Automatic Pitch Spelling: From Numbers to Sharps and Flats

Emilios Cambouropoulos
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, 1010, Vienna, Austria
emilios@ai.univie.ac.at
http://www.ai.univie.ac.at/~emilios

Abstract

In this paper a computational model is described that transcribes polyphonic MIDI pitch files into the Western traditional music notation. Input to the proposed algorithm input is merely a sequence of MIDI pitch numbers in the order they appear in a MIDI file. No a priori knowledge is required such as key signature, tonal centers, time signature, voice separation and so on. Output of the algorithm is a sequence of 'correctly' spelled pitches. The algorithm was evaluated on 8 complete piano sonatas by Mozart and had a success rate that is greater than 96% (10476 pitches were spelled correctly out of 10900 notes that required accidentals – overall number of pitches in 8 sonatas is 40058). The proposed algorithm was also compared to and tested against other pitch spelling algorithms. Pitch spelling algorithms are important not only for applications such as musical notation software packages but also for a multitude of tonal analytical tasks such as key-finding and harmonic analysis.

1 Introduction

In computer applications pitch is most commonly encoded as MIDI pitch numbers. In tonal music, however, enharmonic spelling of pitches conveys useful information about diatonic scales, tonal centres, harmonic and melodic salience, and so on. It is therefore often useful to have access to the 'correct' pitch spelling which may facilitate other musical tasks such as harmonic analysis, melodic pattern matching, motivic analysis etc. Depending on the musical task at hand, a more refined representation, such as the traditional pitch representation, may be more efficient (despite its seeming redundancy at the lowest pitch level) as it allows higher level musical knowledge to be represented and manipulated in a more precise and parsimonious manner.

Perhaps the most obvious and practical use of a pitch spelling algorithm is transcription of MIDI pitch into traditional note names for musical notation software applications. Most musical notation packages allow the user to set manually the key signature of a musical work. As this initial key signature determines a fixed spelling of pitches for the full length of the piece, spelling can be severely disrupted in many cases such as abrupt modulations (see Figure 5). A flexible pitch spelling algorithm could be useful for such applications.

In this paper a simple and effective pitch spelling algorithm will be presented. The algorithm transcribes polyphonic MIDI pitch files into the Western traditional pitch notation with a success rate that is greater than 96% (the algorithm was tested on 8 complete piano sonatas by Mozart – 424 pitches were misspelled out of 10900 notes that required accidentals – overall number of pitches in 8 sonatas is 40058). No a priori knowledge is required such as key signature, tonal centers, time signature, voice separation and so on. The input data is merely a sequence of MIDI pitch numbers in the order they appear in a MIDI file. The output of the algorithm is a sequence of 'correctly' spelled pitches.

In the first part of this paper some theoretical issues regarding pitch intervals will be discussed and two existing approaches to the pitch spelling problem will be presented. In the second part, a spelling algorithm will be described and some extensive evaluation and comparison tests will be presented. In the course of this discussion, musical examples will be given that highlight various aspects of the pitch spelling task.

2 Pitch Spelling and Interval Optimisation

Pitch spelling can be seen as a process that follows naturally the application of key-finding algorithms (Longuet-Higgins & Steedman 1971; Bharucha 1987; Krumhansl 1990; Vos & Van Greenen 1996). Longuet-Higgins and Steedman suggest that after a key is determined and relations of non-key notes to the main key notes are established 'it is a trivial matter to transcribe the solution into standard musical notation' (Longuet-Higgins & Steedman 1971). Rowe (2000) proposes a spelling algorithm that is based on a stacked-thirds technique; this algorithm also requires some tonal pre-processing, namely, that the root of each chord is determined in advance (Parncutt's root determination algorithm is used – Parncutt 1997).

It is less common to have pitch spelling algorithms used as precursors to harmonic analysis (one such case is Temperley 1997). Of course the tasks of spelling pitches and key finding are strongly linked as they both relate in one way or another to properties of diatonic scales and more generally to the hierarchic organisation of pitches in a tonal system. It is interesting, however, to explore the possibilities of notating a musical score correctly without having access to established tonal regions and keys.

