In *Proceedings of the Sixth International Conference on Music Information Retrieval,*
Joshua D. Reiss and Geraint A. Wiggins, editors, pages 552-557, London, September 2005.
Queen Mary, University of London.

VOISE: LEARNING TO SEGREGATE VOICES IN EXPLICIT AND IMPLICIT POLYPHONY

Phillip B. Kirlin and **Paul E. Utgoff**
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
{pkirlin,utgoff}@cs.umass.edu

ABSTRACT

Finding multiple occurrences of themes and patterns in music can be hampered due to polyphonic textures. This is caused by the complexity of music that weaves multiple independent lines of music together. We present and demonstrate a system, VoiSe, that is capable of isolating individual voices in both explicit and implicit polyphonic music. VoiSe is designed to work on a symbolic representation of a music score, and consists of two components: a same-voice predicate implemented as a learned decision tree, and a hard-coded voice numbering algorithm.

Keywords: voice segregation, explicit polyphony, implicit polyphony, machine learning, theme finding.

1 INTRODUCTION

Musicians, musicologists, music theoreticians, and music librarians (broadly construed) continue to demonstrate strong interest in automated procedures for analyzing and organizing music in its various forms. For organization of music pieces by content, some amount of analysis is required in order to provide a basis for comparisons. Music pieces can be clustered, distinguished, or indexed by one or more criteria of interest. One such criterion is the degree of similarity of themes or motives.

2 THEME FINDING

We consider a theme to be a pattern of notes that is prominent to a human listener, and that is repeated often enough to become recognizable, and even anticipated. A theme may be altered or systematically varied in its repetitions, typically without sacrificing its recognizability. An experienced listener will hear the statements of the theme (or themes) relatively easily, and a trained musician will be able to isolate them in a printed score. With themes in ear, one gains insight into the music at hand. This can be useful for comparing a given theme against a corpus of known themes, identifying a piece or its quotation, organizing musical works by themes, identifying the music genre, cataloging techniques of variation, or any number of other purposes. A general technique for automatically identifying themes would be of use, and several approaches have been devised (Smith and Medina, 2001; Meek and Birmingham, 2001; Lartillot, 2003).

3 VOICE SEGREGATION

An important aspect of analyzing music for theme identification is to segregate (separate) the stream of notes of the music piece into distinct voices. The term *voice* is generic for a singer or instrumentalist who sounds a single note at a time. It is relatively uncommon for a statement of a theme to be distributed in segments across multiple voices.

The voice segregation problem is to partition the set of notes comprising a piece of music into as many blocks as there are voices. All of the notes within a block of the partition belong to the one corresponding voice, and no notes that belong to this voice will be found in any other block.

It is important to distinguish voice segregation for theme finding and analysis from the similar problem that arises in automated music transcription. Rather than focus on the aesthetic problem of laying out voices on staves (Kilian and Hoos, 2002; Cambouropoulos, 2000), we concentrate on the music-theoretic issue of separating distinct lines of music, without regard to visual properties.

Our focus is on how to automate the process of voice segregation. We assume that segregation occurs prior to analysis of monophonic strings of notes to find themes, but of course having identified a theme can serve to help the segregation process. Indeed, human analysts appear to be somewhat opportunistic, identifying fragments of structure in the set of notes of the music piece, and using such fragments to guide subsequent analysis. We approach voice segregation more simply, assuming that it can be done well even when disregarding information from other avenues of analysis.

