Using Concatenation Cost for Unit Selection of Homosonic Segments in Concatenative Sound Synthesis

Noris Mohd. Norowi, Mas Rina Mustaffa
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia
43400, Serdang,
Selangor, Malaysia
noris@upm.edu.my, MasRina@upm.edu.my

Eduardo Reck Miranda
Interdisciplinary Centre for Computer Music Research
University of Plymouth
Drake Circus
PL4 8AA, Plymouth
United Kingdom
eduardo.miranda@plymouth.ac.uk

***Abstract*— This paper studies the issues surrounding the search and selection process in a general CSS system which may affect the synthesis result, namely the homosonic segments. Homosonic segments are first termed in this study, where it refers to audio files which have one or more of the same sonic properties with each other, but do not sound the same acoustically when played due to the limited audio features extracted during the analysis process. These homosonic segments create confusions within the CSS selection engine. This study proposes a robust solution to overcome this issue by introducing the concatenation cost in addition to the regular target cost. The experiment conducted in this study observes that the use of concatenation cost to help solve the problem is feasible. Further evaluation also suggests that the concatenation cost is an effective solution in solving the challenges involving homosonic segments as the sounds synthesised through concatenation cost function have a better accuracy and possess higher fluency when concatenated from one segment to the next.**

Keywords-concatenative sound synthesis; homosonic audio segments, cocnatenation cost.

I. INTRODUCTION

The ability in computers to perform tasks that were typically thought to require human intelligence is made possible through the advancement in Artificial Intelligence (AI) – the study of man-made computational devices which can be made to act in act in an intelligent manner. One such area that benefited from the rise of these technological advancements is Concatenative Sound Synthesis (CSS). CSS is an art of synthesising new sounds from a composite of many small snippets of audio. CSS resembles that of the query-by-example approach, where it takes in a sound as an input or query (the target unit) and searches for the closest available sound units in the database, which will be concatenated together to produce new sounds (Figure 1).

The approach has been utilised by several computer musicians, whom went to develop several different CSS systems and produced different musical pieces with varying degree of success. Examples include *Improvasher*, *ConQuer*, *CataRT*, *MATConcat* and *Mosievius* [1],[2], [3],[4],[5].

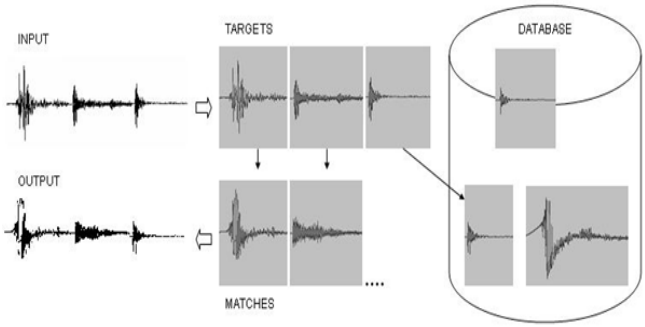

Figure 1. General mechanism of a CSS system

The basic principles of concatenative music synthesis remain similar to those from concatenative speech synthesis, where new sounds are produced from the re-synthesis of an original sound [6],[7]. However, there are several characteristics that set the two apart, other than the media they serve to generate, which gives reference to speech and music respectively. One such attribute is phonemes. In concatenative speech synthesis, phonemes play the most important role, whereas they hold no importance in concatenative speech synthesis. In the same manner, the second attribute which is time, is crucial for music, especially to ensure that the rhythm is in place, but otherwise has very little effect with speech. In general, it can be said that concatenative music synthesis allows more space for creation than concatenative speech is, perhaps due to the closer syntax-semantics quality that speech synthesis has to have in order for it to be understood by its audience. On the other hand, since synthesised music is more artistically-perceived, it is more flexible in terms of intelligibility and naturalness compared to synthesised speech needs to be.

978-1-5090-2954-9/16/$31.00 ©2016 IEEE

Generally, the processes involved in a CSS system starts with a target file, which is also known as the query. The query undergoes both analysis and synthesis phase. During the analysis phase, the target unit are segmented into smaller sound snippets. Following segmentation, relevant information from these sound snippets is then extracted. In the synthesis phase, sound snippets in the database that match closely with the targets are selected and concatenated together forming a long string of sound, which are then synthesised [8]. Figure 2 shows the flow chart of a typical CSS system.

