MUSICAL VOICE INTEGRATION/SEGREGATION: *VISA* REVISITED

Dimitris Rafailidis
Department of Informatics
Aristotle Univ. of Thessaloniki
draf@csd.auth.gr

Emilios Cambouropoulos
Department of Music Studies
Aristotle Univ. of Thessaloniki
emilios@mus.auth.gr

Yannis Manolopoulos
Department of Informatics
Aristotle Univ. of Thessaloniki
manolopo@csd.auth.gr

ABSTRACT

The Voice Integration/Segregation Algorithm (VISA) proposed by Karydis et al. [7] splits musical scores (symbolic musical data) into different voices, based on a perceptual view of musical voice that corresponds to the notion of auditory stream. A single 'voice' may consist of more than one synchronous notes that are perceived as belonging to the same auditory stream. The algorithm was initially tested against a handful of musical works that were carefully selected so as to contain a steady number of streams (contrapuntal voices or melody with accompaniment). The initial algorithm was successful on this small dataset, but was proven to run into serious problems in cases were the number of streams/voices changed during the course of a musical work. A new version of the algorithm has been developed that attempts to solve this problem; the new version, additionally, includes an improved mechanism for context-dependent breaking of chords and for keeping streams homogeneous. The new algorithm performs equally well on the old dataset, but gives much better results on the new larger and more diverse dataset.

1. INTRODUCTION

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 [2]. A perceptual view of voice adopted in previous voice separation modelling attempts [7, 13], allows for multi-tone simultaneities in a single 'voice' – this is the most significant difference of such model(s) with other existing voice separation models [4, 9, 10, 11, 12, 14].

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 is clear that the standard meaning is not sufficient. It is 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.

In Figure 1, all existing algorithms that are based on purely monophonic definitions of voice (except Kilian and Hoos's [8] algorithm that allows fewer voices if forced by the user), would detect five voices that clearly are not independent voices. The VISA algorithm [7] and the new version presented in this paper detect two voices/streams that correspond to melody and accompaniment.

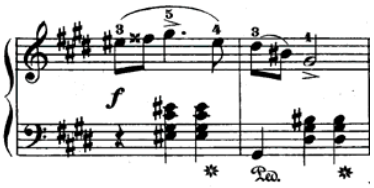

Figure 1 How many voices in this excerpt from Chopin's Mazurka Op.6, No.2?

It is suggested that a general musical voice/stream segregation algorithm should be able to cope with any kind of music, not just musical textures that are constructed by the use of a steady number of monophonic voices (e.g. fugues, chorales, string quartets, etc.). Such an algorithm, among other things, is very useful for developing MIR systems that enable pattern recognition and extraction within musically pertinent 'voices' – for instance, there is no reason to 'look' for melodic patterns in homophonic accompanimental parts of songs.

In this paper, initially, a number of problems related to voice/stream separation not addressed by the model proposed in [7] are presented. A brief description of the first prototype version of the Voice Integration/ Segregation Algorithm (VISA) follows, and, then, a number of improvements to the algorithm are given. After an evaluation of the new prototype on a more extended

SMC 2009, July 23-25, Porto, Portugal

and diverse groundtruth dataset, the paper is concluded by some future suggestions for further improvements.

2. VOICE SEPARATION MODELS

Voice separation models based on a monophonic definition of voice (REFERENCES) attempt to determine a minimal number of lines/voices such that each line consists of successions of tones that are maximally proximal in the temporal and pitch dimensions. Such models perform well on music that is composed of a steady number of voices/lines, but fail to give musicologically or perceptually relevant results in most other cases. The horizontal integration of notes relies primarily on two fundamental auditory streaming principles: *Temporal Continuity* and *Pitch Proximity* [6].

Adopting a perceptual view of voice, which is very close to the notion of auditory stream, two recent studies [7, 12] allow multi-tone simultaneities in a single 'voice'. In addition to the two previously mentioned perceptual principles these models enable vertical integration based on the Synchrony Note Principle [2], whereby 'notes with synchronous onsets and same inter-onset intervals IOIs (durations) tend to be merged into a single sonority.'

VISA [7] starts by identifying synchronous notes that tend to be merged into single sonorities and, then, uses 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. If the reader is not acquainted with VISA, we suggest that section 3.1 be read before the next section (2.1).

