RAGA VERIFICATION IN CARNATIC MUSIC USING LONGEST COMMON SEGMENT SET

Shrey Dutta
Dept. of Computer Sci. & Engg.
Indian Institute of Technology Madras
shrey@cse.iitm.ac.in

Krishnaraj Sekhar PV
Dept. of Computer Sci. & Engg.
Indian Institute of Technology Madras
pvkrajpv@gmail.com

Hema A. Murthy
Dept. of Computer Sci. & Engg.
Indian Institute of Technology Madras
hema@cse.iitm.ac.in

ABSTRACT

There are at least 100 *rāgas* that are regularly performed in Carnatic music concerts. The audience determines the identity of *rāgas* within a few seconds of listening to an item. Most of the audience consists of people who are only avid listeners and not performers.

In this paper, an attempt is made to mimic the listener. A *rāga* verification framework is therefore suggested. The *rāga* verification system assumes that a specific *rāga* is claimed based on similarity of movements and motivic patterns. The system then checks whether this claimed *rāga* is correct. For every *rāga*, a set of cohorts are chosen. A *rāga* and its cohorts are represented using pallavi lines of compositions. A novel approach for matching, called Longest Common Segment Set (LCSS), is introduced. The LCSS scores for a *rāga* are then normalized with respect to its cohorts in two different ways. The resulting systems and a baseline system are compared for two partitionings of a dataset. A dataset of 30 *rāgas* from Charsur Foundation [1] is used for analysis. An equal error rate (EER) of 12% is obtained.

1 Introduction

Rāga identification by machine is a difficult task in Carnatic music. This is primarily because a *rāga* is not defined just by the solfege but by *svaras* (ornamented notes) [13]. The melodic histograms obtained for the Carnatic music are more or less continuous owing to the *gamakā* [2] laden svaras of the *rāga* [23]. Although the svaras in Carnatic music are not quantifiable, for notational purposes an octave is divided into 12 semitones: S, R1, R2(G1), R3(G2), G3, M1, M2, P, D1, D2(N1), D3(N2) and N3. Each *rāga* is characterised by atleast 5 svaras. *Ārohana* and *avarohana* correspond to an ordering of *svaras* in the ascent and descent of the *rāga*, respectively. Ragas with linear ordering of svaras are referred to as linear ragas such as *Mohonam rāga* (S R2 G3 P D2 S). Similarly, non linear ragas have non linear ordering such as *Ananda Bhairavi* raga (S G2 R2 G2 M1 P D2 P S). A further complication arises owing to the fact that although the *svaras* in different *rāgas* may be identical, the ordering can be different. Even if the ordering is the same, in one *rāga* the approach to the *svara* can be different, for example, *todi* and *dhanyasi*.

There is no parallel in Western classical music to *rāga* verification. The closest that one can associate with, is cover song detection [6, 16, 22], where the objective is to determine the same song rendered by different musicians. Whereas, two different renditions of the same *rāga* may not contain identical renditions of the motifs.

Several attempts have been made to identify *rāgas* [2–4, 7, 8, 12, 14, 26]. Most of these efforts have used small repertoires or have focused on *rāgas* for which ordering is not important. In [26], the audio is transcribed to a sequence of notes and string matching techniques are used to perform *rāga* identification. In [2], pitch-class and pitch-dyads distributions are used for identifying *rāgas*. Bigrams on pitch are obtained using a twelve semitone scale. In [18], the authors assume that an automatic note transcription system for the audio is available. The transcribed notes are then subjected to HMM based *rāga* analysis. In [12, 25], a template based on the *ārohana* and *avarohana* is used to determine the identity of the *rāga*. The frequency of the *svaras* in Carnatic music is seldom fixed. Further, as indicated in [27] and [28], the improvisations in extempore enunciation of *rāgas* can vary across musicians and schools. This behaviour is accounted for in [10, 11, 14] by decreasing the binwidth for computing melodic histograms. In [14], steady note transcription along with n-gram models is used to perform *rāga* identification. In [3] chroma features are used in an HMM framework to perform scale independent *rāga* identification, while in [4] hierarchical random forest classifier is used to match *svara* histograms. The *svara*s are obtained using the Western transcription system. These experiments are performed on 4/8 different *rāgas* of Hindustani music. In [7], an attempt is made to perform *rāga* identification using semi-continuous Gaussian mixtures models. This will work only for linear *rāgas*. Recent research indicates that a *rāga* is characterised best by a time-frequency trajectory rather than a sequence of

[1] http://www.charsurartsfoundation.org

[2] *Gamakā* is a meandering of a *svara* encompassing other permissible frequencies around it.

