

Pitch Spelling Algorithms

David Meredith

Centre for Computational Creativity

Department of Computing

City University, London

`dave@titanmusic.com`

`www.titanmusic.com`

MaMuX Seminar

IRCAM, Centre G. Pompidou, 1, place I. Stravinsky, 75004 Paris

Saturday 13 December 2003

1. The concept of a pitch spelling algorithm



- A *pitch spelling algorithm* is an algorithm that attempts to compute the correct pitch names of the notes in a passage of tonal music, when given only the onset-time, MIDI note number and (possibly) the duration of each note.
- Practical applications of pitch spelling algorithms:
 1. Required for MIDI-to-notation transcription.
 2. Required for audio-to-notation transcription.
 3. Useful in music information retrieval and musical pattern discovery.

1. The concept of a pitch spelling algorithm

1. I'm going to talk to you for 45 minutes about the problem of constructing a reliable *pitch spelling algorithm*—that is, an algorithm that reliably computes the correct pitch names (e.g., C \sharp 4, B \flat 5 etc.) of the notes in a passage of tonal music, when given only the onset-time, MIDI note number and possibly the duration of each note in the passage.
2. There are good practical and scientific reasons for attempting to develop a reliable pitch spelling algorithm.
3. First, until such an algorithm is devised, it will be impossible to construct a reliable *MIDI-to-notation transcription algorithm*—that is, an algorithm that reliably computes a correctly notated score of a passage when given only a MIDI file of the passage as input.
4. Second, existing audio transcription systems generate not notated scores but MIDI-like representations as output.
5. So if you want to produce a notated score from a digital audio recording, you need a MIDI-to-notation transcription algorithm (incorporating a pitch spelling algorithm) in addition to an audio transcription system.
6. Third, knowing the letter-names of the pitch events in a passage can be extremely useful in music information retrieval and musical pattern discovery (Meredith, Lemström, and Wiggins, 2002).
7. In particular, the occurrence of a motive on a different degree of a scale might be perceptually significant even if the corresponding chromatic intervals in the patterns differ. In this extract here, for example,

the three patterns A, B and C are perceived as being three occurrences of the same motive even though the corresponding chromatic intervals are different in the three patterns. [PLAY EXAMPLES]

8. Note that in this example, one important aspect of the perceived similarity between patterns A, B and C is nicely represented in the notation by the fact that they all have the same scale-step interval structure $\langle -1, +1, +1 \rangle$.
9. In other words, the pitch names of the notes in this passage are chosen so that the scale-step interval structures of these three patterns are the same.
10. Such matches can be found using fast, exact-matching algorithms if the pitch names of the notes are encoded, but exact-matching algorithms cannot be used to find such matches if the pitches are represented using just MIDI note numbers.
11. If a reliable pitch spelling algorithm existed, it could be used to compute the pitch names of the notes in the tens of thousands of MIDI files of works that are freely available online, allowing these files to be searched more effectively by a music information retrieval (MIR) system.

2. Pitch spelling in common practice Western tonal music

Three chords are shown in a grand staff (treble and bass clefs) in common time (C).
Chord a: Treble clef has a D#4 and F#4 dyad; Bass clef has a C3 and E3 dyad.
Chord b: Treble clef has a Bb4 and Db5 dyad; Bass clef has a C3 and E3 dyad.
Chord c: Treble clef has a D#4 and F#4 dyad; Bass clef has a C#3 and E#3 dyad.

(Piston, 1978, p. 8)

A sequence of chords in a grand staff (treble and bass clefs) in common time (C). The bass line consists of sustained dyads: C3-E3, Bb3-Db4, and C#3-E#3. The treble line has a melodic line starting with a whole note chord (C4-E4-G4), followed by a half note chord (D#4-F#4), and then a quarter note melodic line (G4, F#4, E4, D4).
Chord 1: Treble clef has a C4, E4, G4 triad; Bass clef has a C3, E3 dyad.
Chord 2: Treble clef has a D#4, F#4 dyad; Bass clef has a Bb3, Db4 dyad.
Chord 3: Treble clef has a G4, F#4, E4, D4 quarter notes; Bass clef has a C#3, E#3 dyad.

C: I

+II2

I

(Piston, 1978, p. 390)

2. Pitch spelling in common practice Western tonal music

1. In the vast majority of cases, the correct pitch name for a note in a passage of tonal music can be determined by considering the rôles that the note plays in the harmonic, motivic and voice-leading structures of the passage.
2. For example, when played in isolation in an equal-tempered tuning system, the first soprano note in (a) here would sound the same as the first soprano note in (b).
3. However, in (a), this note is spelt as a $G\sharp_4$ because it functions as a leading note in A minor; whereas in (b), the first soprano note is spelt as an $A\flat_4$ because it functions as a submediant in C minor.
4. Similarly, the first alto note in (b) would sound the same as the first alto note in (c) in an equal-tempered tuning system.
5. However, in (b) the first alto note is spelt as an $F\flat_4$ because it functions in this context as a subdominant in C minor; whereas, in (c), the first alto note functions as a leading note in $F\sharp$ minor so it is spelt as an $E\sharp_4$.
6. So, in general, the pitch name assigned to a note in a passage of tonal music is not arbitrary. In most cases, the pitch name of each note is carefully chosen so that the resulting score represents as well as possible certain important aspects of the way that the music is intended to be perceived and interpreted.
7. This illustrates the important and more general point that a correctly notated staff notation score of a passage of Western tonal music 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. In this

respect, a correctly notated score serves a similar function to, say, a Schenkerian analysis or an analysis using Lerdahl and Jackendoff's (1983) *Generative Theory of Tonal Music*.

8. Of course, there do exist cases where the pitch name of a note cannot be uniquely determined by considering the harmonic, motivic and voice-leading structures of its context.
9. For example, as Piston (1978, p. 390) observes, the tenor $E\flat_4$ in the third and fourth bars of the bottom figure here should be spelt as a $D\sharp_4$ if one perceives the harmonic progression here to be $^+II^2 - I$ as proposed by Piston. But spelling the soprano $E\flat_5$ in the fourth bar as $D\sharp_5$ does not seem to represent the perceived structure of the melody correctly.
10. Indeed, this $E\flat_5$ seems to be functioning as the ninth of a supertonic chord whereas the tenor $E\flat_4$ can be interpreted as the raised root of a supertonic chord. If one interprets the passage in this way, then the $E\flat_4$ should be spelt as a $D\sharp_4$ and the soprano $E\flat_5$ should be left unchanged, which would lead to an $E\flat$ sounding simultaneously with a $D\sharp$ in the second half of the fourth bar!

3. Computationally modelling the cognitive processes underlying pitch spelling

- General agreement among experts on how each note should be spelt within a tonal context.
- *Question:* What are the cognitive processes involved when a musically trained individual determines the correct pitch name of a note in a passage of tonal music?
- *Answer:* Construct a reliable pitch spelling algorithm that is consistent with what is known about expert music perception and cognition.
- Evaluate algorithms by comparing output with encodings of authoritative published scores.