The algorithm presented herein has been developed as a means to explore more systematically the ideas and principles of musical auditory streaming in symbolic musical data; it is an exploratory prototype that requires further development. The proposed prototype is not directly comparable to other voice separation algorithms as its underlying definition of 'voice' is different and has a different aim. In this paper we will compare our new version of VISA with the earlier version [7].

2.1. Problems of VISA and improvements

VISA was initially tested on ten musical examples [7] that were carefully selected so as to contain a steady number of streams (i.e. musical works comprising of contrapuntal melodic lines, or of melody and homorhythmic accompaniment). The algorithm performed well on this limited dataset. However, we discovered that the algorithm ran into serious problems when tested on music that contained non-homorhythmic homophonic accompanimental textures or diverse musical textures (homophonic and polyphonic together).

The main problem of this early version of the algorithm is that, when a new voice/stream appears, it is available for continuation throughout the rest of the piece. This is no serious problem in contrapuntal polyphonic works where the number of voices remains steady throughout a musical work. However, in homophony we usually have a single stream, i.e. one harmonic homorhythmic stream, or two streams, i.e. one melodic voice plus rhythmically independent accompaniment. Occasionally, additional rhythmically independent lines may appear locally but these usually disappear after their emergence rather than remain active throughout the rest of the piece.

When the early version of VISA breaks a homophonic piece into three (or more) streams locally, it tries to find the best continuation for these three streams throughout the rest of the piece; occasionally the third stream may erroneously be selected to continue stream 1 or 2, or all three streams may continue in parallel. For instance, in Figure 2 (measure 11) we have three streams - the algorithm considers the upper voice as stream 3 since it has already allocated streams 1 and 2 to the first notes in the measure – the mistake is then propagated to the rest of the score as the next top notes are closer to stream 3 (actually, stream 2 is abandoned and stream 3 and 1 remain active in reverse order, i.e. stream 3 above stream 1). In Figure 3 the algorithm erroneously locates three streams in measure 29 (as the bass note overlaps with the following notes of the chord they cannot all be placed in one stream – see discussion in Section 4), and from there on it continues 'giving' notes to all three streams rather than returning to two streams (melody and accompaniment). Such a relatively simple mistake may decrease accuracy dramatically as a local increase in streams may be erroneously be propagated throughout the rest of the score.

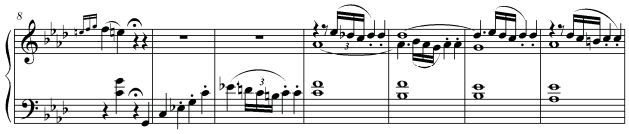

Figure 2 Excerpt from Beethoven's Sonata Op.2, No.1, Allegro Con Brio.

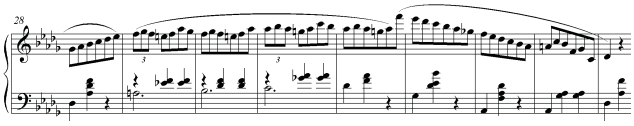

Figure 3 Excerpt from Chopin's Waltz Op. 64, No.1

A simple solution has been introduced to address this problem. The solution is based on the observation that in homophony, music is perceived as a single stream that may be 'fattened' or 'thinned' by adding or subtracting extra streams, whereas in polyphonic music, streams have an

independent life and are equally important. Following this, when the texture is locally homophonic (i.e. many notes start and end together), the algorithm is forced to switch back to streams 1 and 2 after having identified three (or more) streams. This simple modification increased accuracy significantly as seen in Table 1.

A second important improvement involves the breaking of chords consisting of equal duration notes. In the early version of VISA this was partially incorporated in the program in association to the Top Voice Rule, i.e., the top voice should be minimally fragmented. This is handled by adding a penalty to the cost of a voice continuation that does not fulfil this rule. To find the continuation with the minimal cost, a chord may be split so that one note can be assigned to the top voice. In our new proposal, a chord consisting of equal duration notes is split into sub-chords based on the context of existing or forthcoming independent streams. That is, if in the vicinity of the current chord there are more voices, the chord may be split so as to match the adjacent voice structure. The Top Voice Rule, thus, becomes a special case of this general vertical cluster splitting process. For instance, in Figure 4 we perceive a melodic line that lies within a static harmonic stream; the chords marked by an asterisk consist of equal duration notes so initially they are merged into a single vertical cluster by VISA – the proposed function that breaks chords (vertical clusters) 'pulls out' the second note of these chords and assigns it to the independent melodic voice.