A pitch spelling algorithm, that has been developed by the author (Cambouropoulos 1996), selects appropriate traditional pitch names based on a transcription procedure that optimises the "quality" of traditional intervals, i.e. it avoids diminished, augmented and chromatic intervals - at this early stage, the algorithm was applied only to monophonic pitch sequences even though its extension for polyphonic music is a rather straightforward process as will be shown below. Another pitch spelling algorithm, that is described in (Temperley 1997), spells pitches so that they are as close as possible together on the "line of fifths" (Figure 1); this algorithm can be applied to polyphonic pitch sequences. Both algorithms additionally bias the process so that double-sharps and double-flats are avoided.

What is the relation between these two different procedures? Is there a common underlying principle? Which method is more effective? These issues will be addressed below. It will be shown that both approaches are quite effective but the interval optimisation method yields overall better results. Additionally it will be maintained that the "line of fifths" approach is actually a special case of the "interval optimisation" approach.

... B♭♭ F♭ C♭ G♭ D♭ A♭ E♭ B♭ F C G D A E B F♯ C♯ G♯ D♯ A♯ E♯ B♯ F𝄪 C𝄪...

Figure 1 The "line of fifths" is a stretched out "circle of fifths" (its pitch elements are referred to as *tonal pitch classes*).

Implicit in the principle of using the narrowest possible line-of-fifths spelling band is a hierarchic ordering of pitch intervals: tonal pitch classes closer together are preferred and so are the corresponding intervals they form. The line of fifths implies the ordering of pitch intervals that is depicted in Table 1. For instance, two adjacent tonal pitch classes (one step apart in the line of fifths) form a perfect 5th or perfect 4th, two tonal pitch classes that are in a distance of two steps form a major 2nd or minor 7th, and so on. The pitch spelling algorithm that is proposed by Temperley (1997) is in effect avoiding intervals that correspond to greater distances between pitches in the line of fifths (i.e. avoids intervals in the right hand of Table 1).

Distance	1	2	3	4	5	6	7	8	9	10	11	12
Intervals	4P 5P	2M 7m	3m 6M	3M 6m	2m 7M	4a 5d	1a 1d	4d 5a	2a 7d	3d 6a	3a 6d	2d 7a

Table 1 Ordering of pitch intervals according to the distance of their constituent pitches in the line of fifths (P=perfect; M=major; m=minor; a=augmented; d=diminished; interval 1a/1d is the chromatic semitone interval or augmented/diminished octave).

Cambouropoulos (1996) has proposed a hierarchic classification of pitch intervals according to their frequency of occurrence among the degrees of a given set of scales. For the major-minor framework (i.e. major scale, and harmonic and melodic minor scales) this classification corresponds well to the traditional pitch interval naming system. Perfect intervals that are the most frequent intervals form class A, major and minor intervals form class B, rare intervals such as many augmented and diminished intervals form class C, and intervals not encountered between scale degrees (e.g. augmented and diminished 8ve, augmented 3rd, diminished 6th etc) form class D – see Table 2. This classification seems to be in agreement with music theoretic approaches whereby rare intervals such as class C intervals have a special status/function in tonal music (see Browne 1981) and many of class D intervals have "little beyond a theoretical existence." (The Oxford Dictionary of Music, 2nd edition, 1994). The proposed pitch spelling algorithm prefers intervals from classes A and B whereas the intervals of class D are most strongly avoided.

Class	A	B				C				D		
Intervals	4P 5P	2m 7M	2M 7m	3m 6M	3M 6m	2a 7d	3d 6a	4d 5a	4a 5d	1d 1a	3a 6d	2d 7a

Table 2 Classes of pitch intervals ordered according to their frequency of occurrence in the major-minor scale framework (class D intervals are not found in any major or minor scales).

The main issue here is to determine which ordering of pitch intervals is most adequate for pitch spelling algorithms. In this paper the two above pitch interval hierarchies will be compared and tested against the same musical dataset (section 3.2). Further research, however, is required for establishing the most appropriate ordering *in general* (if a single one exists!).

