VISA: THE VOICE INTEGRATION/SEGREGATION ALGORITHM

Ioannis Karydis **Alexandros Nanopoulos** **Apostolos N. Papadopoulos** **Emilios Cambouropoulos**

Department of Informatics
Aristotle University of Thessaloniki
{karydis, ananopou, papadopo}@csd.auth.gr

Dept. of Music Studies
Aristotle Univ. of Thessaloniki
emilios@mus.auth.gr

ABSTRACT

Listeners are capable to perceive multiple voices in music. Adopting a perceptual view of musical 'voice' that corresponds to the notion of auditory stream, a computational model is developed that splits musical scores (symbolic musical data) into different voices. A single 'voice' may consist of more than one synchronous notes that are perceived as belonging to the same auditory stream; in this sense, the proposed algorithm, may separate a given musical work into fewer voices than the maximum number of notes in the greatest chord. This is paramount, among other, for developing MIR systems that enable pattern recognition and extraction within musically pertinent 'voices' (e.g. melodic lines). The algorithm is tested against a small dataset that acts as groundtruth.

1. INTRODUCTION

Recently, there have been a number of attempts [3, 5, 9, 10, 11, 12, 13] for the computational modelling of the segregation of polyphonic music into separate voices. Much of this research is influenced by empirical studies in music perception [1, 6, 7] as well as by musicological concepts such as melody, counterpoint, voice-leading and so on.

It appears that the term 'voice' has different meanings for different research fields (traditional musicology, music cognition and computational musicology). A detailed discussion is presented in [1]. A single musical example is given in Fig. 1 that presents three different meanings of the term voice.

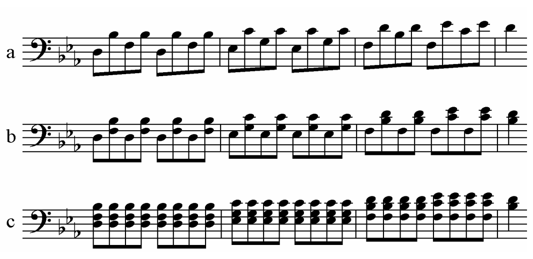


Figure 1 How many voices in each example?

Standard understanding of the term voice refers to a *mono*phonic sequence of successive non-overlapping musical tones; a single voice is thought not to contain multi-tone sonorities. However, if 'voice' is seen in the light of auditory streaming, then, it's clear that the standard meaning is not sufficient. It's possible that a single monophonic sequence may be perceived as more than one voice/stream (e.g., pseudopolyphony or implied polyphony) or that a passage containing concurrent notes may be perceived as a single perceptual entity (e.g., homophonic passages in Fig.1c).

The perceptual view of voice adopted in this study, allows for multi-tone simultaneities in a single 'voice', while bearing the most significant difference of the proposed model with existing ones. In Fig. 1, all existing algorithms (see exception regarding Kilian and Hoos's algorithm in the next section), that are based on purely monophonic definitions of voice, would find two voices in the second example (Fig. 1b) and three voices in the third example (Fig. 1c). It is clear that such voices are not independent voices. In terms of harmonic voices, all examples can be understood as comprising of three voices (triadic harmony). In terms of perceptual voices/streams, each example is perceived as a single auditory stream (harmonic accompaniment); it makes musical sense to consider the notes in each example as a single coherent whole, as a unified harmonic sequence. The proposed algorithm determines a single 'voice'/stream in all three examples.

In this paper, initially, a number of recent voice separation algorithms are briefly described and their main differences to the current proposal are highlighted. Then, the fundamental auditory streaming principles, forming the basis of the proposed model, are presented. The description of the proposed algorithm follows, concluded by evaluation of the algorithm and results on ten different musical works.

2. RELATED WORK

Voice separation algorithms such as [3, 5, 10, 11, 12, 13] assume that 'voice' is a monophonic sequence of successive non-overlapping musical tones. The underlying perceptual principles that organise tones in voices are the principles of temporal and pitch proximity (cf. Huron's [7] Temporal Continuity and Pitch Proximity principles). In essence, these models attempt to determine a minimal number of monophonic lines/voices such that each line consists of successions of tones that are maximally proximal in the temporal and pitch dimensions.

