A Modified Rough Longest Common Subsequence Algorithm for Motif Spotting in an Alapana of Carnatic Music

Shrey Dutta

Computer Science and Engineering Department
Indian Institute of Technology Madras
Chennai, India
Email: shrey@cse.iitm.ac.in

Hema A. Murthy

Computer Science and Engineering Department
Indian Institute of Technology Madras
Chennai, India
Email: hema@cse.iitm.ac.in

***Abstract*—Melody in Carnatic Music is based on the concept of a *raga*. Characteristic motif is a signature melodic phrase that determines the identity of a raga. This paper addresses locating these motifs in an *alapana*, given a query motif. A two-step approach approach is employed. The first is to identify the location of snippets of phrases that exists in the Alapana. Next the conventional approach to the rough longest common subsequence algorithm to spot motifs is modified. In the modified algorithm a more appropriate measure of match is proposed. The algorithm is particularly relevant for short but good matches. The modification enables logical choices of empirical parameters. In this paper, the motifs that are chosen are also long compared to our earlier work. The modified RLCS algorithm reduces the number of false alarms from 130 to 94 and also performs better localization of motifs.**

I. Introduction

Motif Spotting for music is an information retrieval task that locates occurrences of a query motif in a long music segment. The query motif in general is very short compared to that of the test music segment. Several factors need to be considered when dealing with this problem, namely: selection of features, time complexity, tolerance to noise, tolerance to speed variation, allowing partial matches or rough matches rather than exact matches, timbre, etc [1] [2] [3] [4]. The genre of music is also important, namely, western, hindustani, carnatic, chinese, andulasian, turkish, etc [5] [6] [7].

In this paper, the objective is to spot motifs in Carnatic music. In Carnatic music, the motifs are laden with $gamakas$ [8]. In addition, the motifs are similar across musicians but not necessarily identical. The duration of the motifs can also vary quite significantly although the rhythm may be preserved.

There are a number of dynamic programming techniques namely, dynamic time warping (DTW), longest common subsequence (LCS) and their variants, which are used for similar music matching tasks. DTW takes care of the speed variations due to warping but forces the match from end-to-end of both the query and the test. Even unconstrained endpoint DTW will align an entire query with a part of the test sequence [9]. In motif-spotting, there can be instances where one can expect that most of the query is roughly matched with a part of the test. Although LCS does not force the match between query and test to be end-to-end, it does not give importance to local similarity. There is a variant of LCS, known as rough-LCS [1], which address the issue of local similarity match where some leeway is given for partial query matches, where the degree of match at every time instant is accounted for. Other than partial query matches, when the characteristic motif, for example, "Sa Ni Da Pa Da" is rendered as "Sa Ri Ni Da Pa Da", RLCS gives a good match since it gives the longest matched subsequence. In an earlier work on spotting motifs in Carnatic music [3], a two-pass RLCS algorithm was used. It was observed that although the number of true positives was high, the number of false alarms was also high. Two other issues were identified in the algorithm used in [3]: i) A raga in Carnatic Music is made up of a concatenation of phrases. It was observed that motif matches were found across phrases. ii) The length of the motif was rather short.

The motif spotting in this paper is also performed on an extempore enunciation of a $raga$ in Carnatic music. An $Alapana$ is like poetry , where the Alapana can be broken up into phrases. As indicated in Figure 1, the phrases are approximately the same length and can be indicative of the laya that exists in an Alapana. This property in conjunction with a modified version of the RLCS algorithm reported in [3] is used to arrive at a significantly improved result. Further, a longer motif is chosen.

The paper is organized as follows. In Section II, some terminologies with respect to Carnatic music are discussed. In Section III, the selected features are briefed. In Section

978-1-4799-2361-8/14/$31.00

IV,the RLCS algorithm is discussed and in Section V the modifications to the RLCS algorithm are discussed. In Section VI, the two pass dynamic programming approach used to spot the motifs is discussed. Section VII, describes the dataset used in this study which is same as that used in [3] with a difference. The difference is that longer motifs are chosen. The study is also restricted to the raga Bhairavi. In Section VIII the results are presented and analyzed. Section IX is the discussion section. The conclusions and future work are given in Section X.

