following results: ps13 (99.33%), Cambouropoulos (98.71%), Longuet-Higgins (97.65%) and Temperley (97.67%). The corpus consists of music by nine composers ranging from Corelli to Beethoven. The span of musical styles is limited: 80% of the notes are contributed by Baroque music, and there are only ten movements (less than 3% of the corpus, note-wise) by Beethoven, the most recent composer. Some of the results are shown in Table 1. In these excerpted evaluations, the Vivaldi-Bach selections generally scored much higher than the Mozart-Beethoven examples, with the Beethoven selection being the one garnering the most errors in three out of the four methods tested.

Table 2. Result summary for recent pitch spelling algorithms as reported by the authors.

Author(s)	Algorithm	Test Set	No. of notes	% correct
Chew & Chen (present study)	Spiral Array	Beethoven *Sonata Op.79* (mvt 3)	1375	99.93
		Beethoven *Sonata Op.109* (mvt 1)	1516	98.21
		Huang *Song of Ali-Shan*	1571	100.00
		OVERALL	**4462**	**99.37**
Cambouropoulos	Interval Optimization	Mozart *Pno Son.* (K279-284, K330-333)	54418	98.80
		Chopin *Waltzes* (Op.64 Nos.1-3)	4876	95.80
		OVERALL	**59294**	**98.59**
Meredith	ps13	Bach *WTC Book 1* (BWV 846-869)	41544	99.81
		Corelli through Beethoven	**1.73M**	**99.33**
Temperley	Line-of-Fifths	Kostka-Payne corpus	**8747**	**98.82**

Table 2 summarizes the results from the different approaches to pitch spelling as reported by the respective authors, including the one presented in this paper. It is important to stress that, given the varied test data, it is difficult to compare the algorithms. As can be seen from Meredith's 2003 and 2004 studies, and later from our reports on two Beethoven pieces, the results of the pitch spelling algorithms depend highly on the test corpus.

It is not the purpose of this paper to draw numerical comparisons among the various pitch spelling algorithms. The goal of this paper is to present a computational approach to pitch spelling utilizing the Spiral Array and its development through computational testing on a challenging work that reveals some core requirements for robust pitch spelling. We highlight the similarities and differences between our Spiral Array approach and the various algorithms of other authors.

The Algorithm

We present a real-time computational algorithm for pitch spelling using the Spiral Array model first proposed by Chew in 2000. Any approach to pitch spelling must address the problem of determining the local context, for example, by determining a center of gravity on the line of fifths (see Temperley 2001), by providing a hypothesis on the current tonic (see Longuet-Higgins 1976), by calculating the likelihood of a given context (see Meredith 2003, 2004) or by interval optimization over a sliding window (see Cambouropoulos 2003). Our approach consists of two phases, not necessarily in chronological order, namely: (1) a method for assessing the tonal context of a section and assigning the appropriate pitch names; and, (2) strategies for selecting the window of events that define the tonal context.

Our method for assigning appropriate pitch names uses the Spiral Array model. The Spiral Array is a geometric representation for tonal entities that spatially clusters pitches in the same key and closely related keys. The Spiral Array model uses aggregate points inside the structure called *centers of effect* (CEs) to summarize tonal contexts. The CE approach is similar to Temperley's line-of-fifths approach in that a center of gravity is generated by the data. The depth added by going from one to three dimensions allows the modeling of more complex hierarchical relations in the Spiral Array. Given a CE that represents the local context, each pitch name is assigned through a nearest neighbor search in the Spiral Array space.

The Spiral Array model consists of spatial representation for pitch classes, chords and keys. Higher-level entities are generated as weighted sums of their lower level components. Each type of tonal entity forms a spiral, resulting in an array of nested spirals. The same idea of spatial aggregation extends to the concept of the center of effect (CE). Any segment of music can be mapped to the Spiral Array structure and can generate a CE, the sum of the pitch class representations weighted by their durations. This CE has been shown to be an effective tool for determining the key of the section through a nearest neighbor search for the closest key representation (Chew, 2001). For this reason, we use the CE as a proxy for the tonal context in the pitch spelling algorithm.

Tracking an evolving tonal context is a difficult problem. Since the Spiral Array space clusters closely related tonal entities, we do not need to track precisely the occurrence of shifts to nearby keys for the purpose of pitch spelling. It suffices to be able to catch abrupt changes to more distant tonalities in the context of overall key of the piece. We achieve this goal by using a combination of long-term (a cumulative window) and short-term (sliding windows) information to determine the local context.

The Spiral Array Model

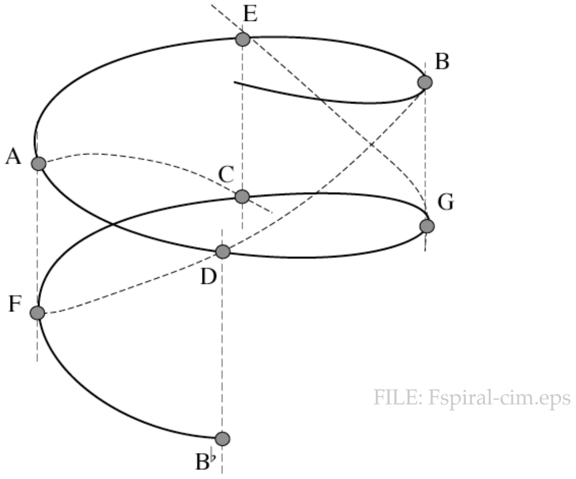
The main difference between our approach and other recent algorithms is the use of the Spiral Array representation for tonal entities. We briefly describe the part of the