There are at least two broad categories of polyphony for voice segregation. In the first case, *explicit polyphony*, there are multiple notes sounding at one or more instants. Under the assumption that a single voice cannot sound two

Figure 1: Measures 1–4, Showing Explicit Polyphony

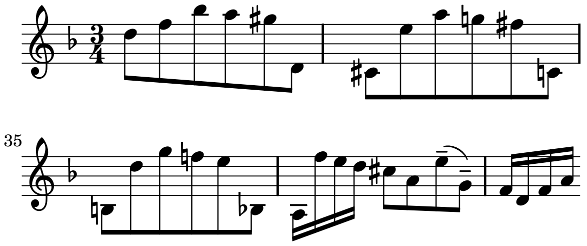

Figure 2: Measures 33–36, Showing Implicit Polyphony

notes simultaneously, one can infer safely that two explicit voices are at work. In the second case, *implicit polyphony*, there is at most one note sounding at any instant. Nevertheless, through leaps to different registers, arpeggiation, and other artistic devices, the illusion of multiple voices can be achieved. Proper analysis will attribute these notes to different implicit voices, even though the notes do not overlap in time. We address both explicit and implicit polyphony.

Consider J.S. Bach's Ciaccona for unaccompanied violin from his D Minor Partita. A ciaccona is built atop a theme in the bass that is repeated and highly varied. The label *ciaccona* can be of tremendous aid to the listener because it foretells the explicit voice in which the theme will usually appear, that the theme will be repeated relentlessly, and that one can expect myriad variations and elaborations.

Observe the first four measures, as shown in Figure 1. The four voices are readily apparent, with the soprano entering on the downbeat of the first full measure. Now compare measures 33–36, shown in Figure 2. Here, the theme is heard in the bass, which occupies its own register in the implicit polyphony. The theme has been varied so that it forms a descending chromatic line. The first measure of a third example, measure 89, appears in Figure 3. Starting at this point, Bach writes a progression of chords along with the annotation to arpeggiate. The composer shows that he conceives explicit polyphony, and that he calls on the performer to render the notes in implicit polyphony to produce the desired aural illusion.

It is important to distinguish between representations used in the task of voice segregation. There has been work on isolating lines of music in low-level audio streams, such as Baumann (2001), and also studies using higher level symbolic representations, including those by Huron (1991), Gjerdingen (1994), and Temperley (2001). However, we have not located any that have harnessed automated learning techniques.

4 LEARNING TO SEGREGATE VOICES

The VoiSe system is capable of taking as input a representation of a music score and segregating the notes of the music into distinct voices, representing separate musical lines in the piece. VoiSe includes two components: a

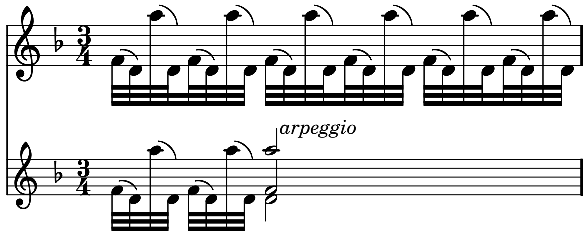

Figure 3: Measure 89, Showing Implicit Polyphony

predicate that determines whether or not two notes are in the same voice, and a separate algorithm that performs the voice segregation.

4.1 The Same-Voice Predicate

The $\mathsf{same\text{-}voice}(\mathsf{note}_1, \mathsf{note}_2)$ predicate examines various features of an ordered pair of notes to determine whether or not these two notes belong to the same voice. We restrict our predicate to note pairs satisfying two conditions. First, the notes must be distinct, and second, the onset of the first note must occur at or before the onset of the second. While there is no guarantee that the resulting predicate will be transitive, a "perfect" predicate would be.

Humans use a variety of techniques in segregating voices. A significant feature in voice segregation is the "vertical" (pitch) distance between two notes; this idea is supported by the Pitch Proximity Principle as described by Huron (2001). We include five features that calculate various measurements of this distance. These are:

- **HalfSteps**, the number of half steps separating two pitches;
- **DiscreteInterval**, a representation of the interval size, such as "M3" for a major third or "P4" for a perfect fourth;
- **IntervalCompoundSize**, a purely numeric measurement of the diatonic interval, such as 3 or 4;
- **IntervalBaseSize**, a measurement similar to **IntervalCompoundSize** but one that disregards intervening octaves between pitches; and
- **IntervalUpOrDown**, a boolean feature that reports whether the interval between two ordered notes is ascending or descending.