II. Carnatic Music, Svaras and Gamakas

Melody in Carnatic music is based on *rāga*. This is very different from the western concepts of tones and scales. Most Indian instruments do not have a specific tuning. Carnatic music is related to just intonation rather than equal temperament [10]. It is primarily an oral tradition where notation has a very little role. A *rāga* in Carnatic music is made of a set of inflected notes called *svaras*. A single *svara* does not represent just a single frequency but a band of frequencies which is referred to as intonation of the *svara*. It has been conjectured by musicians that a quantization of the pitch on the bases of the absolute frequencies of *svaras* is erroneous [8]. This makes motif spotting in Carnatic music a hard problem.

A motif when rendered in isolation must unambiguously correspond to the template of a *rāga*. In the earlier paper [3], short motifs of various *rāga*s were analyzed. This resulted in a large number of false alarms when the motifs were used as queries.

In this paper, we choose longer motifs. These motifs are inspired by the "*rāga* test" [11]. Most people across the globe [12] were able to unambiguously determine the identity of *rāga*s using these motifs. An attempt was made to use the motifs from [11] directly. As the recordings are rather noisy, the same motifs were generated by a professional musician. In particular, we have chosen only the *rāga* "bhairavi" for illustration. In [11] it is observed that there are four motifs that are rendered for the *rāga* bhairavi. From Figure 1, it is clear that the motifs are separated by short pauses. It is also observed that the motifs are more or less of equal length. In this paper, we consider this property as the rhythm in which the *alāpana* is sung. The *alāpana* is first segmented into these snippets.The motif which is a signature phrase of the raga is searched for in these snippets.

To identify the rhythm, we perform voice activity detection (VAD) on the *Alāpana*. The begin and end points of a voiced region are used for motif spotting. The pitch extracted is used as a voice activity detector. The pitch extraction results in zero values of pitch wherever there is a pause. Pitch errors also occur. The pitch errors in terms of octave errors are ignored. A threshold is set on the duration of the unvoiced region and voiced regions to eliminate very short unvoiced and voiced regions.

Fig. 1: Bhairavi motifs seperated by short pauses.

III. Feature Selection

In the previous work [3], the importance of using pitch as a feature for Carnatic music was discussed. Pitch extracted, using the melodia pitch extraction algorithm [13], is used as the basic feature in this work as well. The preprocessing steps are identical to that of [3] with a subtle difference. For completeness, the steps are briefly reviewed.

The basic pitch contour is processed to get the tonic normalized smoothed pitch contour. The processing steps are stated as follows:

1) *Tonic Normalization*: Tonic for various concerts and musicians are different. Thus, tonic needs to be identified and the pitch contour needs to be normalized with this tonic.Techniques developed in [14] [15] are used to estimate tonic. The normalization of the pitch contour with respect to tonic is performed using Equation 1.

$$centFrequency = 1200 \cdot log_2 \left(\frac{f}{tonic} \right) \quad (1)$$

2) *Saddle Point Analysis*: Saddle points are stationary points of a pitch contour where the first derivative is zero. They preserve a lot of information from the pitch contour as stated in [3].
3) *Spline interpolation for smoothing*: The pitch contour is smoothed using cubic spline interpolation on the saddle points.

The tonic normalized smoothed pitch contour and its saddle points are used as features in this work. The RLCS and modified RLCS algorithms are used in two passes: using saddle points in the first pass to reduce the search space and then using the entire tonic normalized smoothed pitch contour on this reduced search space in the second pass.

IV. Rough Longest Common Subsequence

Rough Longest Common Subsequence, a variant of Longest Common Subsequence, performs an approximate match between reference and query while retaining the local similarity [1]. It introduces three major changes in LCS namely, rough match, width-across-reference (WAR) and width-across-query (WAQ) for local similarity and score matrix.