Spiral Array model relevant to the pitch spelling algorithm, namely, the pitch class spiral. In the Spiral Array, pitches map to spatial positions at each quarter turn of an ascending spiral, and neighboring pitches are four scale steps (that is to say, the distance of a Perfect fifth or seven semitones) apart, which results in vertically aligned pitches being two major scale steps (or four semitones) apart:

$$\mathbf{P}(\mathrm{k}) = \begin{bmatrix} x_{\mathrm{k}} \\ y_{\mathrm{k}} \\ z_{\mathrm{k}} \end{bmatrix} = \begin{bmatrix} \mathrm{r}\sin(\mathrm{k}\pi/2) \\ \mathrm{r}\cos(\mathrm{k}\pi/2) \\ \mathrm{kh} \end{bmatrix},$$

where r is the radius of the spiral and h is the vertical ascent per quarter turn. Each pitch is indexed by its distance, according to the number of Perfect fifths, from an arbitrarily chosen reference pitch, C. Note that the pitch class representations is a spiral configuration of the line of fifths, and of the *tonnetz* attributed to the mathematician Euler (see Cohn 1998) or the Harmonic Network used by Longuet-Higgins (1962a, 1962b). The pitch class representations in the Spiral Array are shown in Figure 2. Higher level objects, such as triads and keys are generated as convex combinations of their defining pitches and triads respectively (Chew 2000).

Figure 2. A section of the pitch-class spiral in the Spiral Array.



Some interval preference is implicitly incorporated into the geometric model. As part of its design, the Spiral Array is calibrated so that spatial proximity corresponds to perceived relations among the represented entities. The calibration criteria that pertain directly to the pitch spiral correspond to perceived closeness between pairs of pitches. The aspect ratio,r/h, is set at $\sqrt{2/15}$, in which case, intervals in a major triad (P5/P4, M3/m6) map to the shortest inter-pitch distances, other intervals in the diatonic major scale (M6/m3, M2/m7, M7/m2) map to farther distances, and the tritone (d5/A4) maps to even farther distances.

In the Spiral Array model, any collection of notes generates a *center of effect* (CE). For a sequence of music time series data, the CE is a point in the interior of the Spiral Array that is the convex combination of the pitch positions weighted by their respective durations. Formally, if

n_j = number of active pitches in chunk j,

$\mathbf{p}_{i,j}$ = the Spiral Array position of the i-th pitch in chunk j, and

$\mathrm{d}_{i,j}$ = the duration of the i-th pitch in chunk j,

we divide musical information in equal time slices which we call chunks and define $\mathbf{c}_{a,b}$ to be the CE of all pitch events in the window from chunks a through b:

$$\mathbf{c}_{a,b} = \frac{1}{D_{a,b}} \sum_{j=a}^{b} \sum_{i=1}^{n_j} d_{i,j}\, \mathbf{p}_{i,j}\,, \quad \text{where} \quad D_{a,b} = \sum_{j=a}^{b} \sum_{i=1}^{n_j} d_{i,j}\,.$$

The CE method has been shown to be an effective way to determine the tonal context of a section through a nearest neighbor search for the closest key representation in the Spiral Array model (Chew 2001). In Chew (2001), the CE method using the Spiral Array was shown to determine the key in fewer steps than Longuet-Higgins and Steedman's (1971) shape-matching algorithm utilizing the Harmonic Network or Krumhansl & Schmuckler's (see Krumhansl 1990) probe tone profile method when tested on the canonical *Well-Tempered Clavier* Book 1 fugue subjects. Consequently, we use the CE as a proxy for the key context without explicitly having to determine the actual key.

In the Spiral Array, pitches that are in the same key form compact clusters in the 3D space. The fact that the Spiral Array serves as an effective model for key-finding and clusters pitches in a given key leads to the hypothesis that it would perform well in tracking the key by proxy and assigning pitches by a nearest neighbor search.

Pitch Name Assignment

Using the CE method for generating a center of effect inside the Spiral Array, we use the Spiral Array model to determine spellings for unassigned pitches. Pitches in a given key occupy a compact space in the Spiral Array model. Thus, the problem of finding the best pitch spelling corresponds to finding the pitch representation that is nearest to the current key context. Each plausible spelling of an unassigned pitch is measured against the current CE, and the pitch that satisfies the nearest neighbor criteria is selected to be the appropriate pitch name. See Figure 3 for a graphical depiction of the pitch name assignment process.