The pitch interval ordering that is based on the line of fifths is the same with the frequency-of-occurrence ordering for the perfect, major and minor intervals. Some differences emerge in the ordering of the augmented and diminished intervals. For instance the diminished and augmented 1st/8ve (e.g. chromatic semitone) is preferred over the dim. 3rd and aug. 6th, or the aug. 2nd and dim. 7th. Such differences can have a significant impact on the transcription process. Consider, for instance, the example in Figure 2; the first spelling (A) of the four pitches in each staff is given by the frequency-of-occurrence algorithm (dim. 3rd and aug. 6th preferred) whereas the second spelling (B) is given by the line-of-fifths algorithm (chromatic semitone and aug. 8ve preferred). Of course, spelling depends on a broader context but it would seem more plausible that the first spellings (A) are more adequate for a classical tonal context.

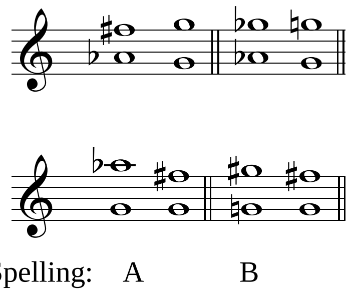


Figure 2 Two sets of pitches (one in each staff) are spelled according to the frequency-of-occurrence ordering of pitch intervals (spelling A) and to the line-of-fifths ordering (spelling B).

It is important to establish a hierarchic ordering between *enharmonic* spellings of intervals (whether a minor 3rd should be preferred over an augmented 2nd) but not between different size intervals (no need to set any preferences among the various perfect, minor and major intervals). The ordering implied by the line of fifths is over-refined at least for pitch spelling tasks. Two preference categories should be sufficient (see Table 3). It should be noted, however, that preferences among the 'less preferred' row of intervals in Table 3 may be useful when additional rules are considered, such as avoidance of double sharps and flats, as these give rise to more complex relationships and optimisation processes.

Num. of Semitones	1 or 11	2 or 10	3 or 9	4 or 8	5 or 7	6
Preferred	2m/7M	2M/7m	3m/6M	3M/6m	4P/5P	4a/5d
Less preferred	1a/1d*	3d/6a	2a/7d	4d/5a	3a/6d*	–

Table 3 Pitch interval preference categories for pitch spelling – intervals indicated by asterisks may form a third even less preferred category (see text).

Finally, it should be noted that, in the context of this study, doubly augmented and doubly diminished intervals (e.g. C*b*-D#) have not been considered as they are extremely rare. This way, for each interval the choice is between two enharmonic interval categories (except for the tritone for which the choice is between the equally preferred augmented 4th and diminished 5th intervals).

3 The Pitch Spelling Algorithm

In this section a simple version of the proposed algorithm is described and evaluated; further extensions and refinements are also proposed.

3.1 The Algorithm

Input to the algorithm is a list of MIDI pitch values (in the order they appear in the original MIDI file) – polyphonic music is represented as a sequence of pitches where simultaneous pitches (i.e. chords) appear in arbitrary order.

The algorithm uses a shifting overlapping windowing technique (Figure 3). All the pitches in each window are spelled according to certain criteria (listed below) but only the ones in the middle one-third section of the window are retained (suggested size of window is 9 or 12 pitches). Then, the window is shifted by one-third and the same process is applied recursively till the end of the pitch sequence is reached. Allowing a larger section to be spelled in each step gives greater stability to the pitch spelling process as a larger pitch context is taken into account and abrupt changes at the edges of the window are avoided.

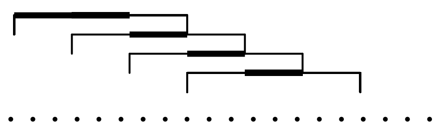


Figure 3 Shifting overlapping window technique - for each window only the middle section of spelled pitches in retained (bold line) - dots represents the pitches of the input sequence.

For each window, the pitch spelling process is based on an optimisation procedure that relies on two fundamental principles:

1) *Notational Parsimony* (i.e. spell notes making minimum use of accidentals)

2) *Interval Optimisation* (i.e. avoid augmented and diminished intervals).

