WELL-TEMPERED SPELLING: A KEY-INVARIANT PITCH SPELLING ALGORITHM

Joshua Stoddard
U. Mass., Amherst
Math & Stat Department

Christopher Raphael
U. Mass., Amherst
Math & Stat Department

Paul E. Utgoff
U. Mass., Amherst
Comp. Sci. Department

ABSTRACT

In this paper is described a data-driven algorithm for the functionally correct spelling of MIDI pitch values in terms of Western musical notation. Input is in the form of MIDI files containing accurate pitch and rhythmic information with corresponding ground-truth spelling information for training and evaluation. The algorithm recovers harmonic information from the MIDI data and spells pitches according to their relation to the local tonic. The algorithm achieved 94.98% accuracy on the pitches that required accidentals in the local key and 99.686% overall. Voice-leading resolution was found to be the best feature of those used to infer the correct spelling. Also, this paper outlines great potential for improvement under this model.

1. INTRODUCTION

In MIDI, pitch information is encoded as an integer pitch-level value. The pitch-level, however, does not uniquely determine the spelling in Western music notation [4]. Different spellings are called *enharmonically equivalent* if they map to the same pitch-level. *Pitch spelling* is the process of retrieving the spelling information lost in the pitch-level representation of pitch.

One obvious application of pitch spelling is in the translation from MIDI to Western music notation. Currently, most music notation software can perform rudimentary pitch spelling on MIDI data, but the results are often prone to enharmonic errors. The spelling of a pitch is strongly influenced by its harmonic and melodic context. This higher-level contextual information often can be retrieved reliably from the pitch-level information in MIDI data, though the problem is highly non-trivial [6]. Thus, a pitch spelling algorithm that can retrieve and make use of more contextual melodic and harmonic information may produce more accurate results.

This paper presents a data driven algorithm for pitch spelling in a harmonic context. The algorithm assigns spellings to pitches in polyphonic, rhythmically accurate MIDI data according to a harmonic parse generated by existing harmonic analysis software [6] and a decision tree structure generated automatically from training data using Breiman et. al.'s CART [1]. On the test data, it showed a success rate of 94.98% (misspelled 70 of 1395 notes) on the cases in which the spelling was not completely determined by the harmonic parse, and 99.686% overall (misspelled 71 of 22,593 total notes). Later, 26 of the 71 'misspelled' cases were discovered to be errors in the 'ground-truth' spelling data with which they were compared. The overall accuracy of the algorithm is limited by the accuracy of the harmonic parse, which is generated independently of the rest of the algorithm. However, the fact that the algorithm was able to capture errors in the ground-truth data speaks to its robustness under imperfect circumstances

The algorithm presented here differs from existing pitch spelling algorithms such as those of Cambouropoulos [2], Meredith [5] and Chew & Chen [3] in that it views pitch spelling as independent, key-invariant Boolean classification problems on the pitch levels falling outside the local key. This means that the pitch-levels that appear in the key signature of the local key are spelled accordingly, and the remaining pitches are spelled relative to these. *Key-invariance* is the assumption that the spelling of a pitch-level given its position relative to the local tonic is independent of the local key. This is in keeping with the fact that in well-tempered tuning, all keys of the same mode (major, for example) have the same harmonic structure.

2. PITCH SPELLING

Pitch spelling serves two functions: to make printed music harmonically consistent, and to make it easy for a musician to read. For discrete-pitch instruments (eg. piano), enharmonic discrepancies in spelling have no effect on intonation, but on a continuous pitch instrument (eg. violin) there is a subtle but audible difference between enharmonically equivalent pitches. This means enharmonic errors can affect the intonation in machine-generated music, but most human musicians will automatically play the most harmonically appropriate enharmonic equivalent to what is printed.

Pitch-Level:	60	61	62	63	64	65	66	67	68	69	70	71
Spellings:	D𝄫 C B♯	D♭ C♯	E𝄫 D C𝄪	E♭ D♯	F♭ E D𝄪	G𝄫 F E♯	G♭ F♯	A𝄫 G F𝄪	A♭ G♯	B𝄫 A G𝄪	B♭ A♯	C♭ B A𝄪

Figure 1. Possible spellings for different pitch-levels

Thus, enharmonic discrepancies are rarely audible in human performance, but enharmonic errors detract from the readability and correctness of a printed score.