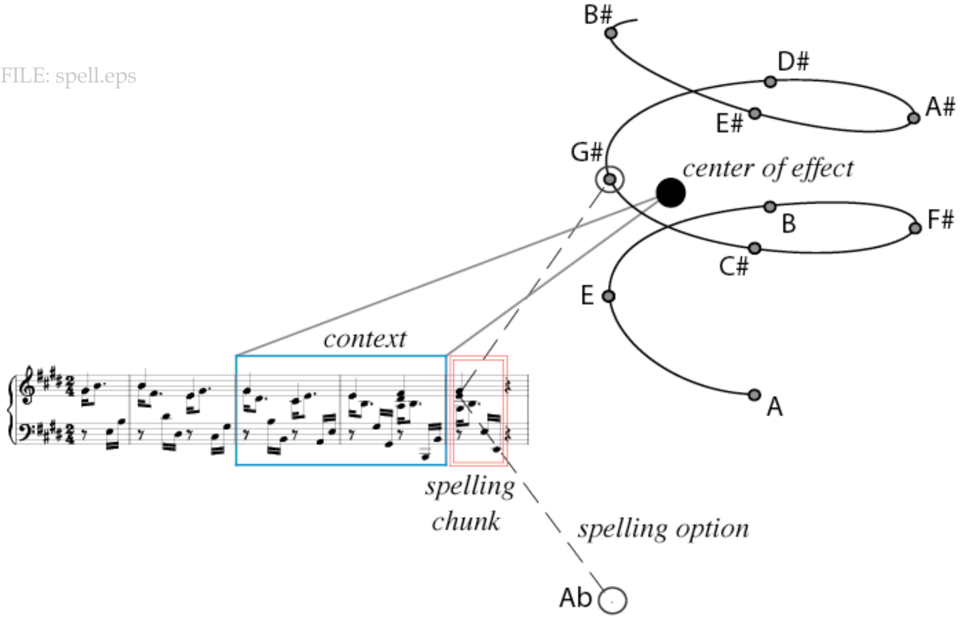
More concretely, each pitch read from the MIDI file will correspond to two or three most probable letter names as shown in Table 3 (the numbers in the brackets indicate the index of the pitch in the Spiral Array):

Table 3. Table of equivalent spellings.

Option 1	*(SA index)*	*Option 2*	*(SA index)*	*Option 3*	*(SA index)*
B#	(12)	C	(0)	D♭♭	(–12)
C#	(7)	D♭	(–5)	B##	(19)
C##	(14)	D	(2)	E♭♭	(–10)
D#	(9)	E♭	(–3)	F♭♭	(–15)
D##	(16)	E	(4)	F♭	(–8)
E#	(11)	F	(–1)	G♭♭	(–13)
E##	(18)	F#	(6)	G♭	(–6)
F##	(13)	G	(1)	A♭♭	(–11)
G#	(8)	A♭	(–4)		
G##	(15)	A	(3)	B♭♭	(–9)
A#	(10)	B♭	(–2)	C♭♭	(–14)
A##	(17)	B	(5)	C♭	(–7)

Note that the index triplets are of the form <*index*–12, *index*, *index*+12>. These indices correspond to pitch positions separated vertically by three cycles of the spiral. The pitch spiral extends to indices of larger magnitude than shown on the table, for example, index –16 or 20 for the (G#,A♭) option. In Table 3, we have shown only the most parsimonious (and common) spellings, that is, spellings up to and including two sharps or two flats.

Figure 3. Pitch name assignment using the Spiral Array.



Having determined the candidates for spelling, the algorithm then selects the spelling that is closest to the current CE. Suppose that we are currently assigning names to the pitches in chunk *j*, and the probable pitch names of the i-th pitch in this chunk are < $I_{i,j}$ – 12, $I_{i,j}$, $I_{i,j}$ + 12 >. Assume that $\mathbf{c}_j$ represents the CE that defines the context of chunk *j*. The index that is consistent with the key context is given by:

$$I^*_{i,j}\left(\mathbf{c}_j\right) = \arg\min\left\{\left\|\mathbf{P}\left(I_{i,j} - 12\right) - \mathbf{c}_j\right\|, \left\|\mathbf{P}\left(I_{i,j}\right) - \mathbf{c}_j\right\|, \left\|\mathbf{P}\left(I_{i,j} + 12\right) - \mathbf{c}_j\right\|\right\}$$

(The standard mathematical construction "arg min" means the argument that minimizes the function. In this case, "arg min" returns the index that minimizes the function from the choices $I_{i,j}$ – 12, $I_{i,j}$, $I_{i,j}$ + 12. The optimal choice is called $I^*_{i,j}$.)

The concept is illustrated graphically in Figure 3. The single-lined box denotes the contextual window. The notes in this window generate a CE inside the Spiral Array. The double-lined box denotes the next chunk to be spelt. The first note in this spelling chunk can be spelt as G# or A♭ (note that this is the only case where the number of options is two, see Table 2). The spelling option closest to the CE is selected, in this case, G#.

Comparison to Other Pitch Spelling Models

Note that the current pitch name assignment mechanism bears some similarities to that of previous models. Like Temperley's center of gravity approach, the center of effect (CE) serves as a kind of center of gravity in the Spiral Array space. The difference lies in the representation (the Spiral Array versus the line of fifths) and the segmentation scheme present in our approach. At this point, it would be useful to compare the interval distance rankings in the different models, including Cambouropoulos' interval optimization model, as shown in Table 4. Distinct from other models, the Spiral Array prioritizes the Major third relation above the seconds and sevenths, and ranks the tritone (A4/d5) much lower than the other models.

Table 4. Spiral Array Pitch-to-Pitch Distances and Interval Preference Comparison.