Feature extraction from the analysis phase allows information about the target unit to be compared with the source units in the database. Most of the time, an exact match can be found, especially if the database size is large. An approximate match can be also be used in the case where no exact match is found. However, a problem arises when there are more than one matching source units to be selected. This condition is termed '*Homosonic*' [2]. This study aims to demonstrate the challenges that exists in current CSS systems during the unit selection process, and proposes a robust new approach to counter this.

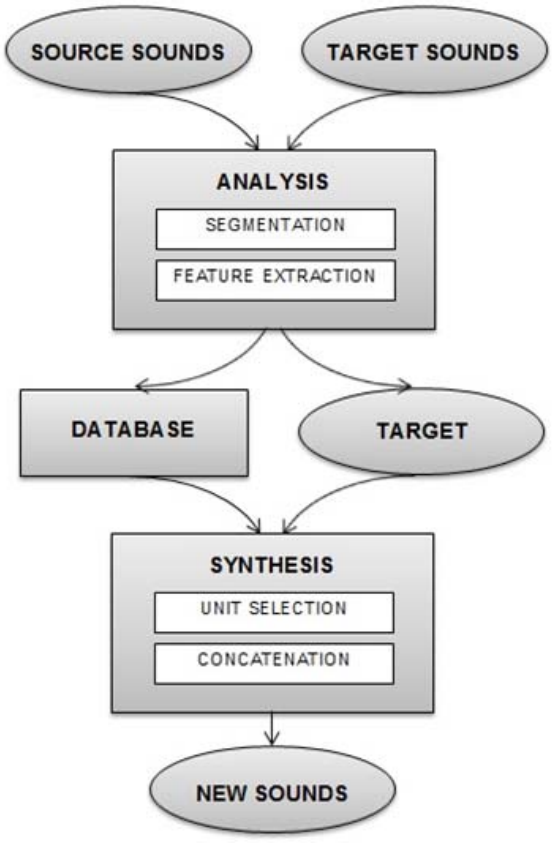

Figure 2. Flowchart of the processes in a typical CSS system

The paper is structured as follows. Section 2 discusses related works in the CSS. The Issues and challenges with regards to homosonic segments in CSS system is provided in Section 3. Section 4 proposes a solution to handle the homosonic segments CSS system, via the use of concatenation distance. Section 5 details the experimental setup of the study and presents an evaluation of the solution proposed. Section 6 concludes the study and lays out future direction of the work.

II. RELATED WORKS

Each of the segmented unit in the target sound file has unique characteristics that can be extracted. These features can be generated from the audio signal, their spectral, acoustical, perceptual, instrumental, harmonic properties, or symbolic score [3] Features are normally extracted automatically in a process known as audio feature extraction – a process of computing a compact numerical representation that can be used to characterise a segment of audio [9]. Feature extraction can result in low-level audio features such as zero crossing rate, energy level, spectral centroid and pitch, and also high-level audio features which take into account the context and semantics of the file, i.e. information on the artist, released dates, genres, etc.

Usually, the use of one feature is not enough for any unique deductions to be made about a sound; therefore it is common that several features are combined into feature vectors. Feature vectors list all features for a single point in time. There has been many researches that are already and continually being conducted on the study of audio feature extraction itself, as it is the fundamental process in fields such as content-based audio retrieval [10],[11]; musical genre classification [9],[12],[13]; audio thumbnailing [14],[15]; and audio recognition [16]. The works on feature extraction are not only limited to western music, but to other forms of music across the world, as evident from the works involving the classification of Chinese folks songs, traditional Malaysian music and Indian popular music respectively [17],[18],[19].

Following feature extraction, an optimal match between a target unit and the source units in the database needs to be selected in a process referred to as unit selection. A match which can be found within the least amount of time and using the fewest possible resources is favoured. In some circumstances where a search method performs better than others, this might be due to its execution design. One popular choice, the basic brute force can guarantee an optimal solution, but this method carries a huge overhead that increases exponentially as the dataset size increases [20]. Therefore, this method is only suitable when the database is small and it is imperative that the most optimal solution is found.

A faster alternative to this is the Viterbi algorithm, which has already been used in several existing CSS systems [8],[21]. The Viterbi algorithm gives the best interpretation of the entire context and reduces computational complexity by using recursion [22]. It is good for solving ambiguity when the confidence level is low but it too, runs the risk of being too exhaustive. The K-Nearest Neighbour (KNN) is another popular search method that is used in the unit selection process of a few CSS systems [3],[5]. It is most advantageous when little prior knowledge is known about the distribution of the data, but strong consistency in the result is required. Although its algorithm is fairly simple and can be quickly computed, the dataset has to undergo fast training, which may take up additional time.