A. Rough match

In the recurrence function of LCS, the cost function is incremented by 1 when there is an exact match. In RLCS, when the distance between a reference point, r_i, and a query point, q_i, is less than a threshold, T_d, they are said to be roughly matched, $r_i \approx q_i$ i.e. $d(r_i, q_j) < T_d \rightarrow r_i \approx q_i$, where $d(r_i, q_i)$ is the distance between r_j and q_j. The cost is incremented by a number, δ, between 0 and 1 instead of 1, based on how good the match is as shown in Equation 2.

$$\delta_{i,j} = 1 - \frac{d_{r_i,q_j}}{T_d} \tag{2}$$

The cost is estimated using the following recurrence:

$$c_{i,j} = \begin{cases} 0 & ; i.j = 0 \\ c_{i-1,j-1} + \delta_{i,j} & ; r_i \approx q_j \\ \max(c_{i-1,j}, c_{j-1,i}) & ; r_i ! \approx q_i \end{cases} \tag{3}$$

In LCS the cost gives the length of the longest common subsequence. The cost of RLCS is not incremented by 1 but it represents the length of the rough longest common subsequence. Later, it is argued that this length is actually a rough length of RLCS rather than its actual length.

B. WAR and WAQ for local similarity

To retain the local similarity, width-across-reference, WAR, and width-across-query, WAQ, are used. WAR and WAQ represent the length of the shortest substring of the reference and the query respectively, containing the LCS. These measures represent the density of LCS in the reference and the query. Small values of WAR and WAQ indicate a dense distribution of LCS. WAR is incremented by 1 if there is a rough match or jump along the reference. Likewise for the WAQ. WAR and WAQ are computed using the following recurrences:

$$w^r_{i,j} = \begin{cases} 0 & ; i.j = 0 \\ w^r_{i-1,j-1} + 1 & ; r_i \approx q_j \\ w^r_{i-1,j} + 1 & ; r_i ! \approx q_i,\ c_{i-1,j} \geq c_{i,j-1} \\ w^r_{i,j-1} & ; r_i ! \approx q_i,\ c_{i-1,j} < c_{i,j-1} \end{cases} \tag{4}$$

$$w^q_{i,j} = \begin{cases} 0 & ; i.j = 0 \\ w^q_{i-1,j-1} + 1 & ; r_i \approx q_j \\ w^q_{i-1,j} & ; r_i ! \approx q_i,\ c_{i-1,j} \geq c_{i,j-1} \\ w^q_{i,j-1} + 1 & ; r_i ! \approx q_i,\ c_{i-1,j} < c_{i,j-1} \end{cases} \tag{5}$$

In Equations 4 and 5, some of the cases and conditions are dropped from [1] for the sake of clarity.

C. Score matrix

WAR, WAQ and cost are used to compute the score of a common subsequence in the following way:

$$Score_{i,j} = \begin{cases} \left(\beta \frac{c_{i,j}}{w^r_{i,j}} + (1-\beta)\frac{c_{i,j}}{w^q_{i,j}}\right) \cdot \frac{c_{i,j}}{n} & if\ c_{i,j} \geq \rho n \\ 0 & otherwise \end{cases} \tag{6}$$

In Equation 6, larger value of $\frac{c_{i,j}}{w^r_{i,j}}$ suggests that the density of the RLCS is high in the reference. Similarly, larger value of $\frac{c_{i,j}}{w^q_{i,j}}$ is indicative of higher density of RLCS in the query. β weighs between these two ratios. Larger value of $\frac{c_{i,j}}{n}$ indicates that a large part of the query has been matched, where n is the length of the query. ρ is the matching rate that represents how long the length of the RLCS should be, with respect to the query.

The algorithm to compute these values using Dynamic Programming is presented in [1].

V. Modified-RLCS

In this section, the modifications made to the existing RLCS algorithm and the rationale behind them are discussed.