Order of Preference	Closest (most preferred) to farthest (least preferred)											
Spiral Array	P4 P5	M3 m6	m3 M6	M2 m7	m2 M7	d4 A5	d1 A8	A1 d8	A4 d5	A2 d7	d3 A6	d6 A3
Line of Fifths	P4 P5	M2 m7	m3 M6	M3 m6	m2 M7	A4 d5	A1 d1	d4 A5	A2 d7	D3 A6	A3 d6	d2 A7
Interval Optimization	P4 P5	m2 M7	M2 m7	m3 M6	M3 m6	A2 d7	d3 A6	d4 A5	A4 d5	d1 A1	A3 d6	d2 A7

In Longuet-Higgins' approach, all unspelled pitches are compared to a key choice. Although we do not explicitly determine the key context, the CE acts as a proxy for the key, and key-to-pitch distances also exist in the Spiral Array. The key representation resides on an interior spiral and serves as a prototypical CE in a given context. Table 5 shows the key-to-pitch distance rankings for the Spiral Array model and the line of fifths approach used by Longuet-Higgins. In Longuet-

Higgin's method, the key is mapped to the same line of fifths as the pitches, while in the Spiral Array, the added dimensions allow the key representation to be defined in the interior of the structure, thus resulting in asymmetry between in the interval relations: for example key C to pitch G (a P5 with C as lower pitch) is not the same as key C to pitch F (a P5 with F as lower pitch or a P4 with C as lower pitch). For the Spiral Array, the intervals are given with the key as the lower pitch. Note that a P4 is ranked lower than P5, M3, M2 and M6, and A1 (for example, C# in the key of C) is ranked higher than m2 (D♭ in the key of C).

Table 5. Pitch-to-Key Distances in Spiral Array versus Line of Fifths.

Order of Preference	Closest (most preferred) to farthest (least preferred)											
Spiral Array	P5	M3	M2	M6	P4	M7	m3	m7	m6	A4	A1	m2
Line of Fifths	P4 P5	M2 m7	m3 M6	M3 m6	m2 M7	A4 d5	A1 d1	d4 A5	A2 d7	D3 A6	A3 d6	d2 A7

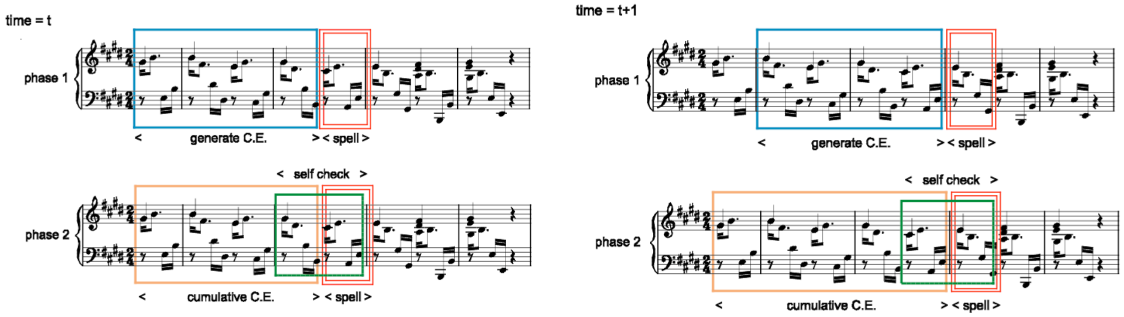
Apart from the pitch representation, an important distinction between our method and that of others (with the possible exception of Longuet-Higgins' method), is that our method processes pitch names in real time using only present and past information as will be evident in the upcoming section. This feature allows the algorithm to be integrated easily into real-time systems for music processing.

Segmentation Strategy

Another important aspect of our algorithm is the determining of context-defining windows for generating the contextual CE. The context of any segment of music is the result of the interaction between the local changes and the larger-scale key of the piece. We propose a two-phase boot-strapping approach to capture the effects of both the local changes as well as the global context in real-time.

We segment the MIDI data into chunks for analysis, so as to batch process the pitch name assignments. Each iteration of the algorithm consists of two phases, and two iterations of the algorithm are shown in Figure 4. In the first phase, we assign the pitch names to the current chunk (indicated by the double-lined box) using the last w_s chunks as the contextual window for generating a CE. In the second phase, the context is given by a convex combination of a smaller local window (the smaller box of size w_r that overlaps both the current chunk and the cumulative window) and the cumulative window. The current chunk is then re-spelt using the hybrid CE.

Figure 4. Two-phase assignment method ($w_s = 4$, $w_r = 2$).



FILE: Algorithm.eps

Formally, if chunk j contains the pitches to be assigned letter names, the assignments are made using the following CE's:

Phase I: pitch names are assigned using the CE: $\mathbf{c}_j = \mathbf{c}_{j\square w_s, j\square 1}$.

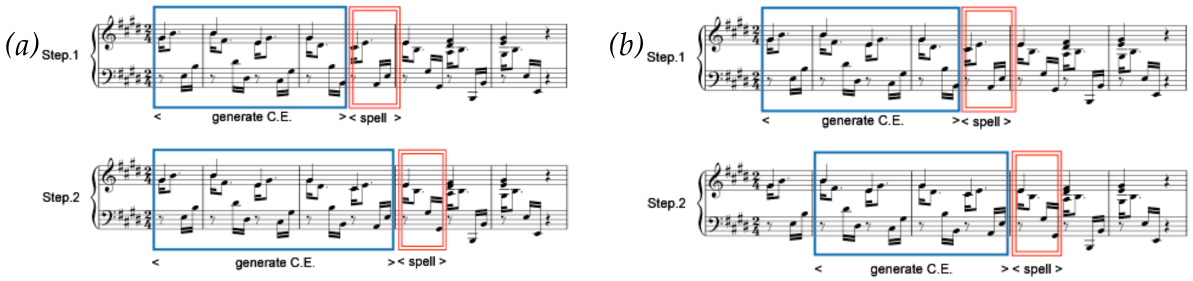
Phase II: pitch names are assigned using a hybrid CE:
$\mathbf{c}_j = f \cdot \mathbf{c}_{j\square w_r+1, j} + \left(1\square f\right) \cdot \mathbf{c}_{1, j\square 1}$, where $0 \square f \square 1$.