Descendant of the local search algorithms such as the adaptive search algorithm and the incremental search algorithm have also been used in several CSS systems such as *Musical Mosaicing* and *Ringomatic* respectively [23],[24]. Local search algorithms always return a solution even if it is far from optimal. In comparison to exact matching, approximated solutions can return different and very interesting results which can be appealing in music synthesis.

Whichever algorithm is used in the path to finding the optimal matching segment, a measure is similarity must be used to compare the distance between the target unit and the units in the database. The most common way to solve this is through the use of Euclidean distance [25] Based on the Pythagoras theorem, the Euclidean distance measures the straight line distance between two points. When multidimensional features are used, the Euclidean distance calculate the distance between two vector points, x and y, and is given in the equation (1) below, where x_j (or y_j) is the coordinate of x (or y) in dimension j.

$$d_{x,y} = \sqrt{\sum_{j=1}^{J} (x_j - y_j)^2} \quad (1)$$

There are many more search methods available that might be as useful for finding the match between the target unit and the source unit. However, it is clear that each of them is designed to carry out search in a slightly different manner. When options are made available, users can decide on which search method is most fitting, taking into account tradeoffs such as accuracy, speed and computational load.

III. HOMOSONIC SEGMENTS IN CSS

In a CSS system, unit selection is the stage which determines which segments will be selected to make the final concatenated sound. Normally, the process is straightforward, where the system scans the database for a source segment that most closely matches the specified criteria (i.e. audio features) of the target segment. However, if the database is really large, several source segments which equally satisfy the criteria set by the target segment may become available. These segments are by no means redundant segments, but are in fact, different sounds that happen to be represented by the same sonic information with one another (Figure 3).

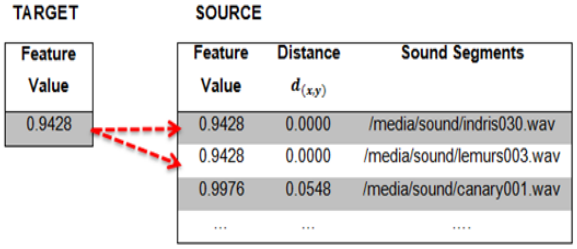

Figure 3. Example of homosonic segments of the source segments during unit selection

Homosonic audio segments are audio segments which possess the same sonic information, but are acoustically and physically different. Homosonic audio can be likened to the term 'Homograph' in the linguistic sense, where it is defined as a word that shares the same written form as another word but has a different meaning, and when spoken, the meanings may be distinguished by different pronunciations. Likewise, homosonic sounds are audio that may be represented to have the same sonic properties with each other, but do not sound the same when played. This can happen when the use of only one (or very few audio) features is compared, and the sound segments may appear to have identical values for these features. Only when additional features are revealed that it becomes apparent that the two sounds have different audio signal make up. For example, two homosonic sounds may be carry the same values when the intensity level is compared, but when played, both sounds are very different timbrally. This happened because in this case, the timbral information had not been included in the initial comparison.

In such situation, two most common solutions are practiced in existing CSS systems: (1) to select the source segment that appears on the top of the list; or (2) to randomly select any of the segments that have the same sonic information. The former solution presents noticeable weaknesses, the most obvious being the tendency to select only the first matching source segment that appears in the list of possible solutions, disregarding other equally qualified segments. Since the list is typically arranged

alphabetically, source segments represented with the filename that begins with letters that are further down the alphabetical order are almost never selected, unless a 'taboo list' function or selection without replacement is enabled. The flaw is even more intensified when there are several segments in the target segment that occur more than once, which can give way to a very tediously repetitive sound. The latter solution reduces the chances of re-selecting the first line of segments in the list of matching units, but the randomness of this process suggests that there is very little intelligence or reasoning behind the selection. Thus, a more intelligent solution to overcome the issue surrounding homosonic segments in CSS is needed.

IV. CONCATENATION DISTANCE

Concatenation distance is another 'cost' that is sometimes measured in addition to the target distance during the unit selection stage in concatenative sound synthesis. Whilst the target distance measures the similarity or closeness between the target unit and the source unit in the database, the concatenation distance measures the quality of the join between two consecutive units. This is why the concatenation distance is interchangeably referred as the join cost. The relationship between the target distance and the concatenation distance is illustrated in Figure 4.