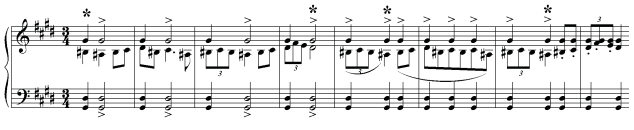

Figure 4 Opening of Chopin's Mazurka Op. 6, No.2.

Further smaller modifications that improve the algorithm's performance include the following: Firstly, the pitch distance between notes/chords takes into account not only the pitch of the current notes and the last notes of preceding voices, but also the second-to-last notes of preceding voices. In case the last pitches of two voices coincide, the algorithm could not decide which current pitch should be assigned to which of the two unison pitches; taking into account the second-to last pitches resolves such ambiguous cases. Secondly, the distance metric takes into account not only pitch and temporal distance, but additionally a new parameter that favours homogeneity of streams in terms on number of co-sounding tones. In other words, linking a chord cluster with many tones to a single note is discouraged and contributes to a larger distance, whereas linking similar density clusters adds smaller cost. We discovered that this homogeneity factor solved problems in a number of cases; however, there are cases where this factor is counterproductive.

3. THE REVISED *VISA* ALGORITHM

The previous algorithm posed by Karydis et al. [7] and also our current revised implementation consist of two steps: fist, vertical integration which merges notes with same onsets and durations if the musical context is homophonic, and second, links notes/chords horizontally into voices/streams.

3.1. Brief description of VISA

The original Voice Integration/Segregation Algorithm [7] accepts as input a musical piece in symbolic form and outputs the number of detected musical voices/streams. At present, the algorithm is applied to quantized musical data; expressively performed musical data require quantization before being fed into the algorithm. The appropriate number of streams is determined automatically by the algorithm and can be lower than the maximum number of notes of the largest chord.

VISA moves in a step-wise fashion through the input sequence of musical events (individual notes or concurrent note sonorities). Let the entire musical piece be represented as a list L of notes that are sorted according to their onset times. A sweep line, starting from the beginning of L, proceeds through the onset times in L. The set of notes that have onsets equal to a position of the sweep line is denoted as sweep line set (SLS).

For a set of concurrent notes at a given point (SLS), we have to determine when to merge them according to the *Synchronous Note Principle*. Because it is possible that synchronous notes may belong to different voices, we need a way to decide if such merging should be applied. For each SLS, the algorithm examines a certain musical context (window) around them. If inside the window, most co-sounding notes have different onsets or offsets, then it is most likely that we have polyphonic texture (independent monophonic voices), so occasional synchronous notes should not be merged - each note is considered to be a singleton cluster. If most notes are concurrent (same onsets and IOIs) implying a homophonic texture, then they should be merged - concurrent notes form a cluster. This way, each SLS is split into a number of note clusters. At the present stage, the window size w and homophony/polyphony threshold T have been determined manually (same for all the data) by finding values that give optimal results for the selected test data set.

For each SLS in the piece, we have a set of previously detected voices (V) and the current set of note clusters (C). Between every detected voice of V and each note cluster of C, we draw an edge to which we assign a cost. The cost function calculates the cost of assigning each cluster to each voice according to the *Temporal Continuity Principle* and the *Pitch Proximity Principle*. Notes that overlap receive a cost value equal to infinity.

A dynamic programming technique finds the best matching (lowest cost) in the bipartite graph between previous voices and current clusters. If voices are fewer than clusters, then one or more voices may be (temporarily) terminated. If clusters are fewer than voices, then new voices may appear. The matching process, additionally, takes into account two constraints. The first one is that voice crossing should be avoided. Therefore a sub-optimal solution in terms of cost may be required that avoids voice crossing. The second one is that the top voice should be minimally fragmented (Top Voice Rule by [11]). This is handled by adding a penalty to the cost of a matching that does not fulfill this rule - to find the matching with the minimal cost - a cluster may be split into sub-clusters, so that one can be assigned to the top voice.

3.2. Revised Version of VISA

3.2.1. Numbering of Voices/Streams

In music that is primarily homophonic, the tendency is to have one or two stable streams (pure homorhythmic texture, or melody and harmonic accompaniment), whereas further independent voices/streams appear only locally (see discussion in Section 2.1). To avoid keeping 'alive' extra voices/streams (e.g. third or fourth stream), a simple solution has been introduced: when the texture is locally homophonic (i.e. many notes start and end together), the algorithm is forced to switch back to streams 1 and 2 after having identified three (or more) streams. That is, when in the MatchingVoicesToClusters procedure we have more voices than clusters and also the context is homophonic, the current clusters are assigned to the basic streams 1 and 2.