The first principle essentially avoids enharmonic spelling of notes that can be notated without any accidentals, e.g. C is preferred over B# & D♭♭. The second principle attempts to spell notes in a way that the more frequent diatonic intervals are used (i.e. perfect, major and minor intervals) and the augmented and diminished intervals are avoided.

Penalty values are introduced for the notational parsimony principle and for the different categories of intervals presented in Table 2:

Notational Parsimony:	
‘normal’ spelling of note	0
enharmonic spelling of note	4
Interval Optimisation:	
intervals of class A or B	0
intervals of class C	1
intervals of class D	3

Some examples of interval penalty values according to the above principles are presented: for the interval F#-G# the penalty value is p=0, for F-G# $\longrightarrow$ p=1 (aug. 2nd interval), for E#-G# $\longrightarrow$ p=4 (enharmonic spelling of first note), for E#-G $\longrightarrow$ p=5 (enharmonic spelling of first note and interval of category C, i.e. dim. 3rd) and for E-E# $\longrightarrow$ p=7 (enharmonic spelling of second note and interval of category D, i.e. chromatic semitone).

For each window, all possible spelling sequences are computed – sequences that contain *both* double-sharps *and* double-flats are disallowed by creating the different spelling sequences only from within two different tonal-pitch-class areas wherein one area excludes double- sharps and the other excludes double-flats (see Figure 4). This way computational efficiency is also improved (worst case is $2 \cdot 2^n$ spelled sequences where n is the window size – for n=9 there are 1024 sequences). The number of sequences can be reduced by applying further constraints such as rejecting sequences that contain more than one enharmonic spelling (e.g. not more than one double-sharp in a sequence).

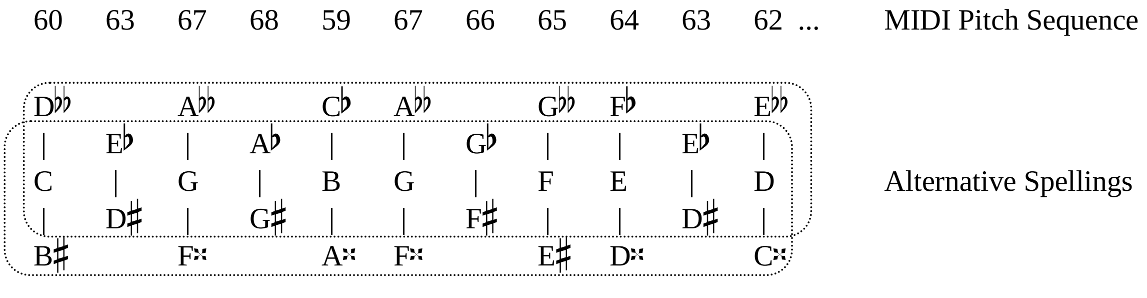


Figure 4 Beginning of the theme of Bach's *Musical Offering* – spelling sequences are selected from within the two different boxes of the available tonal pitch classes.

For each spelled pitch sequence, all the penalty values given above for every possible interval (i.e. all intervals between contiguous and non-contiguous pitches in the sequence) are summed and an overall penalty value is computed. The sequence with the lowest penalty value is selected.

The implementation of the spelling algorithm in this study actually employs a *variable* length window that always contains 9 distinct pitches (repeated pitches are omitted). This way the algorithm avoids unnecessary ambiguity that is introduced when too few different pitches appear in a given window (e.g. in a section that has just 3-4 repeating pitches as in the case of an alberti bass). This variable window length technique improves the performance of the algorithm both in terms of transcription quality (avoiding misspellings due to lack of appropriate pitch context) and efficiency (as larger windows can be used without adding to the computational complexity).

3.2 Evaluation and comparison of pitch spelling models

The proposed pitch spelling algorithm was tested on a set of 8 complete piano sonatas (K279-K283, K331-K333) by Mozart. This dataset comprises of 40058 notes of which 10900 notes are notated with accidentals (natural signs are not counted). The MIDI pitch versions of the sonatas were spelled by the algorithm and compared to the original scores; the mismatches between the two were determined giving a percentage of correct spelled notes over the number of notes with accidentals in the score.