3. Computationally modelling the cognitive processes underlying pitch spelling

1. Fortunately, however, such cases where it is difficult to determine the correct pitch name of a note in a tonal work are relatively rare—particularly in Western tonal music of the so-called ‘common practice’ period (roughly the 18th and 19th centuries).
2. In the vast majority of cases, those who study and perform Western tonal music agree about how a note should be spelt in a given tonal context.
3. This poses an interesting problem for cognitive science, namely: what are the cognitive processes involved when a musically trained individual determines the correct pitch name of a note in a passage of tonal music?
4. A good way of trying to answer this question is to attempt to construct a reliable pitch spelling algorithm that operates in a way that is consistent with what is known about expert music perception and cognition.
5. The vast majority of notes in authoritative published editions of scores of common practice tonal works are generally agreed to be spelt correctly by those who understand Western staff notation.
6. Therefore a pitch spelling algorithm or computational model of pitch spelling can be evaluated quantitatively by running it on tonal works and comparing the pitch names it predicts with those of the corresponding notes in authoritative published editions of scores of the works.
7. In other words, such authoritative scores can provide us with a ‘ground truth’ that we can compare with the output of a pitch spelling algorithm.

8. However, this can only be done accurately and quickly if one has access to encodings of these authoritative scores in the form of computer files that can be compared automatically with the pitch spelling algorithm's output.

4. A comparison of three pitch spelling algorithms

- Three pitch spelling algorithms compared:
 - Cambouropoulos (1996, 1998, 2000, 2001, 2002)
 - Longuet-Higgins (1976, 1987, 1993)
 - Temperley (1997, 2001)
- Two test corpora used:
 - all pieces in first book of Bach's *Das Wohltemperirte Klavier* (BWV 846–869) (41544 notes)
 - 1655 movements from works by 9 baroque and classical composers (Corelli, Vivaldi, Telemann, Bach, Handel, B. Marcello, Haydn, Mozart and Beethoven) (1729886 notes)
- Corpora derived from MuseData collection of encoded scores (www.musedata.org).

4. A comparison of three pitch spelling algorithms

1. In order to get a clearer idea of the ‘state of the art’ in the field, I’ve compared the performance of three pitch spelling algorithms on two test corpora.
2. The algorithms I’ve compared are those of Cambouropoulos (1996, 1998, 2000, 2001, 2002), Longuet-Higgins (1976, 1987, 1993) and Temperley (1997, 2001).
3. As test corpora, I’ve used the first book of J. S. Bach’s *Das Wohltemperirte Klavier* (BWV 846–869), which contains 41544 notes; and a second corpus containing about 1.73 million notes and consisting of 1655 movements from works by 9 baroque and classical composers.
4. Both corpora were derived from the MuseData collection of encoded scores (www.musedata.org).

5. Longuet-Higgins's (1976, 1987, 1993) algorithm

Pitch name	...	F \flat	C \flat	G \flat	D \flat	A \flat	E \flat	B \flat	F	C	G	D	A	E	B	F \sharp	C \sharp	G \sharp	D \sharp	A \sharp	E \sharp	B \sharp	...
Sharpness	...	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	...

- Pitch spelling is one function of the *music.p* program.
- Input: list of triples, $\langle p, t_{\text{on}}, t_{\text{off}} \rangle$.
- Designed to be used only on monophonic melodies.
- Algorithm does *not* use 3d ‘tonal pitch space’ model.
- Assumes every note is no more than 6 steps from tonic on line of fifths.
- Assumes first note is tonic or dominant of opening key.
- Assumes consecutive notes always less than 12 steps apart on line of fifths.
- If two consecutive notes separated by more than 6 steps on line of fifths, then interpreted as evidence for a modulation.

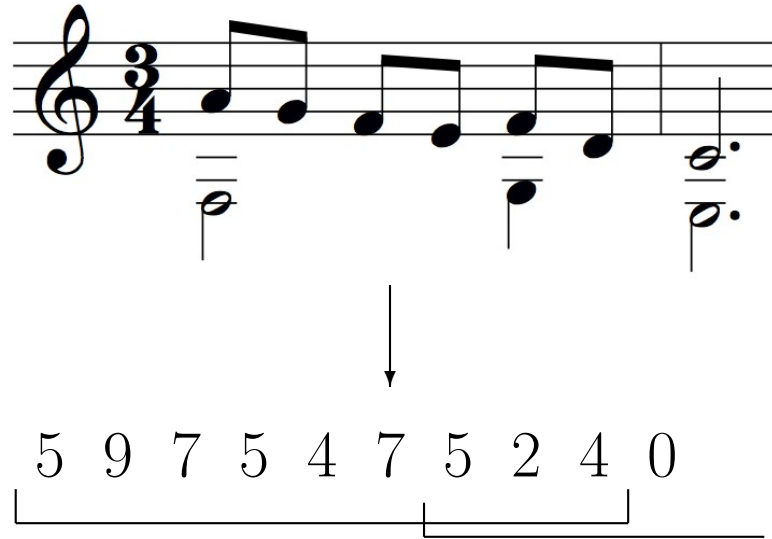
5. Longuet-Higgins's (1976, 1987, 1993) algorithm

1. Pitch spelling is one of the tasks performed by Longuet-Higgins's (1976, 1987, 1993) `music.p` program.
2. The input to `music.p` must be in the form of a list of triples, $\langle p, t_{\text{on}}, t_{\text{off}} \rangle$, each triple giving the “keyboard position” p together with the onset time t_{on} and the offset time t_{off} in centiseconds of each note.
3. The keyboard position p is just the MIDI note number minus 48. So, for example, the keyboard position of middle C is 12.
4. Longuet-Higgins (1987, p. 114) intended the `music.p` program to be used only on monophonic melodies and explicitly warns against using it on “accompanied melodies” or what he calls “covertly polyphonic” melodies (i.e., compound melodies).
5. It is perhaps also worth pointing out that the pitch spelling algorithm implemented in `music.p` does *not* use Longuet-Higgins's well-known three-dimensional ‘tonal space’ model of tonality (Longuet-Higgins, 1987, p. 110–111).
6. However, Longuet-Higgins does actually describe this multi-dimensional model in the paper where the `music.p` program is published. This has probably led some readers to assume (incorrectly) that the program implements Longuet-Higgins's three-dimensional ‘tonal space’ model.
7. In fact, the only pitch spaces used in the pitch spelling algorithm implemented in `music.p` are the circle of fifths and the ‘line of fifths’.
8. The algorithm computes a value of “sharpness” q for each note in the input (Longuet-Higgins, 1987, p. 111). The sharpness of a note is a number indicating the position of the pitch name of the note

on the line of fifths (as shown here) (Temperley, 2001, p. 117). It is therefore essentially the same as Temperley's (2001, p. 118) concept of "tonal pitch class".