Qualitatively, f determines the balance between the local and global contexts. When $f = 1$, the current chunk is re-spelt using only the local context; when $f = 0$, the chunk is re-spelt using only the global context. To initialize the process, the notes in the first chunk are assigned pitch names with indices closest to 2, thus biasing the notation towards fewer sharps and flats. A CE is generated based on this preliminary assignment, and the original assignments are revisited to make them consistent with the chosen key context. The second pass on the spelling ensures that the first batch of name assignments are self-consistent.

Note that this bootstrapping algorithm also serves as a general framework for other techniques for capturing tonal contexts, such as, the cumulative window approach (reported in the authors' earlier paper) and the sliding window approach.

The cumulative window approach uses only the global context to assign pitch spellings: this is achieved by setting $w_s = 0$ and $f = 0$ (see Figure 5(a)). The sliding window approach uses only the recent window of events to assign pitch spellings: this is achieved by setting w_s to the desired window size and by setting $w_r = 0$ and $f =$ 1 (see Figure 5(b)). When $b < a$ in $\mathbf{c}_{a,b}$, a CE is not generated and the algorithm defaults to a do-nothing step.

Figure 5. Special cases of the bootstrapping algorithm: (a) cumulative CE method: w_s=0 and f=0. (b) sliding window method: w_s=4, w_r=0 and f=1.



FILE: Algorithm_1.eps

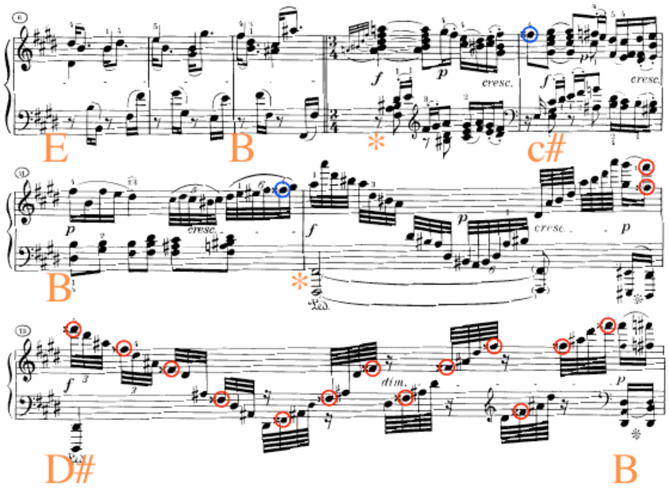
FILE: Algorithm_2.eps

Computational Results

The pitch-spelling algorithm was implemented in Java as part of the MuSA (music on the spiral array) package, an implementation of the Spiral Array and its associated algorithms. The algorithm assigns letter names to MIDI numbers in real-time as the music unfolds. We compare these algorithm-assigned pitch names with those chosen by the composer as notated in the score to evaluate the algorithm's efficacy. We created our data sets by manually scanning the scores using the Sibelius music notation program, checking the digital scores visually and aurally for optical recognition errors, and generating spelling solution tables from the score by hand to ensure accuracy. The correct spelling solutions are typed into an Excel spreadsheet for comparison with the algorithm's output. The program requires only MIDI input. For all the experiments, we used MIDI files generated from the Sibelius scores and set the chunk size to one beat. To separate the problems of beat tracking and pitch spelling, the beat size was read from the MIDI files. Note that the algorithms generalize to real-time spelling of MIDI data captured from live performance, in which case, the chunk size could be some unit of time.

The real-time bootstrapping algorithm was tested on two movements from Beethoven's (1770-1827) *Piano Sonatas* and on You-Di Huang's (b.1912) *Song of Ali-Shan*, a set of variations for voice and piano. Beethoven's thirty-two piano sonatas are a staple in the piano literature; the late sonatas, in particular, are considered masterworks that bridge the classical and romantic eras. The two movements that we have chosen are the third movement of *Sonata No. 25 in G major*, Op.79 (composed 1809) and the first movement of *Sonata No.30 in E major*, Op.109 (composed 1820). The late sonata, Op.109 is a particularly challenging example because it is harmonically complex and contains numerous subtle and sudden key changes. This is the data set that inspired the bootstrapping algorithm for pitch spelling. Figure 6 shows the local keys present in each part of the selection shown in Figure 1, one of the explanations for the challenges posed by this test set.

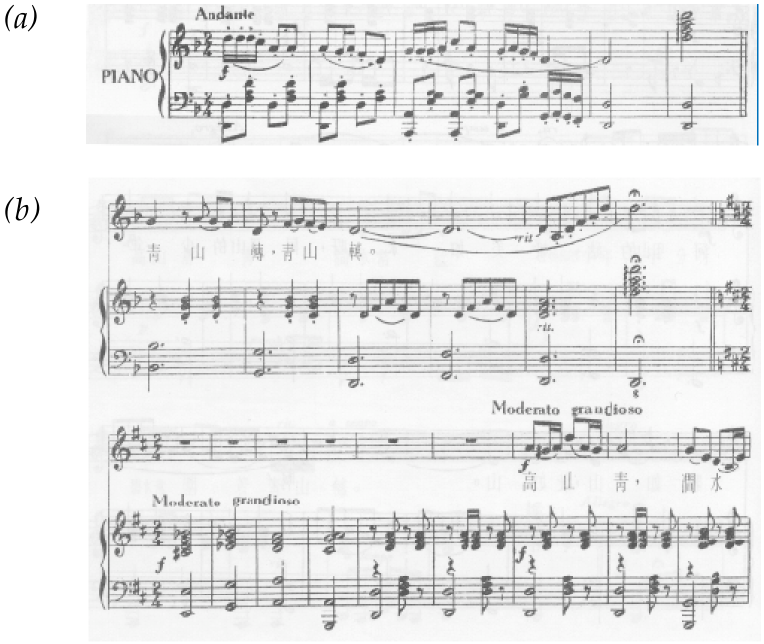
Figure 6. Spelling challenges: Intermediate keys.



