Determining Context-Defining Windows: Pitch Spelling using the Spiral Array

Elaine Chew* **and Yun-Ching Chen**
Integrated Media Systems Center and D. J. Epstein Dep of Industrial & Systems Engineering
University of Southern California, Los Angeles, California, USA.
[echew,yunchinc]@usc.edu

Abstract

This paper presents algorithms for pitch spelling using the Spiral Array model. Accurate pitch spelling, assigning contextually consistent letter names to pitch numbers (for example, MIDI), is a critical component of music transcription and analysis systems. The local context is found to be more important than the global, but a combination of both achieves the best results.

Keywords: pitch spelling, music analysis, algorithm design.

1 Pitch Spelling

Pitch spelling is a critical first step in any content-based music processing system. This paper presents three real-time pitch spelling algorithms based on context-defining windows. The algorithms summarize music information in windows of varying sizes to determining local and long-term tonal contexts using the *Spiral Array* model (Chew, 2000). The Spiral Array is a geometric model for tonality that clusters closely-related pitches and summarizes note content as spatial points in the interior of the structure. These interior points, called *centers of effect*, serve as proxies for the key context in pitch spelling. The appropriate letter name is assigned to each pitch through a nearest neighbor search in the Spiral Array space.

The problem of pitch spelling is an artefact of equal temperament tuning in western tonal music – several pitches are approximated by the same frequency (these pitches are said to be enharmonically equivalent). In a MIDI file, enharmonically equivalent pitches are represented by the same numerical value indicating its frequency and not its letter name. The name of the pitch determines the notation and serves as a clue to the key context. The local key context determines the pitch spelling in a musical excerpt. The problem of pitch spelling is to assign appropriate pitch names that are consistent with the key context.

Funded in part by the Integrated Media Systems Center, an NSF ERC, Cooperative Agreement No. EEC-9529152. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

2 The Algorithms

Each algorithm in this section generates pitch spelling assignments based on tonal contexts of varying ranges. The notes in each window generate a center of effect (CE) in the Spiral Array. Suppose the music is divided into chunks. The CE for chunks a through b is given by $\mathbf{c}_{a,b} = \sum_{i=a}^{b} \sum_j p_{ij}.d_{ij}/D_{a,b}$ where (p_{ij}, d_{ij}) are the pitch and duration of the j-th note in the i-th chunk and $D_{a,b} = \sum_{i=a}^{b} \sum_j d_{ij}$.

We use the evolving CE as a proxy for the key context. A cumulative window generates a CE that represents the long-term tonal context. A sliding window CE represents the local tonal context. For the ij-th note, suppose that $index_{ij}$ is the spelling whose index in the Spiral Array is closest to 0. This choice will bias the notation towards fewer sharps ($\sharp$)and flats ($\flat$). The most probable pitch name assignments are then given by the triplet: $I_{ij} = \{index_{ij} - 12, index_{ij}, index_{ij} + 12\}$. Of the three plausible indices, we choose the one corresponding to a pitch position that is closest to the CE.

2.1 Algorithm 1: Cumulative CE

Our original algorithm (Chew & Chen, 2003) used a cumulative window to summarize the tonal context. The algorithm advanced one chunk at a time. At time t, we examine and assign pitch names to the notes in the t-th chunk. The notes in the previous $t-1$ chunks are used to generate a CE: $\hat{\mathbf{c}}_t = \mathbf{c}_{0,t-1}$. This method had an error rate of 1/1375 (that is to say, 99.93% correct) for Beethoven's Piano Sonata Op.79 Mvt.3 and 73/1516 (95.18% correct) in the more complex Sonata Op.109 Mvt.1.

2.2 Algorithm 2: Sliding Window

We propose a sliding window algorithm that is more sensitive to changes in local tonal contexts. Figure 1 shows the sliding window method for a window of size 4. For a window of size w, the CE is defined as: $\hat{\mathbf{c}}_t = \mathbf{c}_{t-w,t-1}$. The sliding window method eliminated several of the spelling errors in the original algorithm. However, it was not able to detect quickly enough the sudden changes to distant keys in the exposition and the recapitulation of the Beethoven Op.109 Mvt.1.

2.3 Algorithm 3: Two-phase Assignment Method

We propose another sliding window method where the algorithm is allowed to re-visit previous decisions in a second win-

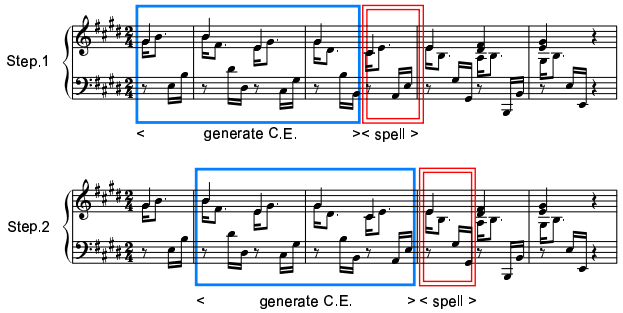

Figure 1: Sliding Window ($w = 4$).