9. Longuet-Higgins's algorithm tries to spell the notes so that the distance on the line of fifths between each note and the tonic at the point at which the note occurs is less than or equal to 6 (which corresponds to a tritone) (Longuet-Higgins, 1987, p. 113).
10. The algorithm assumes at the beginning of the music that the first note is either the tonic or the dominant of the opening key and chooses between these two possibilities on the basis of the interval between the first two notes (Longuet-Higgins, 1987, p. 114).
11. The algorithm also assumes that two consecutive notes in the music are never separated by more than 12 steps on the line of fifths (Longuet-Higgins, 1987, p. 111). In fact, if two consecutive notes in the music are separated by more than 6 steps on the line of fifths, then the algorithm treats this as evidence of a change of key.

6. Cambouropoulos's (1996, 1998, 2000, 2001, 2002) algorithm



The image shows a musical staff in 3/4 time with a treble clef. The melody consists of nine notes: G4, A4, B4, A4, G4, F4, E4, D4, and C4. Below the staff, a sequence of pitch classes is shown: 5, 9, 7, 5, 4, 7, 5, 2, 4, 0. A downward arrow points from the first note of the melody to the first number of the sequence. Brackets are drawn under the sequence: one bracket under the first seven numbers (5, 9, 7, 5, 4, 7, 5) and another bracket under the last three numbers (5, 2, 4, 0).

- Assumes 3 possible spellings for ‘white note’ pitch classes and 2 possible spellings for ‘black note’ pitch classes.
- Computes all spellings for each window (on average, over 5000 spellings per window for a window size of 9).
- Computes penalty score for each spelling in a window and chooses spelling with least penalty score.

6. Cambouropoulos's (1996, 1998, 2000, 2001, 2002) algorithm

1. Cambouropoulos's method involves first converting the input representation into a sequence of pitch classes in which the pitch classes are in the order in which they occur in the music (the pitch classes of notes that occur simultaneously being ordered arbitrarily).
2. Having derived an ordered set of pitch classes from the input, Cambouropoulos's algorithm then processes the music a window at a time, each window containing a fixed number of notes (between 9 and 15). Each window is positioned so that the first third of the window overlaps the last third of the previous window.
3. Cambouropoulos allows 'white note' pitch classes (i.e., 0, 2, 4, 5, 7, 9 and 11) to be spelt in three different ways (e.g., pitch class 0 can be spelt as B \sharp , C \natural or D $\flat\flat$) and 'black note' pitch classes to be spelt in two different ways (e.g., pitch class 6 can be spelt as F \sharp or G \flat).
4. Given these restricted sets of possible pitch names for each pitch class, the algorithm computes all possible spellings for each window. So, on average, if the window size is 9, there will be over 5000 possible spellings for each window.
5. A penalty score is then computed for each of these possible window spellings. The penalty score for a given window spelling is found by computing a penalty value for the interval between every pair of notes in the window and summing these penalty values.
6. A given interval in a particular window spelling is penalised more heavily if it is an interval that occurs less frequently in the major and minor scales.

7. An interval is also penalised if either of the pitch names forming the interval is a double-sharp or a double-flat.
8. For each window, the algorithm chooses the spelling that has the lowest penalty score.

7. Temperley's (1997, 2001) algorithm

- Searches for spelling that best satisfies following three preference rules:
 - TPR 1** (Pitch Variance Rule). Prefer to label nearby events so that they are close together on the line of fifths.
 - TPR 2** (Voice-Leading Rule). Given two events that are adjacent in time and a half-step apart in pitch height: if the first event is remote from the current center of gravity, it should be spelled so that it is five steps away from the second on the line of fifths.
 - TPR 3** (Harmonic Feedback Rule). Prefer TPC representations which result in good harmonic representations.
- Requires duration of each note.
- Requires tempo (i.e., uses absolute performed duration).
- Cannot deal with cases where two or more notes with the same pitch start at the same time.
- Needs to compute metrical and harmonic structure in order to compute pitch names.

7. Temperley's (1997, 2001) algorithm

1. Temperley's (1997, 2001) pitch spelling algorithm is implemented in his *harmony* program which forms one component of his and Sleator's *Melisma* system.
2. The input to the *harmony* program must be in the form of a "note-list" (Temperley, 2001, pp. 9–12) giving the MIDI note number of each note together with its onset time and duration in milliseconds.
3. Temperley's (2001, pp. 115–136) pitch spelling algorithm searches for the spelling that best satisfies three "preference rules".
 - (a) The first of these rules stipulates that the algorithm should "prefer to label nearby events so that they are close together on the line of fifths" (Temperley, 2001, p. 125). This rule bears some resemblance to the basic principle underlying Longuet-Higgins's algorithm (see above).
 - (b) The second rule expresses the principle that if two tones are separated by a semitone and the first tone is distant from the key centre, then the interval between them should preferably be spelt as a diatonic semitone rather than a chromatic one (Temperley, 2001, p. 129).
 - (c) The third preference rule steers the algorithm towards spelling the notes so that what Temperley calls a "good harmonic representation" results (Temperley, 2001, p. 131).
4. Note, however, that Temperley's algorithm requires more information in its input than the other algorithms. In particular, it needs to know the duration of each note and the tempo at each point in the passage. It also needs to perform a full analysis of the metrical and harmonic structure of the passage in order to generate a high quality result.
5. Also, it cannot deal with cases where two or more notes with the same pitch start at the same time.

8. Results of running algorithms on first book of J. S. Bach's
Das Wohltemperirte Klavier

<i>Algorithm</i>	<i>% notes correct</i>	<i>Number of errors</i>
Cambouropoulos	93.74	2599
Longuet-Higgins	99.36	265
Temperley	99.71	122

Total number of notes in corpus = 41544.

8. Results of running algorithms on first book of J. S. Bach's

Das Wohltemperirte Klavier

1. When these three algorithms were run on the first book of J. S. Bach's *Das Wohltemperirte Klavier*, the results obtained were as shown here.

9. The *ps13* algorithm



- Step 1** For each note n and each pitch class p , compute $CNT(p, n)$ which is the number of times that p occurs in a context surrounding n that includes K_{pre} notes preceding n and K_{post} notes following n .
- Step 2** For each note n and each pitch class p , compute the letter name $L(p, n) \in \{A, B, C, D, E, F, G\}$ that n would have if p were the tonic at the point where n occurs (assuming that the notes are spelt as they are in the harmonic chromatic scale on p).
- Step 3** For each note n and each letter name ℓ , compute the set of tonic pitch classes, $X(n, \ell)$, that would lead to n having the letter name ℓ .
- Step 4** For each note n and each letter name ℓ , compute the sum, $N(\ell, n)$, of the values of $CNT(p, n)$ for all the tonic pitch classes $p \in X(n, \ell)$.
- Step 5** Make the letter name of n equal to that value of ℓ for which $N(\ell, n)$ is a maximum.