FILE: Beeth109_whyhard.eps

You-Di Huang is a celebrated Chinese-born composer and music educator who currently resides in Taiwan. The *Song of Ali-Shan* (composed in 1967), A028 in d minor, composed for violin/chorus and piano, are based on a popular Taiwanese song extolling the beauty of the maidens of Ali-Shan (Mount Ali). This data set presents some different challenges for the bootstrapping algorithm. The theme (shown in Figure 7(a)), and much of the piece, is pentatonic with a distinct Chinese flavor. The first variation is pentatonic in style and in duple (2/4) meter. The second variation switches to a triple meter. The third variation returns to a duple meter but transitions to a seven-pitch scale (see Figure 7(b)), using chromatic pitches to elaborate on the original pentatonic melody. The meter change would test the robustness of the choice of window size under different pitch-time organizations. The first half of the piece is predominantly in a d minor mode, while the second half proceeds in the key of D major. Even though d minor and D major share the same tonic, they differ by two important pitches (F/F# and B♭/B).

Figure 7. Spelling challenges in the Song of Ali-Shan: (a) theme for "The Song of Ali-Shan"; (b) a transition to the parallel major key with change from triple to duple meter.



FILE: Ali_Bar0.eps

FILE: Ali_Bar078eps

Table 6. Table of results for cumulative window experiments.

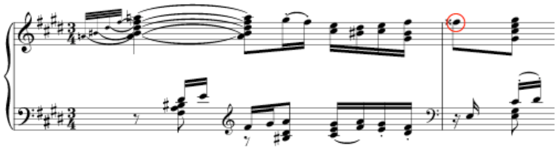
Piece	No. of notes	No. of errors	% correct
Beethoven Op.79 (3rd mvt)	1375	1	99.93
Beethoven Op.109 (1st mvt)	1516	73	95.18
TOTAL	2891	74	97.44

Brief Review of Prior Experiments

We briefly summarize the results of an earlier pitch spelling experiment using the cumulative window algorithm, leading to the motivation for the bootstrapping approach. Note that a cumulative window approach is similar in spirit to Longuet-Higgins and Steedman's (1971) algorithm in that a a global context is used to assign pitch names. In the Beethoven Op.79 example, the cumulative window algorithm achieved a correct spelling rate of 99.93% (one error out of 1375 notes; overlapping and tied notes were counted only once). In the Beethoven Op.109 example, the same algorithm achieved a correct spelling rate of 95.18% (73 errors out of 1516 notes). The overall correct rate was 97.44% as summarized in Table 6.

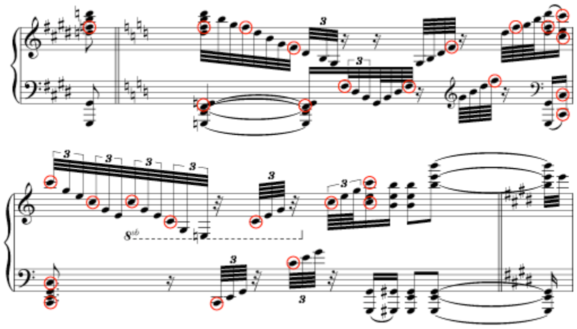
Figure 8. Sources of error in the cumulative window approach (notation shown is correct spelling, and circles indicate locations of misspellings, not shown):

(a) linear motion in Op. 109 resulting in spelling error in bar 10 (circled);



FILE: B109_Linear.eps

(b) sudden key change prior to bar 61 in Op. 109 resulting in spelling errors (circled).



FILE: B109_SuddenKey.eps

The errors in both examples can be categorized into two types: that based on voice leading conventions, and that due to unexpected local key changes. Figure 8(a) shows an example of an error (circled) due to ignoring spelling conventions for stepwise motion. Errors of the second type can be caused by insensitivity to either subtle key changes or rapid adoption of a new (and distant) key context. Figure 8(b) shows an example of errors (circled) due to an inability to adapt quickly enough to a new key context (book-ended by the double bar lines). As before (in Figure 1), the notation shown is the correct spelling, and the circles indicate places in the music where a misspelling (not shown) occurred.

Approaching the Bootstrapping Algorithm

A natural solution to the lack of sensitivity in local contextual changes in the cumulative window method is to use a sliding window approach. Ensuing experiments using a sliding window approach produced the results shown in Table 7. The tests were carried out using only the more challenging Beethoven Op.109 example so as to better detect spelling improvements. As expected, the sliding window allowed the contextual CE used in the pitch spelling assignments to be more sensitive to local key changes and the number of spelling errors was reduced from the original 73 for the cumulative window algorithm (shown in Table 6) to 31 for $w_s = 4$ in the sliding window algorithm (see Table 7).

Table 7: Table of results for sliding window experiments using the Beethoven Op.109 example.