A. Rough and actual length of RLCS

In [1], $c_{i,j}$ is defined as the length of the RLCS. But, it actually represents a rough length of RLCS because it is incremented by $\delta_{i,j}$ when there is a rough match. The resulting value of $c_{i,j}$ need not be an integer. Therefore, it cannot be the actual length of any sequence. The actual length of RLCS is defined by the following recurrence:

$$c^a_{i,j} = \begin{cases} 0 & ; i.j = 0 \\ c^a_{i-1,j-1} + 1 & ; r_i \approx q_j \\ \max(c^a_{i-1,j}, c^a_{j-1,i}) & ; r_i ! \approx q_i \end{cases} \tag{7}$$

Instead of just considering how good the rough length of the RLCS with respect to the query length is, it is conjectured that it is important to consider how good the rough length of the RLCS is with respect to the actual length of the RLCS.

$$\frac{c_{i,j} + c_{i,j}}{c^a_{i,j} + n}$$

gives equal importance to both the ratios. This term is similar to the F1 score where precision and recall are given equal importance.

B. RWAR and RWAQ

WAR and WAQ represent the width of the shortest substring that contains the RLCS. As discussed in the previous subsection, $c_{i,j}$ represents the rough length which is shorter than the actual length of the RLCS. Therefore, it is not clear whether $\frac{c_{i,j}}{w^r_{i,j}}$ really represents the density of the RLCS in the reference. This term also penalizes based on the degree of match, while a penalty has already been accounted for in

the term, $\frac{c_{i,j}}{n}$. Therefore, some rough width across reference, RWAR, and rough width across query, RWAQ, are required that represent the rough width of the shortest substring containing the RLCS. On a rough match, cost is incremented by a value between 0 and 1 based on how good the match is. At the same time, when a rough match is obtained, the WAR and WAQ are also incremented by the same value resulting in RWAR and RWAQ, respectively. When there is no match, RWAR and RWAQ are incremented by 1 whereas the cost is not incremented. Therefore, RWAR and RWAQ account for the density of the RLCS in the reference and query better. RWAR and RWAQ can be computed by the following recurrences:

$$rw^r_{i,j} = \begin{cases} 0 & ; i.j = 0 \\ rw^r_{i-1,j-1} + \delta_{i,j} & ; r_i \approx q_j \\ rw^r_{i-1,j} + 1 & ; r_i! \approx q_i,\ c_{i-1,j} \geq c_{i,j-1} \\ rw^r_{i,j-1} & ; r_i! \approx q_i,\ c_{i-1,j} < c_{i,j-1} \end{cases} \tag{8}$$

$$rw^q_{i,j} = \begin{cases} 0 & ; i.j = 0 \\ rw^q_{i-1,j-1} + \delta_{i,j} & ; r_i \approx q_j \\ rw^q_{i-1,j} & ; r_i! \approx q_i,\ c_{i-1,j} \geq c_{i,j-1} \\ rw^q_{i,j-1} + 1 & ; r_i! \approx q_i,\ c_{i-1,j} < c_{i,j-1} \end{cases} \tag{9}$$

C. Matched rate on the query sequence

In Equation (6), ρ is an empirical parameter that is set based on the required match rate on the entire query sequence. The score is updated by a non-zero value, only if the rough length of the RLCS is greater than ρn. It is not clear how to set the value of ρ or what it means for the rough length to be greater than a fraction of the query length. Instead, it would be better to update the score by a non-zero value if the actual length is greater than ρn. This makes the interpretation clear and makes it easy to set the value of ρ.

The score update of the modified-RLCS is given by the following equation:

$$Score_{i,j} = \begin{cases} \left(\beta \frac{c_{i,j}}{rw^r_{i,j}} + (1-\beta)\frac{c_{i,j}}{rw^q_{i,j}}\right) \cdot \frac{c_{i,j}+c_{i,j}}{c^a_{i,j}+n} & if\ c^a_{i,j} \geq \rho n \\ 0 & otherwise \end{cases} \tag{10}$$

VI. A Two-Pass Dynamic Programming search

Two passes of modified-RLCS are used for the task of motif spotting. In the first pass, saddle points of the query motif and test Alapana are used. This reduces the search space and gives the potential motif regions, each represented as a group of overlapping sequences. These overlapping sequences are obtained by performing modified-RLCS by sliding a window on the test Alapana. In the second pass, the entire tonic normalized smoothed pitch contours of the query motifs and the test groups are used. This second pass is required to locate the motifs within each group. The entire procedure is detailed in [3].