In Stage 2, neighbour-note and passing-note errors corrected.

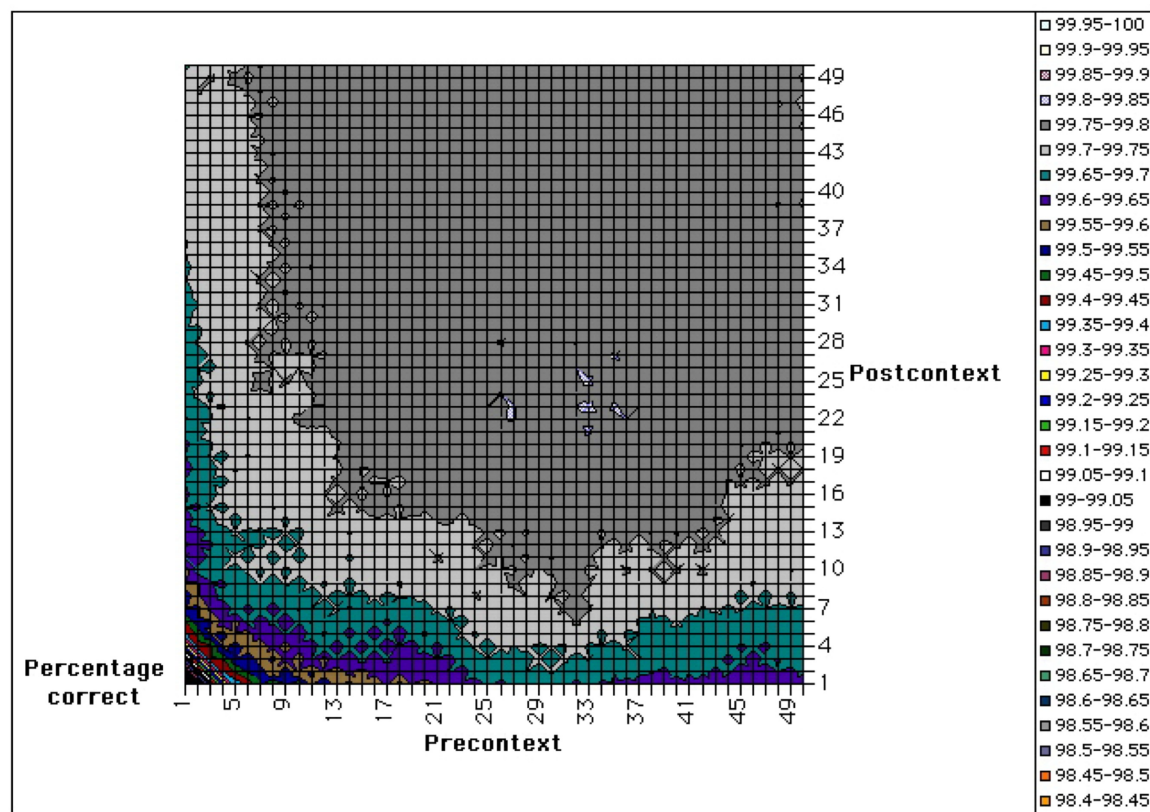
9. The *ps13* algorithm

1. Having done this comparison and gained a better idea of the ‘state of the art’ in the field, I attempted to construct a new algorithm that improved on Temperley’s.
2. I experimented with about 30 different algorithms and I’ll now briefly describe the one that performed best, an algorithm that I call *ps13*.¹
3. At the highest level of description, *ps13* can be broken down into two stages, which I’ll call Stage 1 and Stage 2.
4. Stage 1 involves carrying out the following steps: [SEE SLIDE]
5. Stage 2 of the algorithm corrects those instances in the output of Stage 1 where a neighbour note or passing note is erroneously predicted to have the same letter name as either the note preceding it or the note following it. That is, the second stage of the algorithm corrects errors like these shown on the staff at the top here.
6. In the first step of Stage 1, the algorithm essentially counts how many times each pitch class occurs within some specified context surrounding a particular note.
7. Krumhansl (1990, pp. 66–75) showed that there is a high correlation between the frequency with which a pitch class occurs within a passage and its perceived tonal stability as measured experimentally (Krumhansl and Kessler, 1982).
8. This suggests that the value $CNT(p, n)$, calculated in the first step of Stage 1, gives an approximate measure of the perceived tonal stability of the pitch class p at the point in the music where n occurs.

¹Patent pending (Meredith, 2003).

9. In *ps13* the value $CNT(p, n)$ is used as a measure of the likelihood of p being perceived to be the tonic at the point where note n occurs.
10. Note that, unlike Temperley's algorithm, *ps13* uses neither duration nor tempo and can deal with situations where two or more notes with the same pitch start at the same time.