Kilian and Hoos's [9] model is pioneering in the sense that multi-note sonorities within single voices are allowed. The pragmatic goal of the algorithm is the derivation of reasonable score notation - not perceptually meaningful voices (see [9], p.39). The results are not necessarily perceptually valid (e.g., a 4-part homophonic piece may be 'forced' to split into two musical staves that do not correspond to perceptually pertinent streams). The algorithm does not discover automatically the number of independent musical 'voices' in a given excerpt; if the user has not defined the maximum number of voices, the algorithm automatically sets the maximum number equal to the maximum number of co-sounding notes – in this case the algorithm becomes similar to all other algorithms mentioned above (see discussion in [8]).

3. PERCEPTUAL PRINCIPLES FOR VOICE SEPARATION

Bregman [1] offers an in depth exploration of processes relating to perceptual integration/segregation of simultaneous auditory components. Coordinated and synchronously evolving in time sounds tend to be perceived as components of a single auditory event. Concurrent tones that start, evolve and finish together tend to be merged perceptually. The proposed principle (below) relates to Huron's Onset Synchrony Principle [7] but it differs in a number of ways as discussed by Cambouropoulos [2].

Synchronous Note Principle: Notes with synchronous onsets and same inter-onset intervals IOIs (durations) tend to be merged into a single sonority.

The horizontal integration of musical elements (such as notes or chords) relies primarily on two fundamental principles: *Temporal Continuity* and *Pitch Proximity* [7].

It is suggested, that a voice separation algorithm should start by identifying synchronous notes that tend to be merged into single sonorities and then use the horizontal streaming principles to break them down into separate streams (most algorithms ignore the vertical component). This is an optimisation process wherein various perceptual factors compete for the production of a 'simple' interpretation of the music in terms of a minimal number of streams.

4. *VISA*: THE VOICE INTEGRATION/SEGREGATION ALGORITHM

This section describes the proposed voice separation algorithm *VISA*.

4.1. Merging Notes into Single Sonorities

During vertical integration, according to the synchronous note principle, we have to determine when to merge concurrent notes and thus require a merging criterion.

Given a set of concurrent notes *S*, the algorithm examines the frequency of appearing concurrency in a certain musical excerpt (window) around them. If inside the window most co-sounding notes have different onsets/offsets, then it is most likely that we have independent monophonic voices so occasional synchronous notes should not be merged. Thus, by having a user-defined threshold *T* that signifies frequency, if the ratio of concurrency is more than *T*, we merge the notes of *S* as a single sonority.

4.2. The Algorithm

The Voice Integration/Segregation Algorithm (VISA) receives as input the musical piece in the form of a list *L* of notes that are sorted according to their onset times, a window size *w*, and the threshold *T*. The output is a set of lists *V* (initially empty). After termination, each list contains the notes of each detected voice, sorted by onset time. Notice that VISA does not demand a-priori knowledge of the number of voices. The proposed algorithm is illustrated in Fig. 2.

```
VISA(NoteList L, Set NoteList V, int w, float T)
begin
    V ← ∅;
    while ((SLS ← getNextSweepLineSet(L)) ≠∅)
    begin-while
        C ←ClusterVertically(SLS, w, T);
        if (|V| < |C|)
        begin-if
            MatchVoicesToClusters(V, C);
        else
            MatchClustersToVoices(C, V);
        end-if
    end-while
end
```

Figure 2 The VISA algorithm.