Rhythm also plays a large part in voice segregation. Frequently a passage will contain a repeated rhythm where notes that fall on specific beats are always in the same voice, while notes that fall on other beats are always in a different voice. VoiSe considers ten rhythmic features:

- **BeatDifPart**, the total number of beats separating two notes;
- **BeatDifMeasure**, the number of beats separating two notes, relative to a single measure;
- **BeatDifMeasureAbs**, the absolute value of **BeatDifMeasure**;
- **BeatDifMeasMod**, the (fractional) number of beats separating two notes, relative to a single beat;
- **Overlapping**, whether two notes overlap in time at all; that is, whether the second note starts before the first one ends;

- **InDifferentMeasures**, which reports whether two notes are in different measures;
- **Note1MeasBeat** and **Note2MeasBeat**, which give the numeric beat on which each note falls within a measure;
- **Note1MeasBeatMod** and **Note2MeasBeatMod**, which are similar to their non-mod counterparts, but calculate beat placement relative to the single beat.

The issue of beat differences "relative" to a measure or beat warrants some explanation. While the total number of beats separating two notes is certainly a useful feature, we can also calculate this distance by leaving out multiples of a certain number of beats. For example, if we disregard the measure in which each note occurs, we obtain a notion of beat distance calculated as though the two notes appeared in the same measure. For example, in common time, a note that falls on beat one of measure 16 and a note that falls on beat three of measure 17 have an absolute beat distance between them of six. However, their beat distance relative to one measure is only two, because if the two measures were "overlayed" one upon the other, there would be only two beats separating the notes in question.

Taking this idea one step further, the **BeatDifMeasMod** feature listed above is calculated relative to a single beat. For example, in common time, a note that falls on beat one of measure 18 and a note that falls on beat 1.5 (one eighth note from the start of the measure) of measure 19 have an absolute beat distance of 4.5, however their beat distance relative to whole beats is only 0.5.

Often when comparing two notes to determine whether they reside in the same voice or not, the task becomes more challenging as the distance between the notes increases. This is especially true in free textures where voices may appear and vanish throughout the piece. Due to this inherent ambiguity, we impose a left-aligned window that restricts the predicate to train only on note pairs where the notes in the pair are separated by less than a certain musical "distance."

4.2 The Voice-Numbering Algorithm

This same-voice predicate is not of much use on its own. We can reach our ultimate goal of voice segregation by producing a voice numbering: a mapping from notes to unique identifiers, such as integers. A perfect voice numbering indicates that notes mapped to the same integer are in the same voice, and conversely, that notes mapped to different integers are in different voices. We can include the previously learned predicate in a voice numbering algorithm, which operates under restrictions that note pairs that are determined to be in different voices be mapped to distinct integers, and conversely, that note pairs that are found to be in the same voice be mapped to the same integer. Without a perfect predicate, however, it is possible that these requirements will not be completely met.

In restricted musical textures, such as chorales or other vocal music, there are frequently a maximum of four explicit voices throughout the entire piece. In our algorithm, assign-voice-numbers, shown in Figure 4, we do not place a limit on the number of voices that may occur throughout the course of a piece. Our reasoning is that when the restrictions of human singing and vocal ranges are removed, a wider variety of textures is possible. For example, voices may seem to appear in the middle of a piece and then vanish, only to reappear later. If this occurs often, it can be difficult to determine whether two voices separated by a sizable break in time actually belong to the same voice.

```
function assign-voice-numbers(P, W, E)
  { P is the learned same-voice predicate, W is the window used to train P, E is the music excerpt whose
    voices we are numbering, f is the mapping we are defining from notes to integers. }
  nextvoice ← 0
  for each note n ∈ E, in order of note onset, do
    if f(n) is undefined then
      define f(n) := nextvoice
      nextvoice ← nextvoice +1
    else
      look ahead, in order of note onset, for the next note
        m ∈ E that satisfies P(n,m) and (n,m) ∈ W.
      if multiple notes are found whose onsets coincide
        then
        choose m to be the note that is the smallest number of half-steps away from n.
      define f(m) := f(n).
  return f
```

Figure 4: The Voice Numbering Algorithm

5 EXPERIMENTS

In the following four experiments, we trained the same-voice predicate on a musical excerpt, then ran the assign-voice-numbers algorithm with the newly-learned predicate on a different excerpt. In each of the first three experiments, the training and testing excerpts were of similar texture and style. For the fourth, both the training and testing selections were formed as the union of the respective excerpts from the first three experiments.