In the middle of the excerpt in Figure 5 we have three Voices, V1: {N5, N10}, V3: {N6, N9, N11, N12} and V2: {N7, N8}. In the next SLS, note N13 is closer to V2 but is assigned to V1 because the algorithm prefers to abandon V3 moving back to the main two voices.

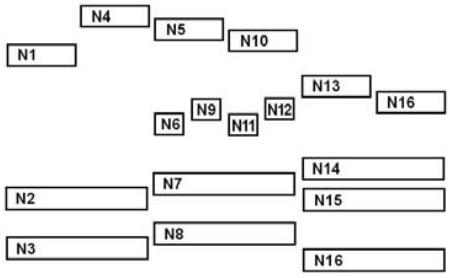

Figure 5 The third voice {N6, N9, N11, N12} is abandoned and, N13 continues the first voice – see text.

3.2.2. Vertical Integration and BreakCluster Method

If a number of notes are integrated vertically (they have same durations) and if the local context is homophonic, then the *BreakCluster* procedure is activated. This procedure looks ahead in the next three SLSs (more generally it can be designed to look in the local neighborhood before and/or after the current SLS); if it finds (using ClusterVertically) that there exist more clusters in one of the following SLSs than in the current SLS, it moves backwards from the SLS (with more clusters) breaking one by one its preceding clusters till it breaks the current SLS cluster. Preceding clusters are broken according to how close notes in the to-be-broken clusters are to the notes of the SLS with more clusters.

In the example of Figure 6, notes in the current SLS1 are clustered vertically into a single cluster as they have same onsets and durations, and also the context is homophonic. In this case, *BreakCluster* is activated and checks whether in any of the next three SLSs there are more clusters than in the current SLS. SLS2 and SLS3 contain a single cluster, but ClusterVertically splits SLS4 into 3 clusters: {N13}, {N14}, and {N15, N16}. Now, moving backwards it breaks the cluster in SLS3 into three clusters based on pitch proximity: {N9}, {N10} and {N11, N12}, then breaks SLS2 into {N5}, {N6} and {N7, N8} and, finally, the current cluster SLS1 into {N1}, {N2} and {N3, N4}. In a different scenario, if an SLS before the third SLS contained more than one clusters, then the BreakCluster procedure would have moved from that SLS backwards to the current SLS.

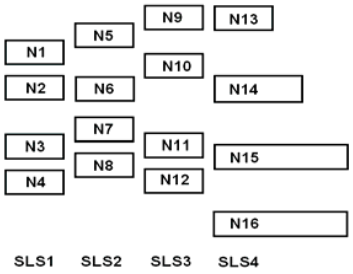

Figure 6 Breaking vertical clusters based on context.

3.2.3. Matching notes to voices and cost calculation

As mentioned in Section 3.1, after determining the clusters for each SLS, a bipartite graph is created for matching notes to voices. Each cell (i,j) of the graph designates the cost (distance) between the last cluster assigned to voice i and the current cluster j. In the previous implementation only pitch and time difference is taken into account for the calculation of the cost. In the current implementation we add a factor that relates to the difference of the number of notes in the two clusters. This difference, that is a kind of homogeneity factor (see Section 2.1), is calculated as $d_{h=}|n_i - n_j|/ n_i + n_j$ (where n_i is the number of notes in cluster i) and contributes by 25% to the total cost (along with 50% pitch difference contribution plus 25% inter-onset difference). In the example of Figure 7, note N6 is closer to cluster {N2,

N3, N4} than cluster {N7, N8, N9} is in terms of average pitch, but the {N7, N8, N9} is assigned to cluster {N2, N3, N4} because the total cost is lower when the homogeneity factor is taken into account.

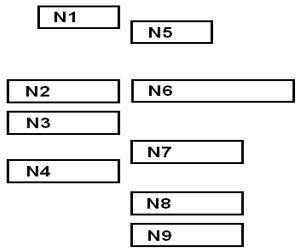

Figure 7 Homogeneity factor (lower chords are assigned to the same voice).