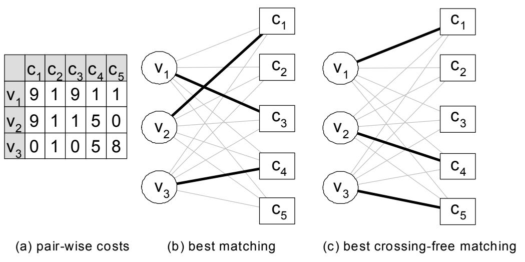
In VISA, a sweep line, starting from the beginning of *L*, proceeds in a step-wise fashion to the next onset time in *L*. The set of notes having onsets equal to the position of the sweep line is denoted as sweep line set (SLS). Next, every SLS is divided into clusters by partitioning the notes in the SLS into a set of clusters *C*. The ClusterVertically procedure, detects contextual information, accepting, thus, *w* and *T* as parameters. If, based on context, we decide to merge concurrent notes, each cluster contains all notes with the same IOI. Otherwise, if merging is not decided, each cluster contains a single note.

Given the set of clusters, *C*, a bipartite graph is formed in order to assign them to voices, where one set of vertices corresponds to the currently detected voices and the other set corresponds to the clusters in *C*. Between every pair of vertices in the graph, we draw an edge to which we assign a cost. Having determined the cost on every edge, we can solve the assignment problem by finding the *matching* with the lowest cost in the bipartite graph. Two cases are possible: (i) If $|V| < |C|$, then we match voices to clusters. This is done by assigning to each of the currently detected voices a cluster, in a way that the total cost is minimised. The remaining clusters that have not been assigned to a voice constitute new voices that are added to V. This is handled inside procedure MatchVoicesToClusters. (ii) Conversely, if $|V| \geq |C|$, we match clusters to voices, i.e., each cluster is assigned to one of the currently detected voices, in a way that the total cost is minimised. Nevertheless, a matching may not be feasible, in which case new voices, enabling a matching, are created.

Finally, we introduce two extra constraints to the problem of a matching; (a) voice crossing should be avoided and (b) the top voice should be minimally fragmented [12]. Section 4.3 presents more details for the inclusion of constraints in the matching procedure.

4.3. The Matching Process

For convenience, we convert the minimisation problem to an equivalent maximisation one. For this reason, the assignment of the cost $w(e_{ij})$ between a voice v_i and a cluster c_j is converted to $\max\{e_{kl}\} - w(e_{ij})$, where $\max\{e_{kl}\}$ is the maximum edge cost determined for the specific instance of the matching problem (and this cost is due to the edge connecting voice v_k and cluster c_l) .

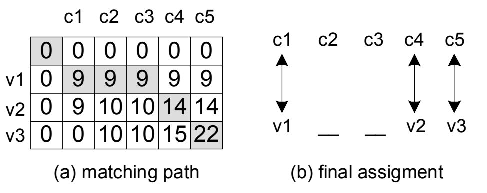


	c_1	c_2	c_3	c_4	c_5
v_1	9	1	9	1	1
v_2	9	1	1	5	0
v_3	0	1	0	5	8

Figure 3 Maximum matching examples

Traditional bipartite matching algorithms do not preserve the order of the matching. In our case, order preservation is important (voice crossing), formulating a new problem that can not be directly tackled by bipartite matching algorithms. Figure 3 illustrates three voices, five clusters and the pair-wise cost for their assignment. A maximum weighted matching (Fig. 3b), with a total cost of 23 does not necessarily avoid voice crossing, while a crossing-free maximum weighted matching with cost of 22 is depicted in Fig. 3(c). The proposed matching can handle larger number of voices/clusters, and is based on [4].

The matching process is depicted in Fig. 4(a), where each cell of the matrix M represents the total matching cost. The matrix is filled according to the recurrence equation (see [8]) of the dynamic programming.



		c1	c2	c3	c4	c5
	0	0	0	0	0	0
v1	0	9	9	9	9	9
v2	0	9	10	10	14	14
v3	0	0	10	10	15	22

Figure 4 The matching process

The best matching cost itself does not provide for the assignment of voices to clusters. To determine the matching path (Fig. 4a) we perform a trace-back process starting at the cell which contains the best matching value. In the trace-back process we never choose a vertical cell, since no gaps are allowed to be placed on the cluster sequence, meaning that all voices must be matched. The final assignment is given in Fig. 4(b).