10. Results of running *ps13* on the first book of J. S. Bach's *Das Wohltemperirte Klavier*

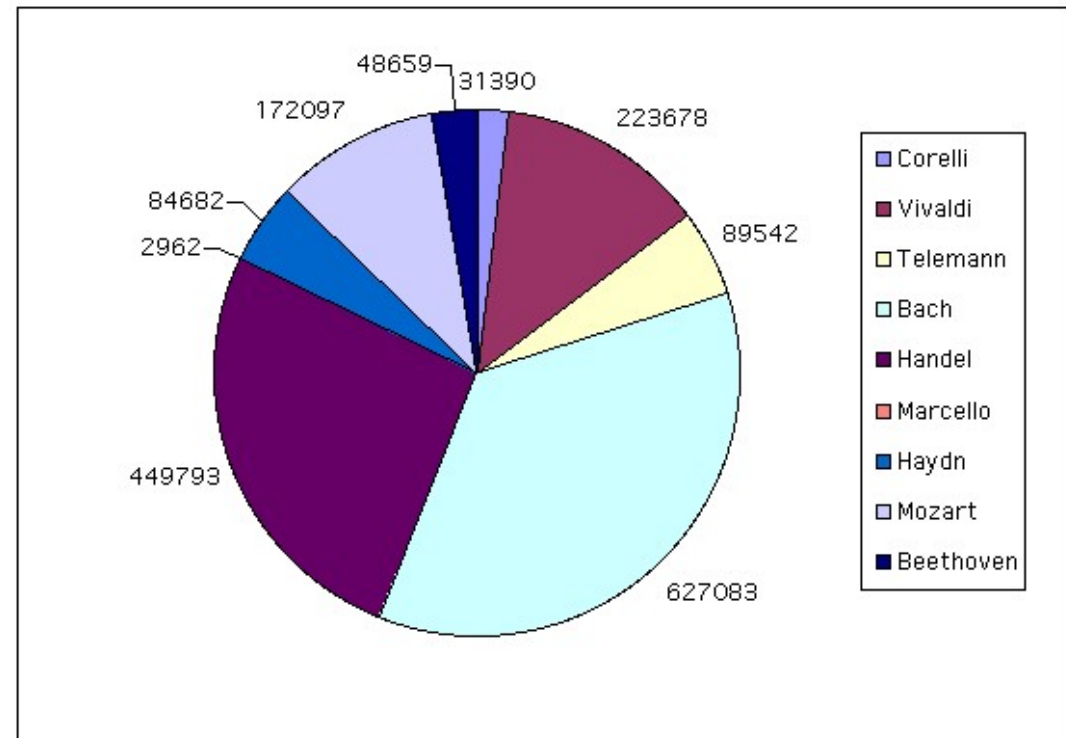
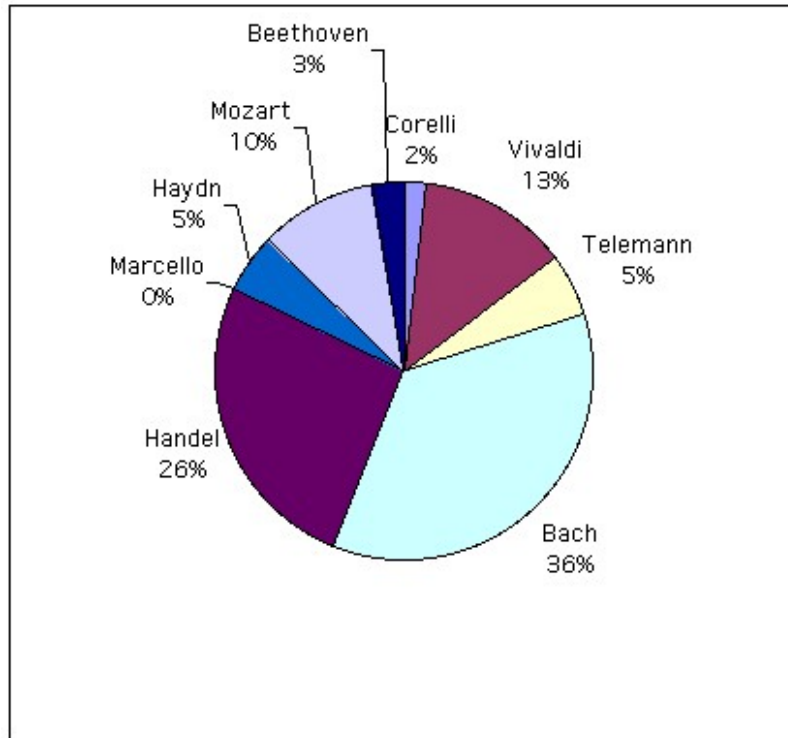


- when $K_{pre} = 33$ and $K_{post} \in \{23, 25\}$, *ps13* makes 81 mistakes on this corpus (i.e., 99.81% notes spelt correctly).

10. Results of running *ps13* on the first book of J. S. Bach's *Das Wohltemperirte Klavier*

1. In the first step of Stage 1, *ps13* counts how many times each pitch class occurs within some specified context surrounding a particular note, defined by the values of K_{pre} and K_{post} .
2. In order to explore the effect that varying the values of K_{pre} and K_{post} has on the performance of *ps13*, I ran the algorithm 2500 times on the test corpus, each time using a different pair of values $\langle K_{\text{pre}}, K_{\text{post}} \rangle$ chosen from the set $\{\langle K_{\text{pre}}, K_{\text{post}} \rangle \mid 1 \leq K_{\text{pre}} \leq 50 \wedge 1 \leq K_{\text{post}} \leq 50\}$.
3. I found that *ps13* made fewer than 122 mistakes (i.e., performed better than Temperley's algorithm) on the test corpus for 2004 of the 2500 $\langle K_{\text{pre}}, K_{\text{post}} \rangle$ pairs tested (i.e., 80.160% of the $\langle K_{\text{pre}}, K_{\text{post}} \rangle$ pairs tested).
4. *ps13* performed best on the test corpus when K_{pre} was set to 33 and K_{post} was set to either 23 or 25. With these parameter values, *ps13* made only 81 errors on the test corpus—that is, it correctly predicted the pitch names of 99.81% of the notes in the test corpus.
5. The mean number of errors made by *ps13* over all 2500 $\langle K_{\text{pre}}, K_{\text{post}} \rangle$ pairs was 109.082 (i.e., 99.74% of the notes were correctly spelt on average over all 2500 $\langle K_{\text{pre}}, K_{\text{post}} \rangle$ pairs). This average value was better than the result obtained by Temperley's algorithm for this test corpus. The standard deviation in the accuracy over all 2500 $\langle K_{\text{pre}}, K_{\text{post}} \rangle$ pairs was 0.08%.
6. The worst result was obtained when both K_{pre} and K_{post} were set to 1. In this case, *ps13* made 1117 errors (97.31% correct).
7. However, provided K_{pre} was greater than about 14 and K_{post} was greater than about 21, *ps13* predicted the correct pitch name for over 99.75% of the notes in the test corpus.