A musician needs to be able to read, digest, and anticipate musical ideas from a score in real time, so ease in reading is very significant. Ease in reading can mean different things in different contexts and to different musicians, but in an overwhelming majority of cases there is a unique best spelling for each pitch.

Some of the most important considerations for the spelling of a piece are harmonic consistency, voice-leading consistency, and notational parsimony. *Harmonic consistency* refers to the idea that chords have well-defined functions and spellings. For example, a C major triad consists of the pitches c, e, and g, and any other enharmonic spelling of this set of pitches would refer to a functionally different chord. The harmonic function of a chord is determined by its harmonic context. In tonal music, chords have specific ways in which they are expected to appear and to resolve, and these expectations are held subconsciously by the listener. Thus, if the harmonic parse is known, the function and hence the spelling of each chord is uniquely determined (with a few exceptions, which will be touched on later). Harmonic consistency is generally the most important consideration in pitch spelling as practiced by humans, and hence enharmonic equivalency literally means 'equivalent up to harmonic functionality.' Harmonic consistency is clearly only applicable to music with a traditional sense of tonality.

Voice-leading consistency refers to the melodic functionality of each note within a single voice. Melodic, or voice-leading functionality is a property of individual pitches in relation to their immediate neighbors. Contextual implications in voice-leading are not as specific as they are for chords, so retrieving voice-leading information from the melodic context is not as accurate as retrieving harmonic data from a harmonic parse. Fortunately, voice-leading functionality is consistent with harmonic functionality, so harmonically consistent spellings automatically exhibit voice-leading consistency. There are a few exceptions to this property, but in those cases the harmonic consistency prevails. Thus, in conjunction with harmonic consistency, voice-leading consistency need only be concerned with the pitches not captured in the harmonic parse. These cases are called *non-chord tones* (NCTs), and their contextual implications are more specific in general.

Notational parsimony requires the spelling of a piece with a minimum of printed accidentals. In theory, if harmonic information is available, this is accomplished by parsing the piece by tonal center (or tonic) such that a maximum of pitches can be spelled in the key-signature of the local tonic without accidentals. In general, this is not equivalent to spelling everything with a natural (♮) whenever possible. In practice, printing a score directly from such a parse (ie. with an absolute minimum of printed accidentals) is likely to change key signatures frequently enough to be awkward and obscure some of the global structure of the piece, but that issue lies outside the scope of this paper.

For a given pitch-level there are several possible enharmonically equivalent spellings, so by necessity, keys of the same mode built on enharmonically equivalent spellings are themselves enharmonically equivalent. Since harmonic function is defined relative to the tonic, spellings based on enharmonically equivalent keys are harmonically identical, effectively negating concern over harmonic consistency. Thus, in the interest of readability and notational parsimony, it is reasonable to assume that any key whose key signature exceeds 7 sharps or flats will be spelled as an enharmonic equivalent with fewer sharps/flats. For example, G♯ major (8 sharps) would be spelled as A♭ major (4 flats).

For a given key and for all pitch-levels that require accidentals when notated in that key (ie. pitch-levels for which no spelling falls within the key signature) there are pitches one half-step in either direction that do fall within the key signature. Thus, it is reasonable to expect that in the context of a key, every pitch-level requiring an accidental will be spelled as either its lower neighbor raised by a half-step or its upper neighbor lowered by a half-step. Under these two assumptions, the space of possible spellings is limited to those in Figure 1.

If the local tonic is known, the space of possible spellings can be reduced further. Given the local tonic, it is harmonically consistent to spell each pitch-level without accidentals (ie. according to the key signature) whenever possible. For example, in a G major passage MIDI pitch-level 66 would always be spelled as f♯ instead of g♭, and in C major all pitch-levels corresponding to white keys on the piano would be given their natural (♮) spellings, as in Figure 2.