Since the first pass is performed using the saddle points, it is very fast compared to the second pass which is performed using the entire pitch contour. In section IX, we also discuss situations where a second pass may not be required.

VII. Dataset

The motifs selected are 3.18 seconds long on an average. The average duration is calculated from the ground truth marked by an expert. These motifs are longer than the ones used in the previous work [3]. The details of the query motifs are given in Table I.

TABLE I: Motifs Queried

rāga Name	Number	Average Duration (seconds)
Bharavi	59	3.18

The experiments are performed on 16 *Alapanas* of *rāga* Bhairavi. The details of the test *Alapanas* are given in Table II. The number of *Alapanas* are less than what is used in the previous work [3]. This is primarily because we have only worked on Alapanas corresponding to vocal recordings.

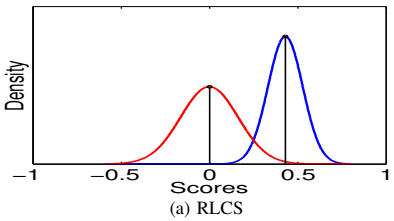
TABLE II: Database of *Alapanas*; N-Al: Number of *Alapanas*; N-Art: Number of Artists; Avg-Dur: Average Duration; Tot-Dur: Total Duration

rāga Name	N-Al	N-Art	Avg-Dur (mins)	Tot-Dur (mins)
Bharavi	16	13	10.65	170.48

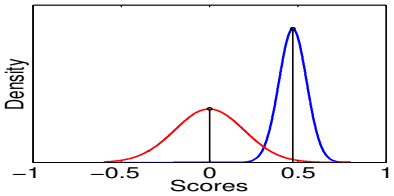
VIII. Experiments And Results

Motif spotting is performed using RLCS and modified-RLCS using the database shown in Table II. The distance function used is a cubic distance function which is the distance function used in [3]. T_d is set to 0.45 in both the methods. ρ is set to zero because the best value of ρ could be different for both the methods which makes the comparison difficult.

In this work, the phrases sung across octaves are not considered. First VAD is performed on the *Alapanas* to get the voiced parts. This segments the Alapana into phrases approximately. It has been observed that these phrases are more or less of equal length. This is what is meant by rhythm in this paper, although no effort is made to quantify the rhythm. Instead of the entire *Alapana*, these voiced regions are used. In the first pass, saddle points of the query motif and test *Alapana* are used and the motif regions or groups are retrieved along with their scores. Each group either corresponds to a motif or a false alarm. A true group consists of one or more true positives. Score distribution of the true positive groups and false alarm groups for both the algorithms are shown in Figure 2. Each score value is subtracted from the mean of scores of the false alarms such that mean of the false alarms' distribution becomes zero for both the algorithms. This enables a better comparison between RLCS and modified-RLCS algorithms. The shared region in the score distribution of true positives groups and false alarms groups is less in the modified-RLCS algorithm than in the RLCS algorithm.



(a) RLCS



(b) Modified-RLCS

Fig. 2: True positive groups' and false alarm groups' score distribution for RLCS and modified-RLCS. The shared reg ion in the score distribution of true positive groups and false alarm groups is less in the modified-RLCS algorithm than in the RLCS algorithm.

Motifs are sparsely present in an *Alapana*. Our purpose is to retrieve as many motifs as possible. Spotting all or most of the motifs is more crucial than removal of all false alarms. Therefore higher penalty is given for missing a motif than for a false alarm group. Score threshold is selected from the minimum detection cost function for both the algorithms. The sequences whose scores are above the score threshold are preserved. The number of true positive groups, true positives, false alarm groups and their average durations in these preserved sequences are shown in Table III. Modified-RLCS has shown a clear improvement over RLCS in terms of false alarms and average duration of true positive groups and false alarm groups.