In a first experiment, the spelling algorithm was applied to the musical dataset using the pitch interval ordering in Table 1 that corresponds to the line of fifths. The distance values indicated in the first row of the table were used as penalty values (instead of the values proposed in the previous section). As for the notational parsimony principle an enharmonically spelled note was given a penalty value of 13 (this is larger by one than the highest distance value in Table 1 – more experimentation would be necessary for determining the most appropriate value). This experiment gives the results depicted in Table 4.

Total Num. of Notes	Notes with accidentals	Misspelled notes	Correct Spelling
40058	10900	649	94%

Table 4

In a second experiment, the augmented and diminished 1st interval (includes the chromatic semitone and aug. and dim. 8^{ve}) that appear in position 7 of Table 1 was taken to position 12 of the table and the intervals following it were displaced by one position to the left. The aim of this experiment was to see how the ordering of pitch intervals may affect the spelling process (the pitch spelling program and settings are exactly the same as in the previous experiment). This single change improved the results by 1.4% as can be seen in Table 5.

Total Num. of Notes	Notes with accidentals	Misspelled notes	Correct Spelling
40058	10900	501	95.4%

Table 5

In a final experiment, the frequency-of-occurrence pitch interval ordering (Table 2) with the penalty values given in the previous section. For this ordering the algorithm generates even better results as can be seen in Table 6.

Total Num. of Notes	Notes with accidentals	Misspelled notes	Correct Spelling
40058	10900	424	96.2%

Table 6

This technique of a step-by-step transcription by overlapping sections seems to be close to the processes that take place while a listener is notating little-by-little a heard melody (melodic dictation). The listener hears and notates a few bars at a time making possible alterations to the immediately preceding notes if this is required by the new input. In practical terms, it enables the algorithm to move smoothly over different tonal regions as illustrated in Figure 5.

The proposed algorithm spells correctly large sections of musical works but makes a limited number of spelling errors as well (see examples in Figures 6-9). These errors are due to a number of factors: a) innate problems of the algorithm's principles (e.g. there is a trade-off for the different pitch interval orderings – it is likely that there exists no single ordering that is appropriate in all cases), b) technical problems relating to the edges of selected windows in the shifting overlapping windowing technique, c) problems relating to the limited scope of the current implementation (i.e. voice-leading concerns are not currently addressed neither are various structural factors taken into account) and d) problems innate to pitch spelling processes *per se* (e.g. uncertainty of spelling of diminished 7th chords).

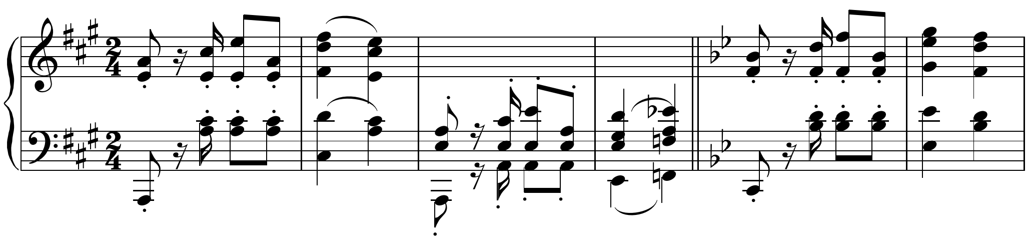


Figure 5 This excerpt from Schumann's *Faschingschwank aus Wien, Op.6:III, Scherzino* contains an abrupt modulation; it is spelled correctly by the algorithm. The overlapping step-by-step transcription process enables a smooth transition into the new tonal region and the successful spelling of pitches in both areas.