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

Figure 2. Possible spellings in C major

Thus, given a harmonic parse, the space of possible spellings at any point in a piece is a subset of those in Figure 1, determined by the local tonic. This means that with an accurate harmonic parse and under the assumption of key-invariance, pitch spelling is reduced to a Boolean classification problem on the five pitch-levels that cannot be spelled without accidentals. In C, they are the black keys on the piano. These cases have fundamentally different functions relative to the tonic, and to capture that, the algorithm treats them as independent classification problems. Thus, it treats spelling c♯ vs. d♭ in C as fundamentally the same as spelling f♯ vs. g♭ in F, but different than spelling f♯ vs. g♭ in C.

It is important to notice that the harmonic feature of interest here is the key *signature* of the local tonic, not the local tonic itself, and not necessarily the composer's key signature. Thus relative keys (eg. C major and A minor) would fall under the same heading. For the remainder of the paper, *local key* refers to the family of keys that share the same key signature.

3. ALGORITHM

The following algorithm is designed to determine which features in the harmonic and/or melodic context of a note are helpful in recovering the accurate spelling information. The algorithm takes MIDI data as input and generates a corresponding harmonic parse by applying the method described in [5]. To generate an accurate harmonic parse, this model requires that the MIDI data contain accurate rhythmic information. A local key parse is then extracted from the harmonic parse. For each note *n* with pitch-level *p* that must be spelled with an accidental in the local key, several features are calculated. Later, these features will be evaluated to determine which are most informative.

1) *History Vector* (H[12]): the distribution of the 12 MIDI pitch-values modulo the octave in a window of a pre-specified length immediately preceding *n* relative to the local key at *n*.[1]

2) *Future Vector* (F[12]): the distribution of the 12 MIDI pitch-values modulo the octave in a window of a pre-specified length immediately following *n* relative to the local key at *n*.

These features are designed to capture general information about the harmonic context before and after *n*, relative to the local key at *n*. The motivation here is that the two spellings considered for pitch-level *p* are related harmonically to *k* via some number of steps along the circle of fifths in opposite directions. A pitch *s* relates to a key harmonically in terms of the distance (a whole number) and direction (a Boolean value) stepwise along the circle of fifths from the local key to the closest key for whom *s* is a member (ie. Spelled without an accidental). The two spellings considered for *p* are raised and lowered versions of pitches in *k*, so they refer to sharpening (adding sharps or removing flats from *k*) and flattening (adding flats or removing sharps from *k*) motion along the circle of fifths respectively. Thus, if a particular pitch *s* is the correct spelling of *p*, it may be reasonable to see energy in H and/or F corresponding to pitches that lie between *k* and *s* on the circle of fifths. For example, a♭ is 3 steps in the flat direction from C major and g♯ is 3 steps in the sharp direction, so if H and F contain significant energy at pitch-levels corresponding to b♭ (1 step flat) and e♭ (2 steps flat) but not those corresponding to f♯ (1 step sharp) or c♯ (2 steps sharp), this may imply a♭ is preferable to g♯.

3) *History Key Gradient* (ΔK_H): the average difference and direction along the circle of fifths (expressed as a single floating point value) between the local key at *n* and the local keys in a window immediately preceding *n*, weighted by rhythmic proximity to *n*.

4) *Future Key Gradient* (ΔK_F): the average difference and direction between the local key at *n* and the local keys in a window immediately following *n*, weighted by rhythmic proximity to *n*.

These features are designed to capture information about the rate of change in local key upon arriving at *n*. The function of these features is similar to that of H and F except that they are computed in terms of the harmonic parse rather than pitch-level information.

5) *Resolution* (R): a ternary feature that attempts to capture voice-leading information about the resolution of *n*. It scans the piece after *n* for the first appearance of pitch *p*+1 or *p*-1. Whichever appears first determines R as +1 or -1 respectively. If neither appears, or if they appear simultaneously, R is 0.

This is by no means an exhaustive model for resolution in general, but it captures the majority of cases. In particular, it is sufficient to capture resolution information in chromatic NCTs.

Ground-truth files containing the 'true' spellings are then given to decision tree software, which automatically generates the spelling algorithm using different sets of the above features. The ground-truth and MIDI data are both generated from MusicXML data. MusicXML is a format designed as a link between different formats for high-level representation of music. In particular, it encodes the necessary spelling information and can be easily translated into MIDI. Unfortunately, MusicXML is not yet widely used, and there is currently not much data available in this format.