According to the previous discussion, the running time of the algorithm is $O(n*m)$ (n>=2, m>=2) where n is the number of voices and m the number of clusters. Evidently, we need $O(n*m)$ time to calculate all elements of the matrix M, and $O(n+m)$ time to reconstruct the matching path.

5. EXPERIMENTS AND RESULTS

The proposed algorithm has been tested on ten pieces with clearly defined streams/voices which are used as groundtruth. The first six pieces include four fugues and two inventions by J.S.Bach; these polyphonic works consist of independent monophonic voices. Two mazurkas and a waltz by F.Chopin consist of a melody (upper staff) and accompanying harmony (lower staff). Finally, the "Harmony Club Waltz" by S.Joplin has two parallel homophonic streams (chordal 'voices') that correspond to the two piano staves. See excerpts in Figs 5, 6, 7.

In this pilot study, our aim is to examine if a single algorithm can be applied to two very different types of music (i.e. pure polyphonic music and music containing clear homophonic textures). All the parameters of the algorithm are the same for all ten pieces, while the number of streams/voices is determined automatically. It should be noted that for the pieces by Chopin and Joplin all other voice separation algorithms would determine automatically at least four different voices (up to eight voices) that do not have perceptual validity (and musicologically are problematic).

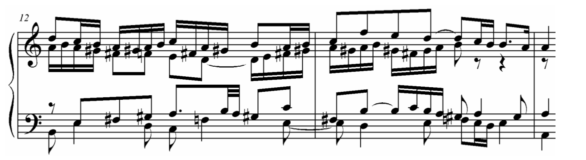


Figure 5 Four independent streams/voices are present in this excerpt from the Fugue No.1 in C major, WTCI, BWV846 by J.S.Bach. The algorithm performs voice separation correctly except for the last five notes of the upper voice which are assigned to the 2nd voice rather than the first voice, as these are closer by a semitone to the last note of the second voice.

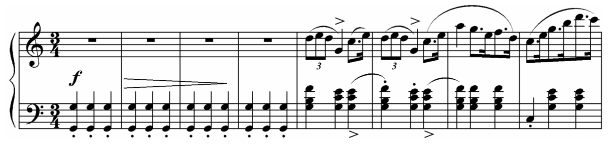


Figure 6 In the opening of the Mazurka, Op.7, No.5 by F.Chopin, the algorithm detects correctly one voice (low octaves) and, then, switches automatically to two voices (melody and accompaniment).

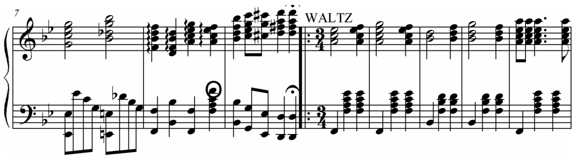


Figure 7 Two independent chordal streams/voices are correctly determined by the algorithm in this excerpt from the "Harmony Club Waltz" by S.Joplin; the only mistake is indicated by the circled note which is placed 'erroneously' in the upper stream (because of pitch proximity).

The evaluation metrics used is the precision of the obtained result. For the previously described musical dataset, Table 1 shows the results. The effectiveness of the proposed methodology is evident by the high precision rates achieved for all ten pieces.

Musical Work	*Precision*
J.S.Bach, Fugue No.1 in C major, BWV846	92,38%
J.S.Bach, Fugue No.14 in F# major, BWV859	95,56%
J.S.Bach, Fugue No.11 in F major, BWV 856	87,31%
J.S.Bach, Fugue No.7 in E major, BWV 852	97,52%
J.S.Bach, Invention No.1 in C Major, BWV 772	99.34%
J.S.Bach, Invention No.13 in A Min, BWV 784	96.45%
F. Chopin, Mazurka, Op.7, No.5	100%
F. Chopin, Mazurka in A Minor, Op. 67, No.4	88.8%
F. Chopin, Waltz in B Minor, Op. 69, No. 2	90.31%
S. Joplin, "Harmony Club Waltz"	98.12%

Table 1 Results in terms of precision for the dataset.