11. Structure of the larger test corpus

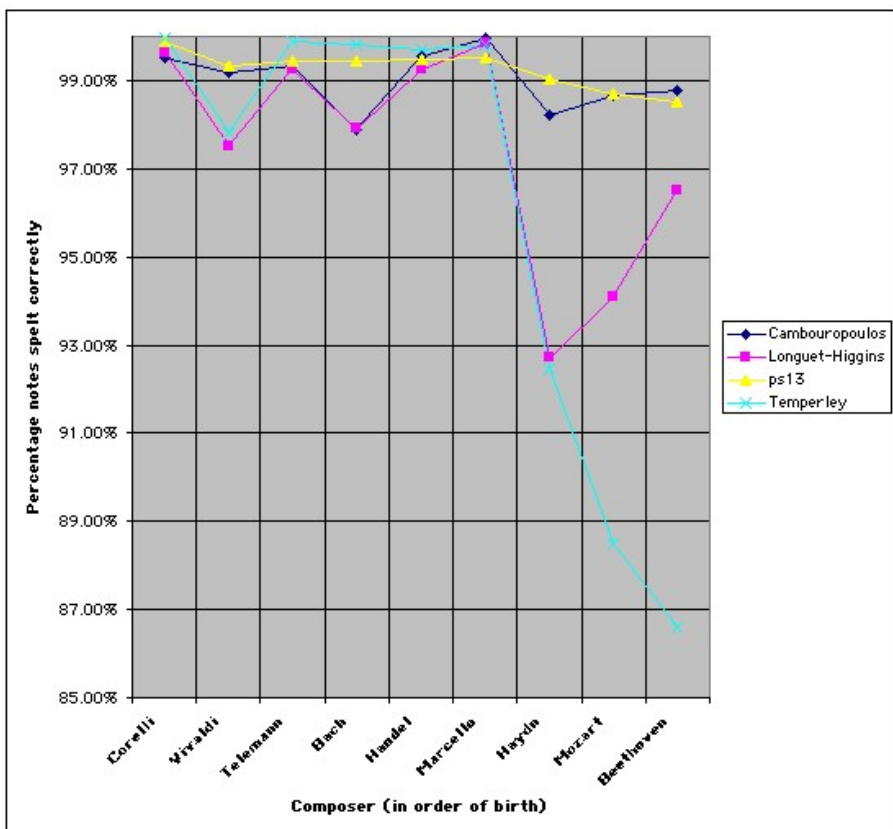


- Total number of notes in corpus = 1729886
- Total number of movements = 1655

11. Structure of the larger test corpus

1. I then ran all four algorithms on a much larger corpus containing 1729886 notes and consisting of 1655 movements from works by 9 baroque and classical composers (Corelli, Vivaldi, Telemann, Bach, Handel, B. Marcello, Haydn, Mozart and Beethoven) (1729886 notes).
2. As you can see, about 80% of the music in the corpus is baroque and the remaining 20% is classical. So the corpus is not very stylistically varied - it contains music written between about 1675 and 1825.
3. Also note that over 60% of the corpus consists of works by Bach and Handel.
4. Finally, note that the corpus is very unevenly distributed between the nine composers.

12. Comparison of algorithms with respect to number of notes spelt correctly



	Cambouropoulos	Longuet-Higgins	ps13	Temperley	Birth year
Corelli	148	120	30	6	1653
Vivaldi	1816	5518	1497	4900	1678
Telemann	604	665	481	111	1681
Bach	13347	13022	3450	1166	1685
Handel	1929	3342	2339	1309	1685
Marcello	1	4	14	5	1686
Haydn	1497	6174	823	6391	1732
Mozart	2300	10147	2250	19832	1756
Beethoven	600	1703	727	6525	1770
Complete test corpus	22242	40695	11611	40245	

	Cambouropoulos	Longuet-Higgins	ps13	Temperley	Birth year
Corelli	99.53%	99.62%	99.90%	99.98%	1653
Vivaldi	99.19%	97.53%	99.33%	97.81%	1678
Telemann	99.33%	99.26%	99.46%	99.88%	1681
Bach	97.87%	97.92%	99.45%	99.81%	1685
Handel	99.57%	99.26%	99.48%	99.71%	1685
Marcello	99.97%	99.86%	99.53%	99.83%	1686
Haydn	98.23%	92.71%	99.03%	92.45%	1732
Mozart	98.66%	94.10%	98.69%	88.48%	1756
Beethoven	98.77%	96.50%	98.51%	86.59%	1770
Complete test corpus	98.71%	97.65%	99.33%	97.67%	

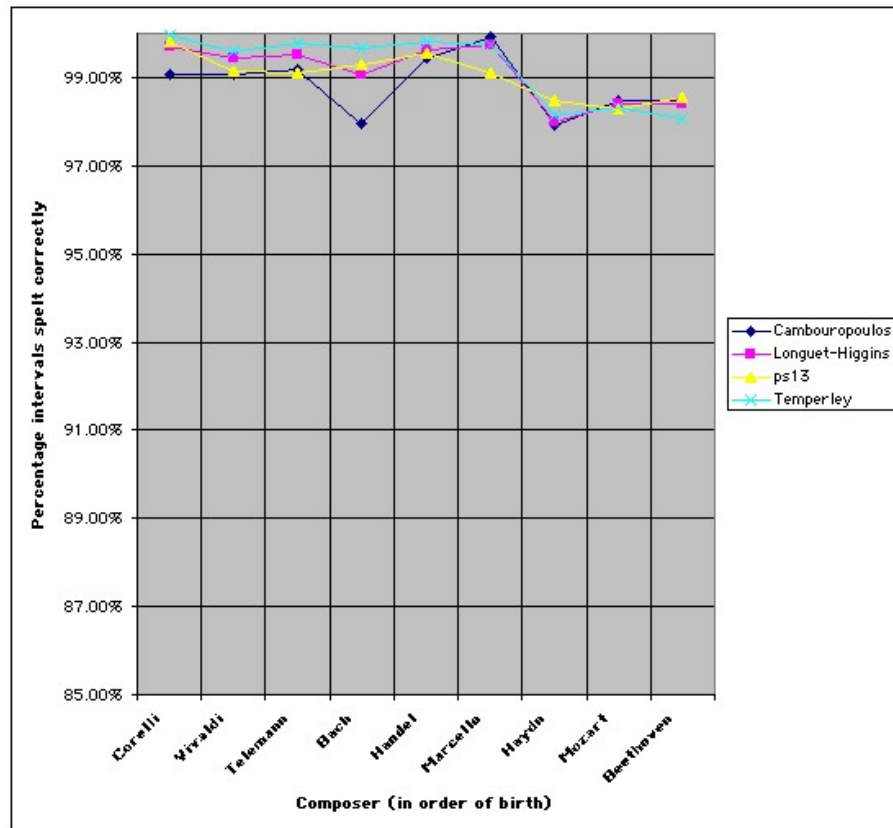
Corelli	T	0 P	0 L	0.0765 C
Vivaldi	P	0 C	0 T	0 L
Telemann	T	0 P	0 C	0.0736 L
Bach	T	0 P	0 L	0.0352 C
Handel	T	0 C	0 P	0 L
Marcello	C	0.1797 L	0.7388 T	0.0389 P
Haydn	P	0 C	0 L	0.0348 T
Mozart	P	0.3665 C	0 L	0 T
Beethoven	C	0 P	0 L	0 T
Complete test corpus	P	0 C	0 T	0.0954 L

12. Comparison of algorithms with respect to number of notes spelt correctly

1. If we consider just the number of notes in this corpus spelt correctly by the algorithms, then we get the results shown here.
2. The bottom row in the top table here gives the number of notes in the corpus spelt incorrectly by each algorithm.
3. Thus, my *ps13* algorithm spells around 12000 of the 1.7 million notes in the corpus incorrectly, while Cambouropoulos's algorithm spells over 20000 notes incorrectly and the algorithms of Longuet-Higgins and Temperley each spell over 40000 notes incorrectly.
4. The bottom row of the second table shows the total percentage of notes in the corpus spelt correctly by the four algorithms. As you can see, *ps13* correctly spells 99.33% of the notes in the corpus, Cambouropoulos's algorithm comes second with 98.71%, Temperley's comes third with 97.67% and Longuet-Higgins's comes fourth with 97.65%.
5. I then analysed the results using the McNemar test, to determine whether the differences between the scores achieved by the algorithms were statistically significant. The results of this analysis are shown in the bottom row of the bottom table.
6. An entry of 0 in this table indicates a p-value of less than 0.0001. So the bottom row of this table tells us that my *ps13* algorithm performed very significantly better than Cambouropoulos's algorithm which in turn performed very significantly better than the algorithms of Temperley and Longuet-Higgins.
7. The p-value for the difference between the scores for Temperley's and Longuet-Higgins's algorithm is greater than .05 indicating that these two scores are not significantly different from each other.