[1] More precisely, the tonic of the major key corresponding to the local key signature at *n*

4. RESULTS

4.1. Data, Priors and Results

The algorithm was run on a set of 31 movements from 15 chamber music pieces by various composers, divided into training and test sets as per Table 2. All the data is from Project Gutenberg [www.gutenberg.net/music], currently the only online archive of public domain sheet music in MusicXML format to our knowledge.

The reduction of pitch spelling to a key-invariant Boolean classification problem relies heavily on having an accurate harmonic parse. The training data was pruned to eliminate pathological cases resulting from an imperfect harmonic parse. The test data was not pruned.

Table 1 shows the prior distributions[2] of all possible spellings in the training and test data respectively on each of the Boolean cases (ie. requiring accidentals). Each of these cases is classified according to its possible spellings in C major, though it is important to remember that these cases do not have the same spellings when they appear in other keys.

Train	c♯/d♭	d♯/e♭	f♯/g♭	g♯/a♭	a♯/b♭
Raised	**200**	**95**	**309**	**779**	14
Lowered	7	57	0	77	**254**
Total	**207**	**152**	**309**	**856**	**268**
Prior	96.6%	62.5%	100%	91.0%	94.8%

Table 1a. Distribution of spellings on the cases requiring accidentals in the training data

Test	c♯/d♭	d♯/e♭	f♯/g♭	g♯/a♭	a♯/b♭
Raised	**139**	**83**	**291**	**568**	14
Lowered	12	71	0	34	**183**
Total	**151**	**154**	**291**	**602**	**197**
Prior	92.1%	53.9%	100%	94.4%	92.9%

Table 1b. Distribution of spellings on the cases requiring accidentals in the test data

Unfortunately, f♯/g♭ is a degenerate case in this data as it is always spelled as the raised fourth scale degree (rather than the lowered fifth). The raised fourth scale degree is more closely related to the home key than the lowered fifth via the circle of fifths (1 step sharp vs. 5 steps flat), so a biased prior was expected. It is unlikely, though, that the lowered fifth scale degree never appears in practice.

[2] The *prior distribution* on a scale degree is the relative frequency of one spelling over the other in the data set.

Composer	**Piece**	**Training**	**Test**
Bach, J. S.	BWV 1047	I – II	-
	BWV 1050	-	II
Beethoven	Op. 18 No. 1	III	-
	Op. 59 No. 2	-	IV
	Op. 59 No. 3	-	III
Brahms	Op. 51 No. 1	II	-
Haydn	Op. 1 No. 1	II – III	V
	Op. 74 No. 1	I & III	-
	Op. 74 No. 2	-	II
Mozart, W. A.	K. 80	I – II	III
	K. 155	I – II	III
	K. 156	I – II	III – IV
	K. 458	II	III – IV
Schubert	Op. 125 No. 1	I & III	IV
Schumann, R.	Op. 41 No. 1	II – III	-

Table 2. Training and Test data

Of the four non-degenerate cases, Table 3 shows the accuracy of the algorithm on the test data under several sets of features. For the first three, the resolution feature was by far the most informative. For a♯/b♭, resolution was not informative, but the other features produced a slight win over the prior.

Feature set	c♯/d♭	d♯/e♭	g♯/a♭	a♯/b♭
All Features	92.1%	70.8%	94.4%	93.4%
ΔK_F, ΔK_F, R	92.1%	81.2%	94.4%	**94.4%**
R only	**92.7%**	**82.5%**	**96.0%**	92.9%
Priors	92.1%	53.9%	94.4%	92.9%

Table 3. Accuracy of the algorithm under several feature sets

4.2. Special Cases that Result in Misspellings

Of the cases that were missed, many resulted from a few specific phenomena. This algorithm is not currently capable of recognizing these cases, but they are few and specific enough that this type of algorithm could likely be developed to handle most of them. Table 4 shows a breakdown of misspellings under the most accurate feature set for each scale degree. Non-chord tones (NCTs), augmented sixth chords (6+), and fully-diminished seventh chords (o7) are specific, mutually exclusive cases, each with a well-defined melodic or harmonic function.