5.1 Data Gathering and Predicate Training

To obtain training data for the same-voice predicate, we employed the Ciaccona by J.S. Bach. We excerpted a number of passages of various lengths, textures, and styles, and produced a voice numbering by hand which we used as ground truth.

We used two types of window to train the predicate: a window based on a maximum number of beats, and a window based on a maximum number of notes. A window based on a number of beats allows the predicate to train only on note pairs for which the end of the first note is no more than a certain number of beats away from the onset of the second. This allows, for example, training on all note pairs separated by a quarter- or half note-sized distance.

In contrast, using a window based on the number of intervening notes restricts the learning algorithm to seeing instances where the number of notes between the pair of notes in question is no more than a fixed integer k. This type of window allows, for example, training on only adjacent pairs of notes, while ignoring the number of beats that occur between them.

After a set of training instances was generated from each given music selection, we used the ITI Algorithm (Utgoff et al., 1997) to learn the same-voice predicate as a decision tree. One predicated was learned per window type and size used.

5.2 Evaluation Functions

We evaluate the same-voice predicate using two distinct methods. First, we calculate the accuracy of the learned predicate through ten-fold cross validation. However, because we would like to run the assign-voice-numbers algorithm on completely separate data, we cannot be sure that the distribution of instances obtained from the new data will match the distribution from the training data. Therefore, when we test the learned predicate on a completely new music excerpt, we present these new accuracies as well. Each predicate is tested only on note pairs that satisfy the same windowing conditions on which the predicate was trained.

Evaluation of the assign-voice-numbers algorithm is less straightforward. There is no immediately apparent procedure to evaluate a mapping of notes to integers, as the integers themselves are meaningless; it is the relationships among the notes that are mapped to same or different integers that must be evaluated. Furthermore, a naïve evaluation method may severely penalize a single mistake in a numbered voice. For example, if the assign-voice-numbers algorithm predicts a set of notes $S = \{n_1, n_2, \ldots, n_k\}$ to be all in the same voice, then it is tempting to compare all of the notes in S pairwise to determine whether each note is actually in the same ground-truth voice as each other note. However, consider what happens when the first half of the notes in S are truly in voice A, and the second half are truly in a different voice B. With this first attempt at evaluating this predicted voice, only 50% of the pairwise comparisons would be true, when in fact there would really only be one mistake made in the voice numbering. If we just "break" the series into two voices at the halfway point, then suddenly the evaluation jumps back up to 100%. Clearly we must construct our evaluation function more carefully.

Therefore, we provide two measurements of accuracy for the assign-voice-numbers algorithm: *soundness* and *completeness*. These measures are similar to *fragment consistency* and *voice consistency* as defined in Chew and Wu (2004), and also to the various measures of *purity* in clustering algorithms.

To describe these quantities, let us assume that we train the same-voice predicate P on musical excerpt E_1 and apply P to number the voices in excerpt E_2. The assign-voice-numbers algorithm produces a mapping, f, from notes in E_2 to integers, representing the predicted voice numbering. Because we also have ground-truth data for E_2, we have another mapping, g, from notes in E_2 to integers, that represents the true voice numbering.

Soundness is designed to give an indication of whether or not note pairs that the assign-voice-numbers algorithm placed in the same voice are, in reality, in the same voice. Soundness is calculated by finding, for each predicted voice V, the percentage of adjacent notes $n_1, n_2 \in V$ that also satisfy $g(n_1) = g(n_2)$. In other words, we calculate the fraction of adjacent note pairs that were predicted correct that are truly correct.