8. The graph here on the left shows the score achieved by each algorithm for each composer, the composers being placed along the horizontal axis in order of birth year.
9. This graph suggests that the algorithms of Temperley and Longuet-Higgins perform much worse on the classical composers (Haydn, Mozart and Beethoven) than they do on the baroque composers.
10. The graph also seems to show that my *ps13* algorithm seems to perform much more consistently across the different composers and styles than the other algorithms.

13. Comparison of algorithms with respect to number of intervals spelt correctly



	Cambouropoulos	Longuet-Higgins	ps13	Temperley	Birth year
Corelli	288	92	60	12	1653
Vivaldi	2064	1264	1937	941	1678
Telemann	727	450	788	204	1681
Bach	12697	5852	4507	2074	1685
Handel	2518	1703	2030	822	1685
Marcello	2	8	26	8	1686
Haydn	1750	1700	1298	1579	1732
Mozart	2596	2734	2955	2874	1756
Beethoven	736	772	695	933	1770
Complete test corpus	23378	14575	14296	9447	

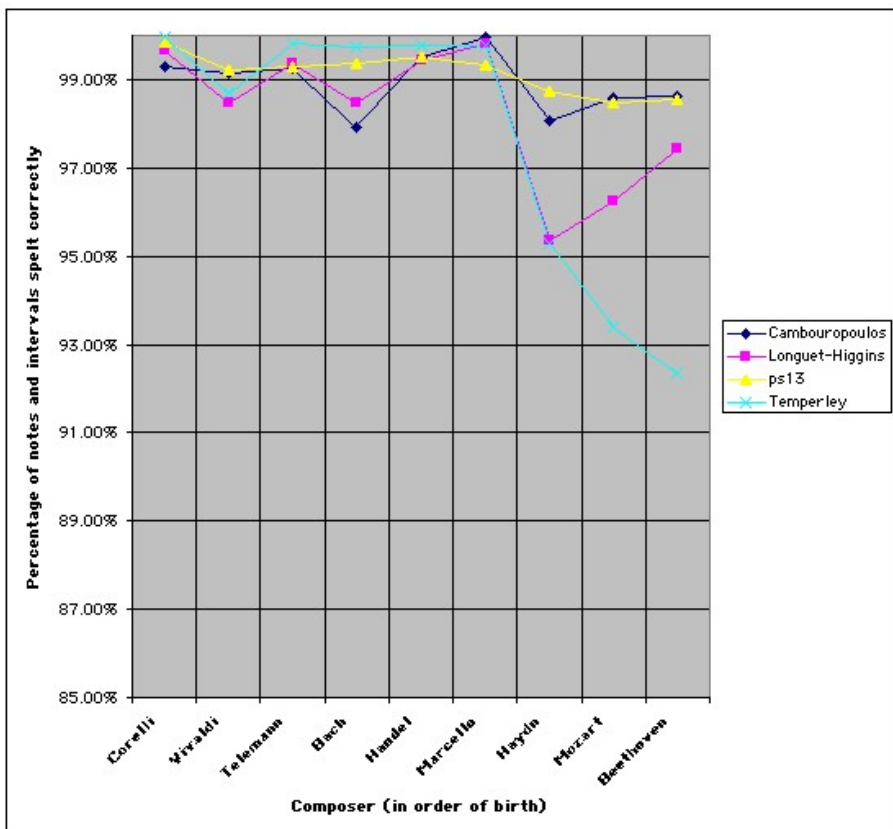
	Cambouropoulos	Longuet-Higgins	ps13	Temperley	Birth year
Corelli	99.08%	99.71%	99.81%	99.96%	1653
Vivaldi	99.08%	99.43%	99.13%	99.58%	1678
Telemann	99.19%	99.50%	99.12%	99.77%	1681
Bach	97.97%	99.07%	99.28%	99.67%	1685
Handel	99.44%	99.62%	99.55%	99.82%	1685
Marcello	99.93%	99.73%	99.12%	99.73%	1686
Haydn	97.93%	97.99%	98.47%	98.13%	1732
Mozart	98.49%	98.41%	98.28%	98.33%	1756
Beethoven	98.49%	98.41%	98.57%	98.08%	1770
Complete test corpus	98.65%	99.16%	99.17%	99.45%	

Corelli	T	0 P	0.0068 L	0 C
Vivaldi	T	0 L	0 P	0.1077 C
Telemann	T	0 L	0 C	0.0029 P
Bach	T	0 P	0 L	0 C
Handel	T	0 L	0 P	0 C
Marcello	C	0.0577, 0.0577 L,T	0.001, 0.002 P	
Haydn	P	0 T	0.0028 L	0.2416 C
Mozart	C	0.0584 L	0.0143 T	0.2459 P
Beethoven	P	0.041 C	0.0252 L	0 T
Complete test corpus	T	0 P	0.0637 L	0 C

13. Comparison of algorithms with respect to number of intervals spelt correctly

1. When I looked through the lists of errors generated by the algorithms, I noticed that, in some cases, many errors were the result of large segments of the music simply being transposed up or down a diminished second.
2. In other words, a single incorrect interval between two notes resulted in a whole segment of notes following the incorrect interval being spelt incorrectly. I therefore decided to compare the algorithms with respect to the number of *intervals* spelt correctly.
3. As you can see from the graph on the left, the algorithms of Longuet-Higgins and Temperley are considerably more successful at spelling intervals than they are at spelling notes - particularly in the music of Haydn, Mozart and Beethoven.
4. Indeed, with respect to the number of intervals spelt correctly, Temperley's algorithm performs significantly better than the other three algorithms.
5. However, as before, all the algorithms seem to perform worse on the classical music than on the baroque music.

14. Comparison of algorithms with respect to total number of intervals and notes spelt correctly



	Cambouroupoulos	Longuet-Higgins	ps13	Temperley	Birth year
Corelli	436	212	90	18	1653
Vivaldi	3880	6782	3434	5841	1678
Telemann	1331	1115	1269	315	1681
Bach	26044	18874	7957	3240	1685
Handel	4447	5045	4369	2131	1685
Marcello	3	12	40	13	1686
Haydn	3247	7874	2121	7970	1732
Mozart	4896	12881	5205	22706	1756
Beethoven	1336	2475	1422	7458	1770
Complete test corpus	45620	55270	25907	49692	

	Cambouroupoulos	Longuet-Higgins	ps13	Temperley	Birth year
Corelli	99.31%	99.67%	99.86%	99.97%	1653
Vivaldi	99.14%	98.48%	99.23%	98.70%	1678
Telemann	99.26%	99.38%	99.29%	99.83%	1681
Bach	97.92%	98.50%	99.37%	99.74%	1685
Handel	99.51%	99.44%	99.52%	99.77%	1685
Marcello	99.95%	99.80%	99.33%	99.78%	1686
Haydn	98.08%	95.35%	98.75%	95.29%	1732
Mozart	98.58%	96.26%	98.49%	93.41%	1756
Beethoven	98.63%	97.46%	98.54%	92.34%	1770
Complete test corpus	98.68%	98.41%	99.25%	98.56%	

Corelli	T	0	P	0	L	0	C
Vivaldi	P	0	C	0	T	0	L
Telemann	T	0	L	0	P	0.8047	C
Bach	T	0	P	0	L	0	C
Handel	T	0	P	0.3702	C	0	L
Marcello	C	0.0201	L	0.8414	T	0.0002	P
Haydn	P	0	C	0	L	0.4477	T
Mozart	C	0.0007	P	0	L	0	T
Beethoven	C	0.1672	P	0	L	0	T
Complete test corpus	P	0	C	0	T	0	L

14. Comparison of algorithms with respect to total number of intervals and notes spelt correctly

1. Finally, I evaluated the algorithms in terms of the total number of notes and intervals spelt correctly and the results are shown here.
2. As you can see, when evaluated in this way, *ps13* performs significantly better than the other algorithms overall. Its performance also seems to be less affected than the other algorithms by the style and period of the music.