Misspellings	c♯/d♭	d♯/e♭	g♯/a♭	a♯/b♭
Total	11	27	21	11
NCT	0	10	1	11
Aug. Sixth	0	7	0	0
Fully-Dim. 7	0	10	14	0

Table 4. Total misspelled cases and the subsets thereof resulting from specific functionalities

4.2.1. *Non-Chord Tones*

Non-chord tone refers to one of a family of cases in which a note is considered to have no harmonic function (ie. it is not considered part of the concurrent chord), but has a specific melodic function and spelling. Figure 4 is an example of NCTs from Mozart K.458. NCTs can be either diatonic (falling within the local key) or chromatic (falling outside the local key). Clearly, the cases of interest here are chromatic NCTs. The spelling captured by the resolution feature (R) is consistent with the melodic function of chromatic NCTs, though it is not a perfect predictor for this data.

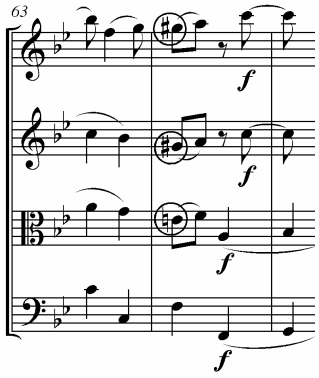

Figure 4. NCTs in Mozart, K. 458 mvt. IV Allegro Assai

4.2.2. *Augmented Sixth Chords*

There are several varieties of augmented sixth chords, but they all contain the 1st, lowered 6th and raised 4th scale degrees (c, f♯, and a♭ in C major). They get their name from the augmented sixth interval formed between the lowered 6th and raised 4th scale degrees. Augmented sixth chords tend to resolve (harmonically) to the dominant (**V**) chord with the lowered 6th and raised 4th resolving (melodically) outward by a half-step, each to the 5th as implied by the spelling, and the 1st resolving down by half-step to the 7th. In practice, however, augmented sixth commonly resolve to other dominant function chords like the cadential 6-4 chord or the dominant seventh (**V**^{7}) as in Figure 5. In these cases, one or more of the tones in the augmented sixth chord do not resolve as expected. These are examples of cases in which the determining feature for the spelling of a pitch is a resolution that is *expected*, but not achieved.

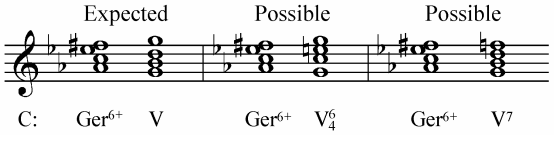

Figure 5. Common resolutions of a German augmented sixth chord

In addition to its voice-leading functionality, the augmented sixth in the spelling of an augmented sixth chord also serves to eliminate ambiguity regarding its harmonic function. Using an enharmonically equivalent spelling, an augmented sixth chord can look like a dominant seventh function chord (in a different key) as shown in Figure 6. Thus, it is very important to the harmonic legibility of a piece that augmented sixth chords are spelled correctly.

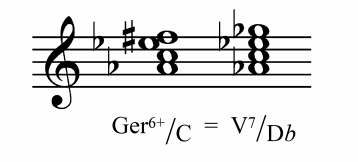

Figure 6. Enharmonically equivalent spellings

4.2.3. *Fully-Diminished Seventh Chords*

Fully-diminished seventh chords, in contrast to the vast majority of functional chords, are not always well-defined in terms of spelling. For most chords, the root structure is orientable in any inversion. For example, in a major triad, the root is the same pitch in every inversion. For fully-diminished seventh chords, on the other hand, inversions are indistinguishable from each other, and the root is not uniquely identifiable, as shown in Figure 7. Technically, a unique best spelling for a fully-diminished seventh chord can be determined by the voice-leading of the individual parts, but in practice, composers often spell fully diminished sevenths arbitrarily.

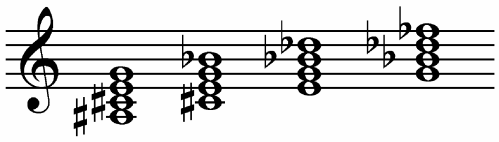

Figure 7. Enharmonically equivalent fully-diminished seventh chords

For c♯/d♭, d♯/e♭ and g♯/a♭, the best results were achieved by spelling directly according to the resolution feature (R). Thus, all of the errors made by the algorithm in these cases resulted from misleading values of the resolution feature (R). These cases are broken down in Table 5.