Parameters			No. of errors	% correct
w_s	w_r	f	(total: 1516 notes)	
4	0	1	**31**	**98.00**
8	0	1	47	96.90
16	0	1	40	97.36

Although the sliding window approach eliminated many of the errors due to local key changes, it remained insensitive to sudden key changes, unable to adapt quickly enough to a switch to an unexpected new key, which occurs on several occasions in the Beethoven Op. 109 example. The sliding window algorithm's inability to adapt to sudden key changes resulted in the design of the bootstrapping algorithm described in this paper.

Bootstrapping Algorithm Experiments

We next applied the bootstrapping algorithm to the Beethoven Op.109 example and the computational results are shown in Table 8. The algorithm was tested using various parameter values for the phase I window size, w_s, the phase II window size, w_r and the relative weight of the local context versus the cumulative context, *f*. The best result had 27 spelling errors (out of a total of 1516 notes) and was accomplished by the following parameter values: (w_s, w_r, *f*) = (4,3,0.8), (4,3,0.7) and (8,6,0.9). The next best result had 30 errors using the parameters (8,6,0.8) and (16,6,0.8).

Since the best results were achieved with high values for *f*, we can deduce that the local context is more important than the global context in pitch spelling. Consequently, a purely sliding window method should perform better than a purely cumulative window method, as exhibited by the results shown in Tables 6 and 7. However, the better results achieved by the combination of both a local and cumulative effect in the bootstrap algorithm implies that pitch spelling, like tonal perception, is a function of both local and global tonal contexts.

Table 8. Table of results for bootstrapping approach for Beethoven Op.109 example.

Parameters			No. of errors	% correct
w_s	w_r	f	(total: 1516 notes)	
4	2	0.6	28	98.15
4	3	0.9	32	97.89
4	3	0.8	**27**	**98.22**
4	3	0.7	**27**	**98.22**
8	2	0.9	31	97.96
8	2	0.8	40	97.36
8	2	0.7	40	97.36
8	4	0.9	40	97.36
8	4	0.8	40	97.36
8	4	0.7	43	97.16
8	6	0.9	**27**	**98.22**
8	6	0.8	**30**	**98.02**
8	6	0.7	47	96.90
16	4	0.8	40	97.36
16	4	0.7	40	97.36
16	6	0.9	31	97.96
16	6	0.8	**30**	**98.02**
16	8	0.9	37	97.56
16	8	0.7	41	97.30

Tests on the *Song of Ali-Shan*

Having shown the pitch-spelling algorithm to be effective in assigning pitch names for classical music by Beethoven, we next tested it on a Chinese piece by Taiwanese composer You-Di Huang. The pitch spelling experiment was performed using the some of the best parameters from Table 8, all the parameters from Table 7 and the cumulative window parameters. The list of experiments is documented in Table 9. The bootstrap parameters are (4,3,0.8), (8,6,0.9) and (16,6,0.8); and, the sliding window parameters are (4,0,1), (8,0,1) and (16,0,1). In spite of the shifting meters and pitch contexts in the piece, all experiments returned 100% perfect spellings (1571 notes out of 1571 notes).

It is important to note that the results, although seemingly uninformative due to its perfection, is not a trivial one. In the piece, not only does the key signature change, the meter (time structure) is also subject to variation while the window sizes, w_s and w_r, remained constant. As a point of comparison, the default Java MIDI-to-note name translator, part of the Java Development Kit, returned results that had 156 errors (a 10% error rate). Recall that the Beethoven Op.79 example also performed well (only 1 error in 1375) using just the cumulative window. This suggests that the pitch name assignment technique is reasonably robust for pieces of intermediate complexity, and the bootstrapping only improves the results in cases where the key changes tend to the extreme.

Table 9. Table of results for experiments using "The Song of Ali-Shan".

Parameters			No. of errors	% correct
w_s	w_r	f	(total: 1571 notes)	
0	0	0	0	100
4	0	1	0	100
8	0	1	0	100
16	0	1	0	100
4	3	0.8	0	100
8	6	0.9	0	100
16	6	0.8	0	100

Compiling the results from the three sets of test data, we gather the overall pitch spelling results for parameters (4,3,0.8) and (8,6,0.9) in Table 10. A composite result using these parameters for the Op.79 and Ali-Shan examples yields a total of 1 error out of 2496 notes (that is to say, 99.97% correct). To better estimate the performance of the algorithm on a test set of more varied difficulty, we include the Op.109 results to obtain the overall performance estimate. Overall, there were 28 spelling errors out of a total of 4462 notes, giving an estimated percentage correct rate of 99.37%. The errors remaining are primarily due to voice leading conventions for chromatic motion, which result in pitch spellings that are not consistent with the local key context or its most closely related keys. For example, in Figure 1(a) and (b), the F double sharp is the result of linear upward motion, and not the local key context, C sharp minor. A simple rule-based system for detecting linear chromatic motion may produce more errors than it corrects. To effectively eliminate voice-leading errors, one approach is to separate out the various voices in a polyphonic piece, a topic for another paper.

Table 10. Overall pitch spelling results.

Piece	No. of notes	No. of errors	% correct
Beethoven Op.79 (3rd mvt)	1375	1	99.93
Beethoven Op.109 (1st mvt)	1516	27	98.22
You-Di Huang's *Song of Ali-Shan*	1571	0	100.00
OVERALL RESULT	4462	28	99.37

Conclusions

This paper has presented the problem of assigning pitch names to MIDI note numbers. Pitch spelling is an essential component in automated transcription and computer analysis of music. Proper naming of pitch numbers results in more accurate key tracking and chord recognition algorithms. Better algorithms for transcription and analysis improve the state-of-the-art in music retrieval and other interactive music applications.