TABLE III: Retrieved groups after the first pass; NTG: Number of True Positive Groups; NTP: Number of True Positives retrieved; TAD: Average Duration of the True Positive Groups in seconds; NFG: Number of False Alarm Groups; FAD: Average Duration of the False Alarm Groups in seconds.

Algorithm	NTG	NTP	TAD	NFG	FAD
RLCS	51	53	9.61	143	12.73
Modified-RLCS	52	52	9.00	99	11.95

These preserved sequences (groups) are used as the tests in the second pass. In the second pass, tonic normalized smoothed pitch contour is used as a feature. The primary objective of the second pass is to locate the motifs within a group and remove as many false alarm groups as possible. The number of true positive groups, true positives, false alarm groups and their durations, after the second pass, is given in Table IV. The score threshold value is selected such that all the hits corresponding to the motifs are retrieved. There is a reduction in the number of false alarms in both the algorithms.

Modified-RLCS results in less false alarms compared to RLCS in both the passes. The duration of the hits is also less as compared to RLCS. This is primarily due to the fact that the actual length of the match is used in the modified-RLCS. In Equation (6) of the RLCS algorithm, the term

$$\frac{c_{i,j}}{n}$$

focuses on getting an RLCS whose rough length is as large as length of query motif but in Equation (10) of modified-RLCS, the term

$$\frac{c_{i,j} + c_{i,j}}{c^a_{i,j} + n}$$

gives equal importance to both: getting an RLCS whose rough length is as large as the length of the query motif and also on getting an RLCS whose rough length is as large as the actual length of the RLCS. Due to this, shorter sequences also get a good score if they represent the motif adequately. This also results in better localization of motifs.

TABLE IV: Retrieved groups after the second pass; NTG: Number of True Positive Groups; NTP: Number of True Positives retrieved; TAD: Average Duration of the True Positive Groups in seconds; NFG: Number of False Alarm Groups; FAD: Average Duration of the False Alarm Groups in seconds.

Algorithm	NTG	NTP	TAD	NFG	FAD
RLCS	50	51	8.25	130	11.12
Modified-RLCS	50	50	7.68	94	11.00

IX. Discussions

A. Importance of VAD in motif spotting

Voice-activity-detection on the Alapanas is a very crucial step. This step removes the noise and reduces the search space. Table V shows the results after the two passes using the modified-RLCS algorithm. The method for selecting the score thresholds after both the passes remains the same. The groups have become much longer though the number of true positive groups and false alarm groups has reduced. Some of the true positive groups have more than one instance of the motif. Therefore, the number of true positives are much more than the number of true positive groups. But they are not at all localized properly even after the second pass. The number of false alarm groups is also very less but the duration is very high, approximately 1 minute. The total duration of the false alarms without VAD is much more than that of those with VAD after each of the passes. This vindicates the use of VAD. Given that the voiced regions are approximately the same duration, VAD could also be used for estimating the rhythm in an Alapana. At this juncture it is difficult to quantify rhythm in the Alapana. Clearly VAD has helped but in some situations it is observed that there is more than one true positive in some groups. This needs careful analysis.

TABLE V: Retrieved Groups after both the passes for modified-RLCS without VAD; NTG: Number of True Positive Groups; NTP: Number of True Positives retrieved; TAD: Average Duration of the True Positive Groups in seconds; NFG: Number of False Alarm Groups; FAD: Average Duration of the False Alarm Groups in seconds.

Pass No.	NTG	NTP	TAD	NFG	FAD
Pass 1	29	57	80.31	20	68.23
Pass 2	38	57	44.33	31	45.45

B. Is the second pass really required after VAD?

Modified-RLCS took around 10 minutes in the first pass to test all the *Alapanas* on a 16 core machine. Since the first pass

uses the saddle points, it is reasonably fast. But, the second pass takes around 1.15 hours to run on the same machine. Since the entire pitch contour is used, the computation time is considerably large. There is hardly any improvement in-terms of the reduction in the number of false alarm groups though the duration of the hits have reduced a little in the second pass. Therefore, if VAD is performed, one can stop after the first pass as well. As shown in Table V, second pass makes a significant difference in terms of both, number and duration of hits, when VAD is not performed.