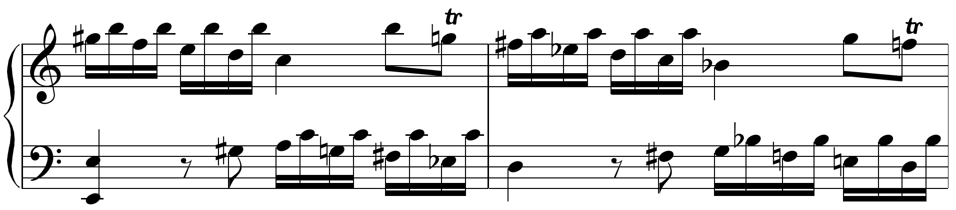
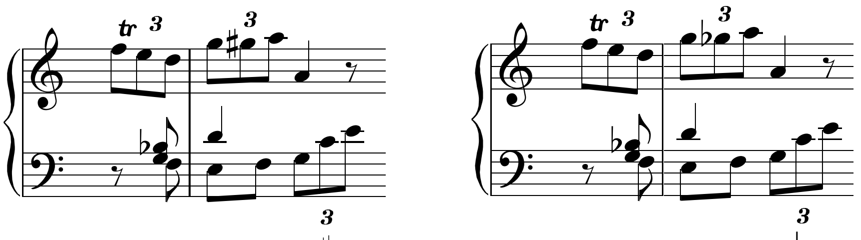


Figure 6 Musical excerpt from Mozart's *Sonata in C major K279* spelled correctly by the algorithm.



Original (G♯)   Incorrect Transcription (A♭)

Figure 7 Excerpt from Mozart's *Sonata in C major K279* – the note G♯ is misspelled as A♭ as the algorithm does not take into account voice leading concerns (the A♭ spelling is selected by as it fits better with the preceding B♭).

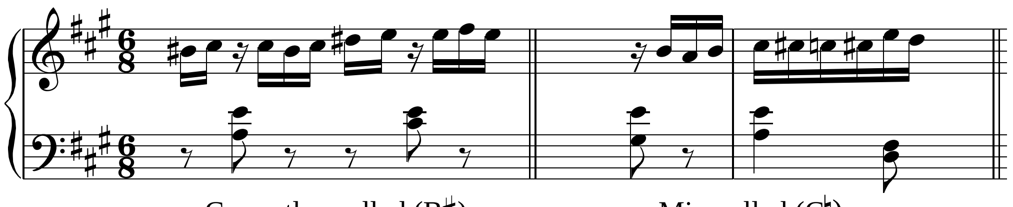


Figure 8 Two excerpts from Mozart's *Sonata in A major K331 (Var.I)* – the first excerpt is spelled correctly by the algorithm (notice the use of B♯) – in the second excerpt the B♯ is misspelled as C♮ because it creates a B♮ - B♯ interval as well as a B♯-D interval which are strongly avoided (the algorithm has no integrated notion of voice leading).

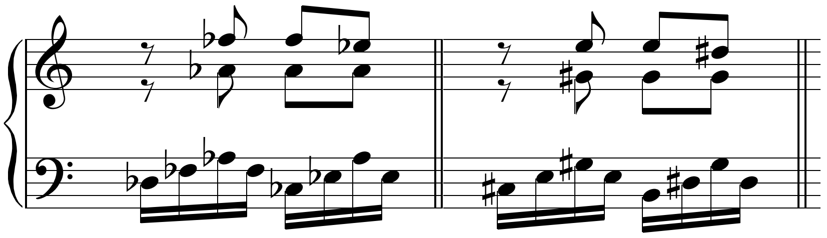


Figure 9 Excerpt from Mozart's *Sonata in B♭ major K333* – the algorithm prefers the more 'economic' spelling with sharps as this spelling avoids the use C♭ and F♭ (of course in the broader context of this sonata section the original spelling is more appropriate).

The main findings from these experiments is that, firstly, algorithms that attempt to spell pitches based on an interval optimisation process are overall very successful. The tonal information embodied in the traditional pitch interval representation is sufficient for spelling correctly the vast majority of pitches in classical tonal music. A second finding is that the ordering of traditional pitch intervals affects the pitch spelling process; further research, however, is necessary for exploring new possibilities that may lead into even better results (pitch interval orderings may well be different for various musical styles and idioms).

3.3 Further improvements

For reasons of clarity and succinctness, the simplest possible spelling approach was used and described in this study (this approach nevertheless produces very good results and reveals its potential). There are a number of ways, however, that the algorithm can be enhanced further.