In the previous implementation for the cost calculation, only the last note/cluster in each voice was taken into account. There are cases, however, where more notes/clusters from the past are necessary to resolve ambiguity. In the current implementation, the pitch for each voice is calculated as the weighted average of the pitch of the last two notes/clusters of each voice (80% of the last cluster and 20% of the second-to-last cluster). In the example of figure 8, notes N1 and N3 have the same pitch, so there is ambiguity in assigning the next notes N4 and N5 to the previous voices. If next-to-last notes are taken into account, then the second voice containing note N2 and N3 will have a lower average pitch than the first voice (containing N1) and will be matched correctly to N5.

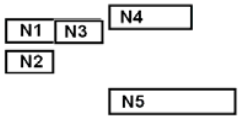

Figure 8 Resolving ambiguity in pitch distance.

4. RESULTS AND FUTURE WORK

The proposed algorithm has been tested on a set of musical extracts for piano.[1] The dataset has been annotated by a music theory research student that was instructed to indicate voices/streams on the scores after listening to the excerpts – a number of musical examples were discussed with him before doing this task – the student did not have knowledge of the computational implementation. The dataset that acted as groundtruth contains the ten pieces used in the initial testing of VISA [7] plus 22 excerpts primarily from piano sonatas by Beethoven (only the openings of the different sections have been annotated, as it is a very tedious task to manual annotate the full scores). The sonatas have been selected as they comprise of diverse musical textures, i.e. homophonic and contrapuntal textures. In future, larger number of music experts may provide groundtruth and/or empirical studies may generate more reliable datasets against which to test algorithms.

The accuracy of the proposed algorithm is measured as the weighted sum for each voice of the proportion of notes correctly assigned to a voice *i* over the total number of notes of voice *i* – each such proportion is multiplied by P_i, where P_i is the percentage of notes belonging to voice *i* against the overall number of notes. Assuming *N* is the number of voices, the accuracy is measured according to equation (1).

$$\text{Accuracy} = \sum_{i=1}^{N} P_i \frac{\#\,\text{notes, correctly assigned to voice i}}{\#\,\text{notes of voice i}} \quad (1)$$

In essence, accuracy counts the total number of notes that have been correctly assigned to the appropriate voice (according to the groundtruth), divided by the total number of notes. This accuracy measure is rather strict in the sense that notes that may have been placed together correctly in the same voice but may have been tagged incorrectly (e.g. placed together in voice *x* instead of voice *y*) are all counted as wrong. This is the main reason why in some cases accuracy is still low.

As can be seen in Table 1, the modifications incorporated in the current version of VISA improve significantly the performance of the algorithm. The average performance of the old version of VISA for the 22 new excerpts is 0.68 (first 22 excerpts in Table 1), whereas the average performance for the new version of VISA is 0.84, which means a 23% increase (16 percent units). The new algorithm does not improve performance on the limited old dataset (last 10 excerpts in Table 1) that was carefully selected to contain excerpts with steady number of 'clean' voices/streams. However, these tests show that overall we have a more flexible algorithm that performs well on diverse musical textures.

Voice/stream segregation is a difficult problem influenced by many different competing factors. The development of computational models such as the VISA algorithm is seen as a means to explore the mechanisms of voice separation to gain a better understanding of the problem with a view to developing more reliable computer models.

The current model can be improved in two ways: firstly, by redesigning the whole algorithm so as to take into account local context in a more integrated manner. Rather than matching clusters of one SLS to the last notes/clusters of previous voices (adding ad hoc cases in which the context is taken into account), it may be more powerful to look continuously for optimal solutions within a larger context.

[1] These pieces were downloaded in Melisma format from the Kern collection (http://kern.humdrum.org)

Secondly, further segregation factors must be taken into account such as tonal fusion, parallelism, pattern similarity, and, even, new overall integration/segregation strategies. For instance, the current method does not allow merging non-isochronous overlapping notes – it is clear, however, that there are cases where this should be allowed (e.g. the notes in the accompaniment of mm. 29-31 in Figure 3 clearly belong to the same voice/stream due to harmonic reasons).