Completeness, on the other hand, gives an indication of how well the assign-voice-numbers algorithm placed notes together that should belong in the same voice. This value is calculated by finding, for each ground-truth voice V, the percentage of adjacent notes $n_1, n_2 \in V$ that also satisfy $f(n_1) = f(n_2)$. We calculate the fraction of adjacent note pairs that should have been placed in the same voice that actually were.

Table 1: Accuracy of Learned Predicate for Experiment 1

Window type and size	C-V Accuracy	New Data Accuracy
beats, eighth note	78.33 ± 13.72	91.61
beats, quarter note	79.38 ± 7.25	91.50
beats, half note	74.09 ± 8.85	89.71
beats, dotted half note	76.90 ± 4.00	91.48
notes, one note	85.71 ± 13.47	92.50
notes, two notes	82.73 ± 13.85	93.18
notes, three notes	81.67 ± 10.24	91.55
notes, four notes	79.23 ± 14.07	91.72

Table 2: Voice Numbering Accuracy for Experiment 1

Window type and size	Soundness	Completeness
beats, eighth note	84.62	71.88
beats, quarter note	82.76	78.13
beats, half note	83.33	78.13
beats, dotted half note	86.21	75.00
notes, one note	89.47	53.13
notes, two notes	88.00	68.75
notes, three notes	84.62	71.88
notes, four notes	84.62	71.88

5.3 Experiment 1

In Experiment 1, we trained the same-voice predicate on measures 1–8 of the Ciaccona. This excerpt presents the melody and harmony in explicit polyphonic style. We then tested the same-voice predicate on measures 9–12, which are similar in style to 1–4, but more ornamented.

The results of this experiment appear in Table 1. The C-V training accuracies are noticeably lower than the new data accuracies, likely due to the additional figuration present in measures 9–12. These ornaments frequently occur in a single voice at a time, while the other voices remain constant. It is therefore usually easy to determine that all the notes of each ornament belong in the same voice, thereby raising the new data accuracy on measures 9–12.

The soundness and completeness values from the assign-voice-numbers algorithm, shown in Table 2, suggest little connection between window size and soundness or completeness in this explicit polyphonic texture. This could be due to the fact that there are few repeated tonal or rhythmic patterns for the decision tree inducer to recognize.

Table 3: Accuracy of Learned Predicate for Experiment 2

Window type and size	C-V Accuracy	New Data Accuracy
beats, eighth note	95.00 ± 8.05	71.74
beats, quarter note	97.50 ± 5.27	72.89
beats, half note	93.08 ± 9.21	84.75
beats, dotted half note	94.12 ± 6.79	82.18
notes, one note	100.00 ± 0.00	87.23
notes, two notes	100.00 ± 0.00	72.04
notes, three notes	97.50 ± 5.27	69.56
notes, four notes	98.00 ± 4.22	71.98

Table 4: Voice Numbering Accuracy for Experiment 2

Window type and size	Soundness	Completeness
beats, eighth note	66.67	40.48
beats, quarter note	62.07	45.24
beats, half note	66.67	57.14
beats, dotted half note	51.43	42.86
notes, one note	81.82	42.86
notes, two notes	66.67	40.48
notes, three notes	66.67	40.48
notes, four notes	64.29	42.86

Table 5: Accuracy of Learned Predicate for Experiment 3

Window type and size	C-V Accuracy	New Data Accuracy
beats, eighth note	95.11 ± 3.02	97.85
beats, quarter note	93.54 ± 3.25	98.17
beats, half note	91.22 ± 2.21	95.67
beats, dotted half note	89.09 ± 1.67	90.89
notes, one note	97.00 ± 4.83	100.00
notes, two notes	96.84 ± 4.44	99.47
notes, three notes	96.55 ± 3.25	99.65
notes, four notes	95.00 ± 3.39	98.40