The main improvement would be to use elementary musical knowledge to guide the spelling process. The present version of the algorithm is applied merely to the sequence of MIDI pitches. Timing information, metrical structure, note accentuation and so on, however, can be taken into account during the spelling process. For instance, timing information can be used to calculate the distance of pitches from the centre of a window and then allow pitches that are closer together to have a stronger effect on the spelling process, i.e. pitches that are further away should contribute less to the overall spelling penalty. It seems plausible that notes such as secondary ornamental notes (e.g. passing and neighbour notes) should affect less the tonal core of a given musical section; intervals between notes that appear on metrically stronger positions or are more accented (e.g. longer duration, extreme pitch register etc) should contribute more to the overall spelling penalty value.

Voice leading is also an important component of pitch spelling. In order to take voice leading into account the various parts/voices of musical work must be known (streaming algorithms are necessary). If the various melodic streams are pre-determined additional rules can cater for voice leading effects. For instance, such a rule, for melodic sequences comprised of three notes, is proposed in (Cambouropoulos 1996): *Amongst equally rating spellings, prefer the spelling in which higher 'quality' intervals appear last*. This rule accounts for asymmetric temporally-ordered aspects of musical perception (Deutsch, 1984; Krumhansl, 1990) according to which listeners, for example, tend to hear the last note of an interval as more prominent. When there are two alternative spellings of two intervals the system should prefer the sequence in which the last interval belongs to a 'better' quality class. This rule gives precedence, for instance, to the sequence G - G♯ - A over the equivalent G - A♭ - A, or to the sequence A - A♭ - G over the equivalent spelling A - G♯ - G.

Structural relationships between notes, mainly in the temporal domain, may contribute to establishing a more refined hierarchic organisation of pitches and intervals which in turn can improve the spelling method.

Conclusions

In this paper an algorithm was presented that attempts to transcribe polyphonic MIDI pitch files into the traditional pitch notation. This simple pitch spelling algorithm has been shown to produce very good results (success over 96%) for music of the classical tonal period. The basic underlying principle of traditional pitch interval optimisation encapsulates important properties of diatonic scales and tonal hierarchies, and is capable of guiding successfully the pitch spelling process. A pitch spelling algorithm, such as the one suggested herein, can be very useful for many applications, from score extraction programs to key finding and tonality inducing processes.

Acknowledgements

This research is part of the project Y99-INF, sponsored by the Austrian Federal Ministry of Education, Science, and Culture in the form of a START Research Prize and support to the Austrian Research Institute for Artificial Intelligence. I would like to thank the Bösendorfer company for providing the data used in this study.

References

Browne, R. (1981) Tonal Implication of the Diatonic Set. *In Theory Only,* 5(6,7):3-21.
Bharucha, J.J. (1987) Music Cognition and Perceptual Facilitation: A Connectionist Framework. *Music Perception,* 5:1-30.
Cambouropoulos, E. (1996) A General Pitch Interval Representation: Theory and Applications. *Journal of New Music Research,* 25(3):231-251.
Deutsch, D. (1984) Two Issues Concerning Tonal Hierarchies: Comment on Castellano, Bharucha & Krumhansl. *Journal of Experimental Psychology: General,* 113:413-416.
Longuet-Higgins, H. C. and Steedman M.J. (1971) On Interpreting Bach. In *Machine Intelligence* (Vol. 6), B.Meltzer & D.Michie (Eds.). Edinburgh University Press, Edinburgh.
Krumhansl, C.L. (1990) *Cognitive Foundations of Musical Pitch.* Oxford University Press, Oxford.
Parncutt, R. (1997) A Model of the Perceptual Root(s) of a Chord Accounting for Voicing and Prevailing Tonality. In *Music, Gestalt and Computing: Studies in Cognitive and Systematic Musicology*, M. Leman (Ed.). Springer, Berlin.
Rowe, R. (2000) Key Induction in the Context of Interactive Performance. *Music Perception* 17(4):511-530.
Temperley, D. (1997) An Algorithm for Harmonic Analysis. *Music Perception* 15(1):31-68.
Vos, P.G. and Van Greenen E.W. (1996) A Parallel-Processing key-Finding Model. *Music Perception* 14(2):185-223.