A bootstrapping framework for real-time pitch name assignments using the Spiral Array model was described which used a variety of data windows to summarize both local and global key contexts. The bootstrapping framework captures both the cumulative window as well as the sliding window variations on the algorithm. The proposed algorithms provide accurate spellings in real time, are computationally efficient and scale well to large data sets. The algorithms were tested on three different test sets: movement 3 of Beethoven's *Piano Sonata No.25 in G major*, Op.79, movement 1 of Beethoven's *Piano Sonata No.30 in E major*, Op.109, and a set of variations on the *Song of Ali-Shan*. The challenging Beethoven Op.109 example inspired the bootstrapping two-step assignment algorithm described in this paper. The first and third test sets posed little problems for all variants of the bootstrapping algorithm, including the sliding window and the cumulative window algorithms. The best overall correct assignment rate was 99.37%.

The two main features of our proposed algorithm are: (1) the use of the Spiral Array, and (2) the bootstrapping algorithm. Future research that separates the two for independent testing will help reveal the relative importance of each component to accurate pitch spelling and lead to the development of better algorithms. Such tests can incorporate aspects of other researchers' models with one of the two components identified, for example, the Spiral Array with Cambouropoulos' windowing approach, or the bootstrapping algorithm with the Line-of-Fifths representation.

As part of future work, more urgency will be placed on the testing of our pitch spelling algorithm on databases such as Meredith's for comparison of our method to other spelling algorithms. We have shown in this paper the algorithm's behavior over a range of parameter values and reported an overall performance estimate of 99.37%. Included in this estimate was the result from the data used in development in order to capture the performance of the algorithm over a more varied test set in terms of tonal complexity. This re-use of development data for the performance estimate can be avoided in the future by conducting quantitative and comparative tests using a larger test set. A large corpus will also result in more statistical significance results regarding the performance of the proposed algorithm.

Based on our experience with a limited test set, it appears that the numerical differences between the performance scores of the different algorithms would be larger if the input data were tonal but more complex. For example, late Beethoven is likely to reveal the differences between the pitch spelling algorithms more distinctly than early Beethoven. This leads us to suggest that future testing should also be performed on a larger corpus of complex tonal music. It would also be an interesting exercise to test the pitch spelling algorithm on contemporary (modern and post-modern) compositions, many of which are tonal in nature, or atonal but adhering to the conventions of tonal spelling.

Acknowledgements

We thank the referees and editors who have provided many detailed comments and helpful suggestions for improving this manuscript. The research has been funded by, and made use of shared facilities at, USC's Integrated Media Systems Center, an NSF ERC, Cooperative Agreement No. EEC-9529152. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of NSF.

References

Cambouropoulos, E. 2001. "Automatic Pitch Spelling: From Numbers to Sharps and Flats." In *Proceedings of the VIII Brazilian Symposium on Computer Music*, Fortaleza, Brazil.

Cambouropoulos, E. 2003. "Pitch Spelling: A Computational Model." *Music Perception*, 20:411–429.

Chew, E. 2000. *Towards a Mathematical Model of Tonality*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Chew, E. 2001. "Modeling Tonality: Applications to Music Cognition." In J. D. Moore and K. Stenning, eds. *Proceedings of the 23rd Annual Mtg of the Cognitive Science Soc.* Edinburgh, Scotland: Lawrence Erlbaum Assoc. Pub., pp. 206–211.

Chew, E. 2002. "The Spiral Array: An Algorithm for Determining Key Boundaries." In C. Anagnostopoulou, M. Ferrand and A. Smaill, eds. *Music and Artificial Intelligence – Proceedings of the Second Intl Conference on Music and Artificial Intelligence*. Edinburgh, Scotland: Springer LNCS/LNAI #2445, pp. 18–31.

Cohn, R. 1998. "Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective. *Journal of Music Theory* 42:167–180.

Krumhansl, 1990. *Cognitive Foundations of Musical Pitch*. Oxford University Press. Oxford: Oxford University Press.

Longuet-Higgins, H. C. 1962a. "Letter to a Musical Friend." *Music Rev* 23:244–248.

Longuet-Higgins, H. C. 1962b. "Second Letter to a Musical Friend." *Music Rev* 23:271–280.

Longuet-Higgins, H. C. and Steedman, M. J. 1971. "On Interpreting Bach." *Machine Intelligence* 6:221–241.

Longuet-Higgins, H. C. 1976. "The Perception of Melodies." *Nature* 263:646–653.

Meredith, D. 2003. "Pitch Spelling Algorithms." In *Proceedings of the Fifth Triennial ESCOM Conference.* Hanover University of Music and Drama, Germany. pp. 204–207.

Meredith, D. 2004. "Comparing Pitch Spelling Algorithms on a Large Corpus of Tonal Music." In U. K. Wiils (ed.) *Computer Music Modeling and Retrieval 2004*. Esbjerg, Denmark: Springer-Verlag LNCS #3310.

Sleator, D. and Temperley, D. 2001. The Melisma Music Analyzer. Available online at *http://www.link.cs.cmu.edu/music-analysis*

Temperley, D. 2001. *The Cognition of Basic Musical Structures*. Cambridge, MA: MIT Press.