15. Conclusions and further work

<i>Notes</i>	<i>ps13</i> (99.33%) > Camb (98.71%) > Temp (97.67%) \simeq LH (97.65%)
<i>Intervals</i>	Temp (99.45%) > <i>ps13</i> (99.17%) \simeq LH (99.16%) > Camb (98.65%)
<i>Ints and notes</i>	<i>ps13</i> (99.25%) > Camb (98.68%) > Temp (98.56%) > LH (98.41%)

- Algorithms based on circle of fifths (Temperley and Longuet-Higgins) mis-spelt many more notes in the classical music than the other algorithms.
- Need to test these and other algorithms on a stylistically more varied corpus.
- Krumhansl (1990, p. 79) claims that “once a key (or key region) has been determined, the correct spellings of the tones will be able to be determined in most cases”.
- What is the best key-finding algorithm to use for pitch spelling?
- Mistakes made by the algorithms need to be studied in detail.
- Need to determine whether or not algorithms are consistent with what is known about perception and cognition of pitch structure in tonal music.

15. Conclusions and further work

1. To sum up, four pitch spelling algorithms were run on a large corpus of baroque and classical music.
2. When the algorithms were evaluated in terms of the number of notes they spelt correctly, it was found that my *ps13* algorithm performed best, correctly spelling 99.33% of the notes in the corpus correctly.
3. While all four algorithms performed well on the baroque music, it was found that the algorithms that were based on the circle of fifths (i.e., those of Temperley and Longuet-Higgins) performed significantly less well than the other algorithms on the classical music in the corpus.
4. However, when the algorithms were evaluated in terms of the number of *intervals* spelt correctly, it was found that all the algorithms performed very well across all styles, with Temperley's algorithm performing best, correctly spelling 99.45% of the intervals in the corpus correctly.
5. When the notes and intervals were taken together, it was found that *ps13* performed best overall, correctly spelling 99.25% of the notes and intervals correctly.
6. It would be interesting to extend this study by testing these and other algorithms on other corpora containing works in a wider variety of tonal styles including, e.g., romantic, impressionist, rock and jazz.
7. Krumhansl (1990, p. 79) claims that “once a key (or key region) has been determined, the correct spellings of the tones will be able to be determined in most cases”.
8. Both Longuet-Higgins's algorithm and my *ps13* algorithm (implicitly) perform something a bit like key-finding as part of the pitch-spelling process. However, the two algorithms give quite different results

when run on the same test corpora. This demonstrates that there are various plausible ways of using the key-structure of a passage to determine pitch names and it is not at all obvious which of these methods will give the best results.

9. Krumhansl's claim needs to be tested by building complete pitch spelling algorithms based on various key-finding algorithms and comparing the performance of these algorithms with those that I've just described.
10. The mistakes made by the algorithms in these experiments also need to be analysed in more depth in order to determine if any further improvements can be made.
11. Finally, an in-depth study needs to be done to determine whether or not the algorithms are consistent with what is known about the perception and cognition of pitch structure in tonal music.

References

- Cambouropoulos, E. (1996). A general pitch interval representation: Theory and applications. *Journal of New Music Research*, **25**, 231–251.
- Cambouropoulos, E. (1998). *Towards a General Computational Theory of Musical Structure*. Ph.D. thesis, University of Edinburgh.
- Cambouropoulos, E. (2000). From MIDI to traditional musical notation. In *Proceedings of the AAAI 2000 Workshop on Artificial Intelligence and Music, 17th National Conference on Artificial Intelligence (AAAI'2000), 30 July–3 August, Austin, TX*. Available online at <ftp://ftp.ai.univie.ac.at/papers/oefai-tr-2000-15.pdf>.
- Cambouropoulos, E. (2001). Automatic pitch spelling: From numbers to sharps and flats. In *VIII Brazilian Symposium on Computer Music (SBC&M 2001)*, Fortaleza, Brazil. Available online at <ftp://ftp.ai.univie.ac.at/papers/oefai-tr-2001-12.pdf>.
- Cambouropoulos, E. (2002). Pitch spelling: A computational model. *Music Perception*. To appear.
- Krumhansl, C. L. and Kessler, E. J. (1982). Tracing the dynamic changes in perceived tonal organisation in a spatial representation of musical keys. *Psychological Review*, **89**, 334–368.
- Krumhansl, C. L. (1990). *Cognitive Foundations of Musical Pitch*, volume 17 of *Oxford Psychology Series*. Oxford University Press, New York and Oxford.
- Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press, Cambridge, MA.
- Longuet-Higgins, H. C. (1976). The perception of melodies. *Nature*, **263**(5579), 646–653.

- Longuet-Higgins, H. C. (1987). The perception of melodies. In H. C. Longuet-Higgins, editor, *Mental Processes: Studies in Cognitive Science*, pages 105–129. British Psychological Society/MIT Press, London, England and Cambridge, Mass.
- Longuet-Higgins, H. C. (1993). The perception of melodies. In S. M. Schwanauer and D. A. Levitt, editors, *Machine Models of Music*, pages 471–495. M.I.T. Press, Cambridge, Mass.
- Meredith, D., Lemström, K., and Wiggins, G. A. (2002). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, **31**(4), 321–345. Draft available online at <http://www.titanmusic.com/papers/public/siajnmr.submit.2.pdf>.
- Meredith, D. (2003). Method of computing the pitch names of notes in MIDI-like music representations. Patent filing submitted to UK Patent Office on 11 April 2003. Application number 0308456.3. Draft available online at <http://www.titanmusic.com/papers.html>.
- Piston, W. (1978). *Harmony*. Victor Gollancz Ltd., London. Revised and expanded by Mark DeVoto.
- Temperley, D. (1997). An algorithm for harmonic analysis. *Music Perception*, **15**(1), 31–68.
- Temperley, D. (2001). *The Cognition of Basic Musical Structures*. MIT Press, Cambridge, MA.