	Old VISA	New VISA
Beethoven, Sonata 2-1 Allegro	0,66	0,86
Beethoven, Sonata 2-1 Adagio	0,82	0,86
Beethoven, Sonata 2-1 Minuet	0,61	0,73
Beethoven, Sonata 2-1 Prestissimo	0,93	0,93
Beethoven, Sonata 2-2 AllegroVivace	0,62	0,80
Beethoven, Sonata 2-2 LargoApp	0,69	0,91
Beethoven, Sonata 2-2 Scherzo	0,49	0,75
Beethoven, Sonata 2-2 Rondo	0,60	0,82
Beethoven, Sonata 2-3 AllegroConBrio	0,40	0,87
Beethoven, Sonata 2-3 Adagio	0,62	0,77
Beethoven, Sonata 2-3 Scherzo	0,74	0,73
Beethoven, Sonata 2-3 AllegroAssai	0,96	0,94
Beethoven, Sonata 10-2 Allegro	0,87	0,89
Beethoven, Sonata 10-2 Allegretto	0,43	0,73
Beethoven, Sonata 10-2 Presto	0,90	0,92
Beethoven, Sonata 13 Grave	0,72	0,98
Beethoven, Sonata 13 AdagioCantabile	0,23	0,56
Beethoven, Sonata 13 Rondo	0,94	0,85
Brahms, Waltz Op39 No8	0,80	0,89
Chopin, Mazurka Op6 No2	0,84	0,93
Chopin, Mazurka Op7 No1	0,70	0,92
Chopin, Waltz Op64 No1	0,43	0,91
Bach, Fugue BWV846	0,92	0,92
Bach, Fugue BWV859	0,96	0,93
Bach, Fugue BWV856	0,87	0,94
Bach, Fugue BWV852	0,97	0,91
Bach, Fugue BWV772	0,99	0,99
Bach, Fugue BWV784	0,96	0,96
Chopin, Mazurka Op7 No5	1,00	0,97
Chopin, Mazurka Op67 No4	0,88	0,88
Chopin, Waltz Op69 No2	0,90	0,96
Joplin, Harmony Club Waltz	0,98	0,92

Table 1 Accuracy for voice separation by the previous and the current implementation of VISA (the last ten pieces were used in the evaluation of the old VISA [7]).

5. CONCLUSIONS

The proposed voice separation algorithm incorporates the two principles of temporal and pitch proximity, and additionally, the Synchronous Note Principle. Allowing both horizontal and vertical integration enables the algorithm to perform well not only in polyphonic music that has a fixed number of 'monophonic' lines, but in the general case where both polyphonic and homophonic elements are mixed together. We have shown in the above preliminary experiment that the proposed algorithm can achieve good performance in diverse musical textures in terms of identifying perceptually relevant voices/streams.

6. REFERENCES

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

[2] Cambouropoulos, E. (2008) Voice and Stream: Perceptual and Computational Modeling of Voice Separation. *Music Perception* 26(1):75-94.

[3] Cambouropoulos, E. (2000) From MIDI to Traditional Musical Notation. In *Proceedings AAAI Workshop on Artificial Intelligence and Music*, Austin, TX.

[4] Chew, E. and Wu, X. (2004) Separating Voices in Polyphonic Music: A Contig Mapping Approach. In *Proceedings 2nd International Symposium on Computer Music Modeling and Retrieval: (CMMR'2004)*, pp. 1-20.

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

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

[7] Karydis, I., Nanopoulos, A., Papadopoulos, A.N. & Cambouropoulos, E., (2007) VISA: the Voice Integration/ Segregation Algorithm. In *Proceedings 8th International Conference on Music Information Retrieval (ISMIR'07),* Vienna, Austria, pp. 445-448.

[8] Kilian j. and Hoos H. (2002) Voice Separation: A Local Optimisation Approach. In *Proceedings 3rd International Conference on Music Information Retrieval* (ISMIR' 2002), Paris, France, pp.39-46.

[9] Kirlin, P.B. and Utgoff, P.E. (2005) VoiSe: Learning to Segregate Voices in Explicit and Implicit Polyphony. In *Proceedings 6th International Conference on Music Information Retrieval (ISMIR'2005),* London, UK, pp. 552-557.

[10] Madsen, S.T. and Widmer, G. (2006) Separating Voices in MIDI. In *Proceedings 9th International Conference in Music Perception and Cognition (ICMPC'2006)*, Bologna, Italy.

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

[12] Rafailidis, D., Nanopoulos, A., Cambouropoulos, E. & Manolopoulos, Y. (2008), Detection of Stream Segments in Symbolic Musical Data. In *Proceedings 9th International Conference on Music Information Retrieval (ISMIR'08),* Philadelphia, PA, pp.83-88.

[13] Szeto, W.M. and Wong, M.H. (2003) A Stream Segregation Algorithm for Polyphonic Music Databases. In *Proceedings 7th International Database Engineering and Applications Symposium (IDEAS'03),* Hong Kong, pp.130-138.