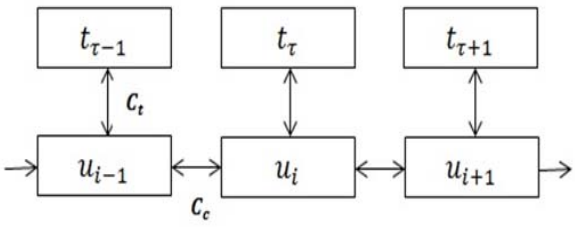

Figure 4. The relationship between target cost C_t and concatenation cost C_c

Where the target distance, C_t compares the feature value between the target segment ($t_{\tau-1}$) and the source segment (u_{i-1}), the concatenation distance, C_c compares the feature value at the beginning of a current segment (u_i) with the feature value at the end of a preceding segment (u_{i-1}), a working example of which is presented more clearly in Figure 23 below. If there are more than one audio features involved in the comparison, weights may be assigned to each feature. Rationally, if u_{i-1} and u_i are consecutive units in the source sound database, then their concatenation cost is equal to zero.

The concatenation cost, C_c can be calculated as follows,

$$C_c(u_{i-1}, u_i) = \sum_{j=1}^{q} w_j^c C_j^c (u_{i-1}, u_i) \quad (7)$$

where i is the current unit, w_j^c is the weighted sum of concatenative sub-costs (if and when applicable), which is denoted by C_j^c (u_{i-1}, u_i), (j=1,…, q), where q is the number sub-costs included, (i.e. if at the point of concatenation, the pitch and cepstral distance are used, then q = 2).

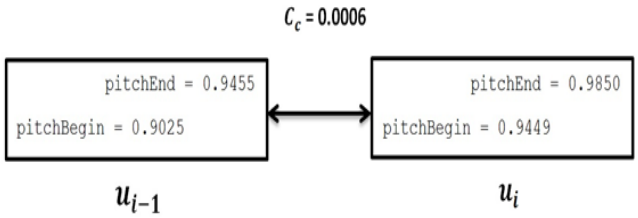

Figure 5. Calculating concatenation cost, C_c

Concatenation distance is first seen used as a measure to reduce segmental mismatch in concatenative speech synthesis that tends to occur at unit boundaries [26]. By selecting adjoining segment with the least distance from the previous segment, the naturalness of the utterance is enhanced, as seen implemented in several well-known concatenative speech synthesis system such as CHATR [6]

The same concept is adapted in several CSS systems, notably *Caterpillar* and *Musical Mosaic*. The use of concatenation distance in general is intended to reduce discontinuity between two adjoining segments, ensuring that the sounds are generated with a smoother flow, although there are some cases where this general rule is overridden, for instance, *Caterpillar* has an added function where it allows certain discontinuity to during an attack and not during a sustain unit. It is therefore hypothesised that the same method will be able to intelligently tackle the issues surrounding homosonic segments in CSS.

V. EVALUATION

To examine the feasibility and efficiency of concatenation distance in overcoming problems caused by homosonic segments and how this affected synthesis result, a bench mark test was conducted. The idea was to determine if the system was able to locate and select the exact same segments as queried through the target segments, if all of the segments that make up the target sounds were available in the source segment database. For this to happen, both target and source sounds used were the same one, which was the

Country file, and another sound file, *Classical* was added into the source sound database to produce the homosonic segments effect. Centroid was the audio features used to match the target and source sounds, and the onset mode was selected for segmentation.

The distribution of the homosonic segments contained in the dataset for this test is given in Figure 6. From the chart, it can be seen that out of the 40 segments of the queried target sound, 27 of them had at least two homosonic segments with equal potential being selected (e.g. Target Segment #2 had 3 homosonic segments to choose from, whilst Target Segment #6 had 5 homosonic segments to choose from, etc.). This not only displays the distribution of the homosonic segments in this test set, but also reinforces the point that a solution is needed to handle unit selection involving homosonic distances, as it is a common very occurrence, as demonstrated here.