Table 6: Voice Numbering Accuracy for Experiment 3

Window type and size	Soundness	Completeness
beats, eighth note	97.70	91.40
beats, quarter note	96.67	93.55
beats, half note	100.00	94.62
beats, dotted half note	96.59	90.32
notes, one note	N/A[1]	0.00
notes, two notes	100.00	49.46
notes, three notes	100.00	49.46
notes, four notes	97.78	94.62

5.4 Experiment 2

In our next experiment, we trained a new same-voice predicate on measures 33–36 of the Ciaccona. This passage is one of the more stark examples of implicit polyphony in the Ciaccona, as the sequence of eighth notes and the large leaps present indicate the presence of multiple voices. We then applied the same-voice predicate to classify note pairs in measures 37–40, a passage that has a texture that is not similar, but not identical to the training data of measures 33–36. The contours of both excerpts are similar; however, the training measures consist mostly of eighth notes, while the testing measures include only sixteenth notes. Additionally, even though the note values are mostly identical through each excerpt, the voice separation is less apparent in measures 37–40 than in 33–36. Table 3 reflects this, as the C-V accuracies are much higher than the new data accuracies.

Because of the dissimilarities between our training and testing examples in this experiment, the results for the assign-voice-numbers algorithm, shown in Table 4, were not as high as from the first experiment.

5.5 Experiment 3

In Experiment 3, we trained a new same-voice predicate on measures 89–92 of the Ciaccona, a passage consisting entirely of rapid arpeggiation of chords. We then tested same-voice on an excerpt of similar texture, namely the next four measures of the Ciaccona. As always, each predicate was tested using an appropriate window; the results are shown in Table 5, and the results of assign-voice-numbers appear in Table 6.

The data show that the predicate's accuracy declines slightly as window size is increased, while the standard deviation falls as well — illustrating a bias-variance trade-off. In this arpeggiated texture, finding the rhythmic pattern becomes easier as more instances are seen.

5.6 Experiment 4

For our final experiment, we combined all three previous training sets and used the aggregate data to train a new same-voice predicate. Because we expected that musical texture plays a large part in determining voice segregation, and we did not have any features to determine texture directly, we did not know what accuracies to expect. We tested our learned predicate on all three testing sets individually and also on their union, mirroring the training procedure.

While the training (C-V) accuracies in Table 7 obtained are lower than the respective accuracies in Experiments 2 and 3, they are higher than in Experiment 1. This may be due to the fact that Experiment 3's training set contains more notes than the other two, and therefore Experiment 3's instances outnumber the other examples', causing them to exert the greatest influence in the learning algorithm.

The testing data accuracies indicate that training on a combination of three different musical textures did not hinder learning the same-voice predicate. When we compared the individual accuracies for each of the three testing sets evaluated by the aggregate predicate against the corresponding accuracies evaluated by the separately-trained predicates, we discovered there was little degradation at all. In a few cases, the individual accuracies for the aggregate predicate *rose*, likely because each testing set shares characteristics of the other training sets that individually-trained predicates cannot capture.

The levels of accuracy attained by the globally-trained predicate imply that it is feasible to combine multiple styles and textures when training, as not much accuracy

[1] No adjacent notes ever classified in same voice

Table 7: Accuracy of Learned Predicate for Experiment 4

Window type and size	C-V Accuracy	New Data Accuracy
beats, eighth note	90.47 ± 4.20	93.70
beats, quarter note	90.19 ± 3.24	92.50
beats, half note	87.70 ± 1.55	90.88
beats, dotted half note	86.68 ± 2.15	85.20
notes, one note	94.74 ± 4.96	94.59
notes, two notes	90.86 ± 5.35	90.10
notes, three notes	92.98 ± 4.14	91.28
notes, four notes	91.33 ± 3.12	92.85

Table 8: Voice Numbering Accuracy for Experiment 4

Window type and size	Soundness	Completeness
beats, eighth note	93.67	79.04
beats, quarter note	89.58	77.25
beats, half note	88.59	78.44
beats, dotted half note	88.44	77.25
notes, one note	85.37	20.96
notes, two notes	81.48	56.89
notes, three notes	89.00	53.89
notes, four notes	93.10	80.84

on unseen data is sacrificed. While the soundness and completeness levels shown in Table 8 are not as high as in two of the earlier experiments, they still suggest that little degradation occurs when training on mixed textures; the feature set is apparently robust enough to distinguish styles.