Misleading R	c♯/d♭	d♯/e♭	g♯/a♭	Totals
Total	11	27	21	59
Expected Res.	1	15	7	23
Octave Problem	2	1	5	8
Uncapturable	8	11	9	28

Table 5. Misspellings due to misleading R values

4.2.4. *Expected Resolution*

In some cases, the determining feature for the spelling of a pitch is its expected resolution, which may never actually be achieved. If this is the case, it is a result of the harmonic progression and not melodically motivated. Some common examples of this phenomenon involving augmented sixth chords are shown in Figure 5. Theoretically, these cases should be able to be captured by the harmonic parse.

4.2.5. *Octave Problems*

The way the R is calculated, the resolution of a pitch *p* is only captured if it occurs in the same octave as *p*. In some cases, a tendency tone is passed between different octaves before it is resolved, and the resolution only occurs once. Thus, some cases were missed because the resolution fell in the wrong octave. For example, in Figure 8, the g♯ in the Viola part in measure 46 is resolved to a♮ in the Violin II part, but the resolution feature captures the g♮ in measure 48.

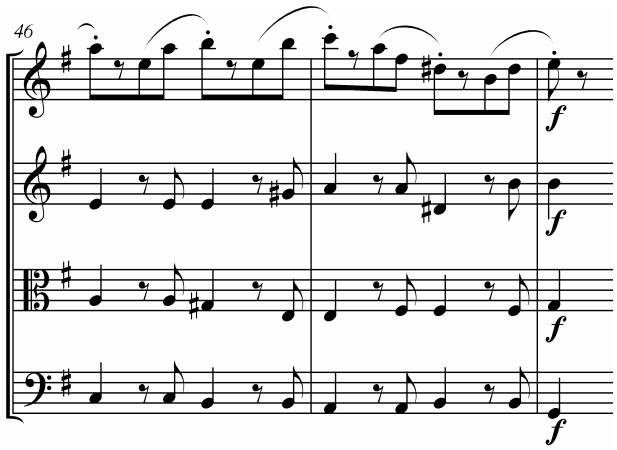

Figure 8. Excerpt from Beethoven, Op. 59 "Razumovsky" No. 2 mvt. IV Presto

4.2.6. *Uncapturable Cases*

All of the special cases mentioned above are capturable under the assumptions of this algorithm. The last category in table 5 refers to all remaining cases in which the spelling is inconsistent with the assumptions of this algorithm. This includes error caused by ambiguity of spelling in fully diminished seventh chords, among other things. Interestingly, upon comparing these cases against published scores, 26 of the 28 cases in this category were discovered to be errors in the MusicXML spelling data.

5. CONCLUSIONS

Overall, this algorithm accurately spelled 94.98% (misspelled 70 of 1395) of the cases requiring accidentals in the local key and 99.686% (misspelled 71 of 22,593) total on all notes in the test data. The results speak for a strong dependence of spelling information on voice-leading resolution, although the quality of the outcome was limited by scarce and imperfect ground-truth data. Also, the vast majority of misspellings generated here can be accounted for in terms of a few tractable cases, so the level of accuracy achieved by this type of algorithm has room to improve dramatically.

ACKNOWLEDGEMENTS

This work is supported by NSF grant ISS-0113496.

REFERENCES

[1] Breiman, L., Friedman, J., Olshen, R. and Stone, C. *Classification and Regression Trees.* Wadsworth, 1984.

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

[3] Chew, E., Chen, Y. "Determinimg Context-Defining Windows: Pitch Spelling using the Spiral Array", *Proceedings of the Fourth International Conference on Music Information Retrieval,* Baltimore, Maryland, USA, 2003.

[4] Hewlett, W. "A Base-40 Representation of Musical Pitch Notation", CCARH Publications, Stanford, CA, 1986.

[5] Meredith, D. "Pitch Spelling Algorithms", *Proceedings of the 5th Triennial ESCOM Conference,* Hannover, Germany, 2003.

[6] Raphael, C., Stoddard, J. "Harmonic Analysis with Probablilistic Graphical Models", *Proceedings of the Fourth International Conference on Music Information Retrieval,* Baltimore, Maryland, USA, 2003.