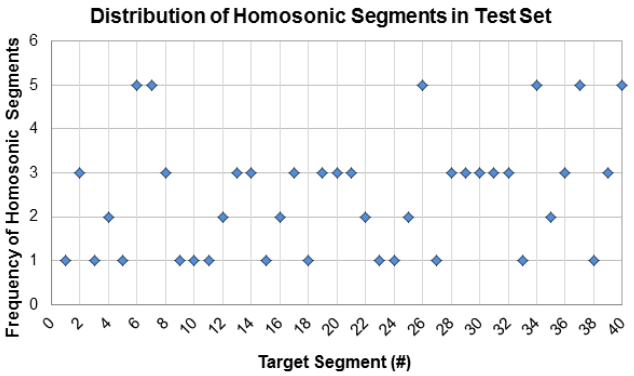

Figure 6. Distribution of homosonic segments in this study set

Since the same target distance was expected between all homosonic segments, the average target distance was not measured in this test, but was replaced with the concatenation distance (to observe the smoothness or flow of the sound at the joint between the segments), as well as the result accuracy (i.e. the ability to correctly select the right target segment) between the concatenation distance-enabled mode and the concatenation distance-disabled mode. The run-time between the two modes was also measured.

The result of the experiment in Figure 7 shows that there is no difference seen between them regarding the target distance. This was expected when homosonic segments were present. On the other hand, the concatenation distance was significantly lowered when concatenation distance was enabled. This suggests that the performance of concatenation distance-enabled mode had managed to obtain better result (i.e. smoother sound flow).

The waveforms in Figure 8 further emphasises the result from this benchmark test. The top row is the waveform of the original target sound. The middle row is the waveform that resulted from the concatenation distance-enabled mode, whilst the waveform in the final row resulted from the concatenation distance-disabled mode. From the figure, it is evident that by enabling the concatenation distance mode, the system generated sound that was more similar to the original target than it had when the mode was disabled. These improvements did, however, occur at the expense of run-time cost, where concatenation distance-enabled mode took almost six times as long to run.

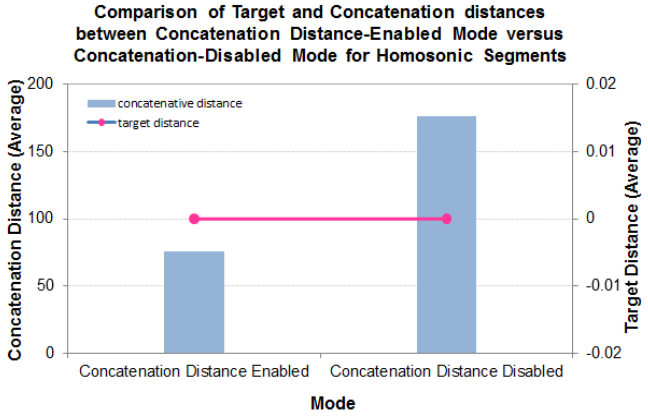

Figure 7. Comaprison of target distance versus concatenation distance

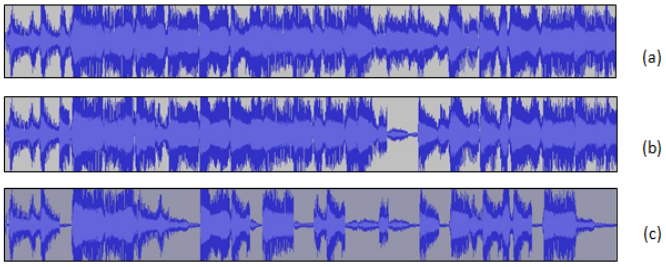

Figure 8. Waveform comparisons; (a) target query; (b) synthesis with concatenation distance enabled; and (c) synthesis with concatenation distance disabled

VI. CONCLUSION

This study has shown that concatenation distance can be used as a solution to overcome the challenges faced by the CSS system when challenged with homosonic segments. It is able to make a more intelligent decision over which source segments to select in the case where several of them possess the same target distance from the target segment. By comparing the concatenation distance of these equally fit segments, the selection is drawn through this second layer filtering. In addition to synthesising sounds with smoother

transitions from one segment to another (i.e. lower concatenation distance), this method is also capable of doing it through with high accuracy compared to when the concatenation distance mode is not in use. The only weakness of the concatenation distance-enabled mode was that it took longer for the sounds to be generated. This is understandable, given that in this particular dataset, almost three quarters of all the target segments had two or more homosonic segments. Occurrence of homosonic segments meant the concatenation distance needed to be calculated for each segment with the same sonic values, and after comparing these segments, the segment with the least concatenation distance was then selected.

Although, it is difficult to ascertain which of the sounds produced via the enablement or disablement sounded better, as it is a highly subjective and personal matter, at least by enabling the concatenation distance mode, the results have been improved numerically, as the system was able to select the intended segments 80% of the time, which is an impressive feat in itself.