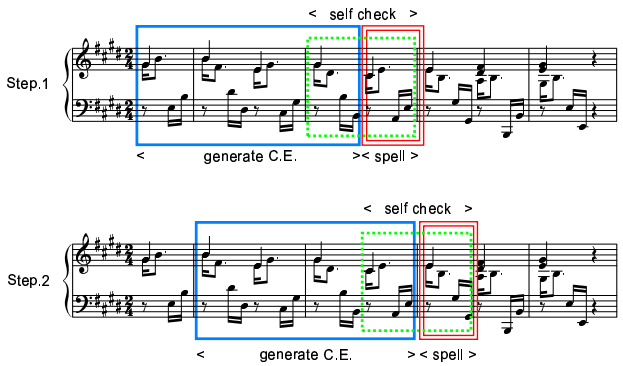

Figure 2: Two-phase assignment method ($w = 4, w_r = 2$).

dow. The additional step ensures local consistency in spelling and heightens the sensitivity to abrupt key changes. Figure 2 depicts this new procedure. After assigning pitch names as before, the two-phase assignment method re-visits the notes in a second window (denoted by dotted lines) that includes the current chunk. Both windows advance simultaneously over time.

Phase 1: The definition of the CE and the first pass at pitch name assignments proceed as in the previous algorithms.

Phase 2: Let the second local window be of size w_r, call this the self-referential window. A second CE is generated: $\tilde{\mathbf{c}}_t = f \cdot \mathbf{c}_{t-w_r,t} + (1-f) \cdot \mathbf{c}_{1,t}$, where f is the weight of the local context relative to the global tonal context. Spelling assignments in the dotted box are re-visited and re-assigned when necessary to make them consistent with this second CE.

2.4 Other Algorithms

Prior to the two-phase method described in Section 2.3, we tried a two-phase method without the cumulative CE. This algorithm did not take into account the global key context. The result was that an erroneous spelling upon a return to the original tonal context can tip the spelling from one with mostly sharps to one with many flats. The result may be locally consistent, but globally disastrous. In this case, a small perturbation can set a pathological course for error-filled spelling.

We also tested a dynamic window algorithm inspired by the hypothesis that a growing distance from the closest key portends a key change and decreasing distance indicates stability. The algorithm can be summarized as follows: when the distance from the CE to the closest key exceeds fd (some fraction f of the minimum distance between any two keys d) by a given threshold, $w = \frac{1}{2}w$; when it is less than fd by the same amount, $w = 2w$. Because the Spiral Array is configured in a 3D space, there is little correlation between the second order effect of Euclidean distance and changing keys. This algorithm was no more effective than the sliding window algorithm.

3 Results and Conclusions

The algorithms were tested on the first movement of Beethoven's *Piano Sonata No.30 in E Major*, Op.109. This late Beethoven sonata contains many sudden key changes and poses a challenge to pitch spelling algorithms. The computational results are shown in Figure 3. The sliding window algorithm consistently produces better results than the cumulative CE method. The best results are shown for window size 4, with 31 errors out of 1516. The two-phase assignment method outperforms the sliding window algorithm. For window size 4, using $w_r = 3$ and $f = 0.8$ or 0.7, we get only 27 errors. The same result is reported for window size 8, using $w_r = 6$ and $f = 0.9$. For window size 16, the best results (30 errors) are seen when $w_r = 6$ and $f = 0.8$.

Method	Parameters			Errors (1516 notes)	% Correct
Cumulative				73	95.18
Sliding Window (w)	4			31	98.00
	8			47	96.90
	16			40	97.36
Two-Phase (w, w_r, f)	4	2	0.6	28	98.15
	4	3	0.8	**27**	98.22
	4	3	0.7	**27**	98.22
	8	2	0.9	31	97.96
	8	4	0.9	40	97.36
	8	6	0.9	**27**	98.22
	16	4	0.8	40	97.36
	16	6	0.8	**30**	98.02
	16	8	0.9	37	97.56

Figure 3: Computational Results.

We conclude that the local context is more important than the global context in pitch spelling. However, we achieve the lowest error rates when we combine both short-term and long-term tonal contexts. Hence, pitch spelling is a function of both local and global tonal contexts.

Future work includes developing a database suitable for pitch spelling benchmark purposes and testing the algorithms against those by Cambouropoulos (2001) and Meredith (2003).

References

E. Cambouropoulos. Automatic pitch spelling: From numbers to sharps and flats. In *Proceedings of the VIII Brazilian symposium on Computer Music*, Fortaleza, Brazil, 2001.

E. Chew. *Towards a Mathematical Model of Tonality*. PhD thesis, MIT, Cambridge, MA, 2000.

E. Chew. Modeling tonality: Applications to music cognition. In *The 23rd Annual Meeting of the Cognitive Science Society*, Edinburgh, Scotland, August 2001.

E. Chew and Y.-C. Chen. Mapping MIDI to the Spiral Array: Disambiguating Pitch Spellings. In *The 8th INFORMS Computer Society Conference (ICS)*, Chandler, AZ, 2003.

D. Meredith. Pitch spelling algorithms. In *Proceedings of the 5th Triennial ESCOM Conference*, Hanover, Germany, 2003.