The results were examined in detail (qualitative analysis). Most wrong results were given in cases where the number of voices changes and erroneous connections are introduced primarily due to pitch proximity (e.g., see last upper five notes in Fig. 5). Kilian and Hoos [9] address this same problem claiming that, in essence, it is unsolvable at the note level. A second kind of problem involves voice crossing. Since voice crossing is disallowed, notes at points where voices cross (in the Bach fugues) are assigned to wrong voices. A third type of mistake relates to the breaking of vertically merged notes into sub-sonorities and allocating these to different voices; in this case the breaking point in the sonority may be misplaced (e.g., circled note in Fig. 7).

6. CONCLUSIONS

In this paper, the notions of voice and auditory stream have been examined. It is suggested that, if 'voice' is understood as a musicological parallel to the concept of auditory stream, then multi-note sonorities should be allowed within individual 'voices'. It is proposed that a first step in voice separation is identifying synchronous note sonorities and, then, breaking these into sub-sonorities incorporated in horizontal streams or 'voices'.

The proposed voice separation algorithm, *VISA*, incorporates the two principles of temporal and pitch proximity, and additionally, the Synchronous Note Principle, performing in the general case where both polyphonic and homophonic elements are mixed together.

7. REFERENCES

[1] Bregman, A (1990) *Auditory Scene Analysis: The Perceptual Organisation of Sound.* The MIT Press, Cambridge (Ma).

[2] Cambouropoulos, E. (2006) 'Voice' Separation: theoretical, perceptual and computational perspectives. In *Proceedings of the 9th International Conference in Music Perception and Cognition (ICMPC2006)*, 22-23 August, Bologna, Italy.

[3] Cambouropoulos, E. (2000) From MIDI to Traditional Musical Notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music*, July 3 - Aug. 3, Austin, Texas.

[4] Cormen, T., Leiserson, C.E., Rivest, R.L. and Stein, C (2001). *Introduction to Algorithms*, The MIT Press, Cambridge (Ma).

[5] Chew, E. and Wu, X. (2004) Separating voices in polyphonic music: A contig mapping approach. In *Computer Music Modeling and Retrieval: Second International Symposium* (CMMR 2004), pp. 1-20.

[6] Deutsch, D. (1999) Grouping Mechanisms in Music. In D. Deutsch (ed.), *The Psychology of Music* (revised version). Academic Press, San Diego.

[7] Huron, D. (2001) Tone and Voice: A Derivation of the Rules of Voice-Leading from Perceptual Principles. *Music Perception*, 19(1):1-64.

[8] Karydis, I., Nanopoulos, A., Papadopoulos, A., Cambouropoulos, E. and Manolopoulos Y. (2007) Horizontal and Vertical Integration/Segregation in Auditory Streaming: A Voice Separation Algorithm for Symbolic Musical Data. *In proceedings of the confernce Sound and Music Computing* (SMC07), Lefkada.

[9] Kilian j. and Hoos H. (2002) Voice Separation: A Local Optimisation Approach. In *Proceedings of the Third International Conference on Music Information Retrieval* (ISMIR 2002), pp.39-46.

[10] Kirlin, P.B. and Utgoff, P.E. (2005) VoiSe: Learning to Segregate Voices in Explicit and Implicit Polyphony. In *Proceedings of the Sixth International Conference on Music Information Retrieval* (ISMIR 2005), Queen Mary, Univ. of London (pp. 552-557).

[11] Madsen, S. T. and Widmer, G. (2006) Separating Voices in MIDI. In *Proceedings of the 9th International Conference in Music Perception and Cognition (ICMPC2006)*, 22-26 August 2006, Bologna, Italy.

[12] Temperley, D. (2001) *The Cognition of Basic Musical Structures*. The MIT Press, Cambridge (Ma).

[13] Szeto, W.M. and Wong, M.H. (2003) A Stream Segregation Algorithm for Polyphonic Music Databases. In *Proceedings of the Seventh International Database Engineering and Applications Symposium* (IDEAS'03).