Future work includes determining an algorithm which can calculate the concatenation distance faster, as this will allow the entire synthesis process to be conducted in real time as opposed to pre-calculation offline, which is the current practice.

REFERENCES

[1] Davies, M. E., Stark, A. M., Gouyon, F., & Goto, M. (2014). Improvasher: A Real-Time Mashup System for Live Musical Input. In NIME (pp. 541-544).

[2] Mohd Norowi, N. (2013). An Artificial Intelligence Approach to Concatenative Sound Synthesis.

[3] Schwarz, D. (2006). Concatenative sound synthesis: The early years. Journal of New Music Research, 35(1), 3-22.

[4] Sturm, B. L. (2004). MATConcat: an application for exploring concatenative sound synthesis using MATLAB. Proceedings of Digital Audio Effects (DAFx), Naples, Italy.

[5] Lazier, A., & Cook, P. (2003, September). MOSIEVIUS: Feature driven interactive audio mosaicing. In Digital Audio Effects (DAFx).

[6] Hunt, A. J., & Black, A. W. (1996, May). Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on* (Vol. 1, pp. 373-376). IEEE.

[7] Syrdal, A. K., Wightman, C. W., Conkie, A., Stylianou, Y., Beutnagel, M., Schroeter, J., & Makashay, M. J. (2000, October). Corpus-based techniques in the AT&T NextGen synthesis system. In *Proc. ICSLP* (Vol. 3, pp. 410-415).

[8] Schwarz, D. (2000, December). A system for data-driven concatenative sound synthesis. In *Digital Audio Effects (DAFx)* (pp. 97-102).

[9] Tzanetakis, G., & Princeton University. (2002). Manipulation, analysis and retrieval systems for audio signals.

[10] Zhang, T., & Kuo, C. (1998, July). Content-based classification and retrieval of audio. In SPIE's 43rd Annual Meeting-Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations VIII, San Diego.

[11] Guo, G., & Li, S. Z. (2003). Content-based audio classification and retrieval by support vector machines. *Neural Networks, IEEE Transactions on*, *14*(1), 209-215.

[12] Klapuri, A. (2004). Signal processing methods for the automatic transcription of music. Finland: Tampere University of Technology.

[13] Mierswa, I., & Morik, K. (2005). Automatic feature extraction for classifying audio data. *Machine learning*, *58*(2-3), 127-149.

[14] Bartsch, M. A., & Wakefield, G. H. (2001). To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the* (pp. 15-18). IEEE.

[15] Müller, M., Kurth, F., & Clausen, M. (2005). Audio matching via chroma-based statistical features. *Proc. ISMIR, London, GB*, 288-295.

[16] Eronen, A., & Klapuri, A. (2000). Musical instrument recognition using cepstral coefficients and temporal features. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on* (Vol. 2, pp. II753-II756). IEEE.

[17] Toiviainen, P., & Eerola, T. (2001). A method for comparative analysis of folk music based on musical feature extraction and neural networks. In *III International Conference on Cognitive Musicology* (pp. 41-45).

[18] Norowi, N. M., Doraisamy, S., & Wirza, R. (2005, September). Factors affecting automatic genre classification: an investigation incorporating non-western musical forms. In *Proceedings of the International Conference on Music Information Retrieval* (pp. 13-20).

[19] Ujlambkar, A. M. (2012). Automatic Mood Classification of Indian Popular Music.

[20] Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, *27*(1), 97-109.

[21] Maestre, E., Ramírez, R., Kersten, S., & Serra, X. (2009). Expressive concatenative synthesis by reusing samples from real performance recordings. *Computer Music Journal*, *33*(4), 23-42.

[22] Forney Jr, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268-278.

[23] Zils, A., & Pachet, F. (2001, December). Musical mosaicing. In *Digital Audio Effects (DAFx)*.

[24] Aucouturier, J. J., & Pachet, F. (2002, October). Music similarity measures: What's the use. In *Proceedings of the 3rd International Symposium on Music Information Retrieval* (pp. 157-163).

[25] Gower, J. C. (1985). Properties of Euclidean and non-Euclidean distance matrices. *Linear Algebra and its Applications*, *67*, 81-97.

[26] Pantazis, Y., Stylianou, Y., & Klabbers, E. (2005). Discontinuity detection in concatenated speech synthesis based on nonlinear speech analysis. In *Proc. de Eurospeech* (pp. 2817-2820).