X. Conclusion And Future Work

In this work, a modified version of RLCS algorithm is presented which gives better scores for sub-sequences which are shorter than the query but have matched reasonably well with most of the query. We have also exploited the rhythm in which an *alapana* is sung to segment it into snippets of phrases. Modified-RLCS was tested on longer motifs and compared favorably with the original RLCS.

We conjecture that inter-saddle point duration may also be important. This will result in a further reduction in the search space. This inter-saddle point duration must be normalised to account for the overall rhythm of a motif which can be sung in different speeds. Motifs across octaves need to be analyzed. More changes need to be made in the RLCS algorithm to find multiple non-overlapping sub-sequences between two *alapanas* and thus, enable automatic discovery of typical motifs. Ultimately, it should be possible to build an application that can automatically analyze and extract motifs online. The motifs can then be used to generate fine indexes for an *Alapana*.

Acknowledgment

This research was partly funded by the European Research Council under the European Unions Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583). The authors would like to thank Raghava Krishnan K. for his invaluable suggestions in completing this paper and Vignesh Ishwar for doing the annotations and recordings.

References

[1] H.-J. LIN, H.-H. WU, and C.-W. WANG, "Music matching based on rough longest common subsequence," *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, pp. 27, 95–110., 2011.

[2] A. Guo and H. Siegelmann, "Time-warped lomgest common subsequence algorithm for music retrival," *In Proc. 5th International Conference on Music Information Retrieval (ISMIR)*, 2004.

[3] V. Ishwar, S. Dutta, A. Bellur, and H. Murthy, "Motif spotting in an alapana in carnatic music," *In Proc. 14th International Conference on Music Information Retrieval (ISMIR)*, 2013.

[4] A. L. Uitdenbogerd and J. Zobel, "Manipulation of music for melody matching," *MULTIMEDIA '98 Proceedings of the sixth ACM international conference on Multimedia*, 1998.

[5] J. C. Ross and P. Rao, "Detection of raga-characteristic phrases from hindustani classical music audio," *Proc. of the 2nd CompMusic Workshop*, July 12-13, 2012.

[6] A. Krishnaswamy, "Inflexions and microtonality in south indian classical music," *Frontiers of Research on Speech and Music*, 2004.

[7] V. Ishwar, A. Bellur, and H. A. Murthy, "Motivic analysis and its relevance to raga identification in carnatic music," in *Workshop on Computer Music*, Instanbul, Turkey, July 2012.

[8] T. M. Krishna and V. Ishwar, "Svaras, gamaka, motif and raga identity," in *Workshop on Computer Music*, Instanbul, Turkey, July 2012.

[9] T. Giorgino, "Computing and visualizing dynamic time warping alignments in R: The dtw package," *Journal of Statistical Software*, vol. 31, no. 7, pp. 1–24, 2009. [Online]. Available: http://www.jstatsoft.org/v31/i07/

[10] J. Serra, G. K. Koduri, M. Miron, and X. Serra, "Tuning of sung indian classical music," *In Proc. of ISMIR*, pp. 157–162, 2011.

[11] R. Verma. Raga test. [Online]. Available: http://www.youtube.com/watch?v=3nRtz9EBfeY

[12] Rasikas. [Online]. Available: http://www.rasikas.org

[13] J. Salamon and E. Gmez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech and Language Processing*, pp. 20(6):1759–1770, Aug. 2012.

[14] A. Bellur and H. A. Murthy, "A cepstrum based approach for identifying tonic pitch in indian classical music," in *National Conference on Communications*, 2013.

[15] A. Bellur, V. Ishwar, X. Serra, and H. A. Murthy, "'a knowledge based signal processing approach to tonic identication in indian classical music," in *Workshop on Computer Music*, July 2012.