6 DISCUSSION AND FUTURE WORK

While the previous four experiments illustrate that training a predicate to segregate voices is a promising approach, there are methods by which VoiSe could be improved.

It is important to recall that a highly accurate learned same-voice predicate does not necessarily imply that assign-voice-numbers will produce a highly sound or complete voice numbering. Because of the different windowing techniques, and the fact that the assign-voice-numbers algorithm must break ties when multiple notes satisfy the selection criteria, a highly-accurate predicate may lead to a sound but non-complete voice numbering, as seen in Experiment 3.

One may also ask, when measuring accuracy for the assign-voice-numbers algorithm, why the soundness levels are consistently higher than the completeness levels. This phenomenon occurs because as the number of voices in a training excerpt increases, the number of note pairs in different voices increases faster than the number of pairs in the same voice. Therefore, the same-voice predicate learns to err on the side of not classifying note pairs as being in the same voice. Recall that the assign-voice-numbers algorithm works by looking ahead in time, within an appropriate window, for the next note that is in the same voice as the one it is currently examining. If a ground-truth voice drops out for a duration greater than the size of the window that assign-voice-numbers is using, then there is no way that the algorithm can correctly place notes on opposites sides of the gap in the same voice. When this occurs, assign-voice-numbers ends up dividing a ground-truth voice into sections, which by definition will lower the completeness measurement.

VoiSe is also limited in the fact that it is allowed a single pass through the data to make same-voice determinations and produce a voice numbering. While this tactic certainly works up to a point in our experiments, we suspect that humans can solve this problem more accurately because they may make multiple passes over the music. That is, if one determines that a group of notes definitely is or is not in the same voice, one can use this fact at other places in the piece where a similarity to the former group is discovered. This suggests an iterative approach, in which VoiSe's predictions can be refined over time.

REFERENCES

U. Baumann. A procedure for identification and segregation of multiple auditory objects. In S. Greenberg and M. Slaney, editors, *Computational Models of Auditory Function*, pages 268–280. IOS Press, Berkeley, 2001.

E. Cambouropoulos. From MIDI to traditional musical notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Models for Composition, Performance and Analysis*, 2000.

E. Chew and X. Wu. Separating voices in polyphonic music: A contig mapping approach. In *Computer Music Modeling and Retrieval: Second International Symposium*, pages 1–20, 2004.

R. O. Gjerdingen. Apparent motion in music? *Music Perception*, 11(4):335–370, 1994.

D. Huron. Tonal consonance versus tonal fusion in polyphonic sonorities. *Music Perception*, 9(2):135–154, 1991.

D. Huron. Tone and voice: A derivation of voice-leading from perceptual principles. *Music Perception*, 19(1): 1–64, 2001.

J. Kilian and H. H. Hoos. Voice separation — a local optimisation approach. In *Proceedings of the Third Annual International Symposium on Music Information Retrieval*, pages 39–46, 2002.

O. Lartillot. Discovering musical patterns through perceptive heuristics. In *Proceedings of the Fourth Annual International Symposium on Music Information Retrieval*, pages 89–96, 2003.

C. Meek and W. P. Birmingham. Thematic extractor. In *Proceedings of the Second Annual International Symposium on Music Information Retrieval*, pages 119–128, 2001.

L. Smith and R. Medina. Discovering themes by exact pattern matching. In *Proceedings of the Second Annual International Symposium on Music Information Retrieval*, pages 31–32, 2001.

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

P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44, 1997.