	Vocal		Instruments				Total
	Male	Female	Violin	Veena	Saxophone	Flute	
Number of Ragas	25	27	8	3	2	2	30 (distinct)
Number of Artists	53	37	8	3	1	3	105
Number of Recordings	134	97	14	4	2	3	254
Total Duration of Recordings	30 h	22 h	3 h	31 m	10 m	58 m	57 h
Number of Pallavi Lines	655	475	69	20	10	15	1244
Average Duration of Pallavi Lines	11 s	8 s	10 s	6 s	6 s	8 s	8 s (avg.)
Total Duration of Pallavi Lines	2 h	1 h	11 m	2 m	55 s	2 m	3 h

Table 1. Details of the database used. Durations are given in approximate hours (h), minutes (m) or seconds (s).

quantised pitches [5, 8, 9, 19, 20, 24]. In [19, 20], the *sama* of the tala (emphasised by the *bol* of tabla) is used to segment a piece. The repeating pattern in a bandish in Hindustani Khyal music is located using the sama information. In [8, 19], motif identification is performed for Carnatic music. Motifs for a set of five *rāgas* are defined and marked carefully by a musician. Motif identification is performed using hidden Markov models (HMMs) trained for each motif. Similar to [20], motif spotting in an *ālāpana* in Carnatic music is performed in [9]. In [24], a number of different similarity measures for matching melodic motifs of Indian music was attempted. It was shown that the intra pattern melodic motif has higher variation for Carnatic music in comparison with that of Hindustani music. It was also shown that the similarity obtained is very sensitive to the measure used. All these efforts are ultimately aimed at obtaining typical signatures of *rāgas*. It is shown in [9] that there can be many signatures for a given *rāga*. To alleviate this problem in [5], an attempt was made to obtain as many signatures for a *rāga* by comparing lines of compositions. Here again, it was observed that the typical motif detection was very sensitive to the distance measure chosen. Using typical motifs/signatures for *rāga* identification is not scalable, when the number of *rāgas* under consideration increases.

In this paper, this problem is addressed in a different way. The objective is to mimic a listener in a Carnatic music concert. There are at least 100 *rāgas* that are actively performed today. Most listeners identify *rāgas* by referring to the compositions with similar motivic patterns that they might have heard before. In *rāga* verification, a *rāga*'s name (claim) and an audio clip is supplied. The machine has to primarily verify whether the clip belongs to a given *rāga* or not.

This task therefore requires the definition of cohorts for a *rāga*. Cohorts of a given *rāga* are the ragas which have similar movements while at the same time have subtle differences, for example, *darbar* and *nāyaki*. In *darbar* raga, G2 is repeated twice in *avarohana*. The first is more or less flat and short, while the second repetition is inflected. The G2 in *nāyaki* is characterised by a very typical *gamakā*. In order to verify whether a given audio clip belongs to a claimed *rāga*, the similarity is measured with respect to the claimed *rāga* and compared with its cohorts using a novel algorithm called *longest common segment set* (LCSS). LCSS scores are then normalized using Z and T norms [1, 17].

The rest of the paper is organised as follows. Section 2 describes the dataset used in the study. Section 3 describes the LCSS algorithm and its relevance for *rāga* verification. As the task is *rāga* verification, score normalisation is crucial. Different score normalisation techniques are discussed in Section 4. The experimental results are presented in Section 5 and discussed in Section 6. The main conclusions drawn from the key results in this paper are discussed in Section 7

2 Dataset used

Table 1 gives the details of the dataset used in this work. This dataset is obtained from the Charsur arts foundation [3]. The dataset consists of 254 vocal and instrument live recordings spread across 30 *rāgas*, including both target ragas and their cohorts. For every new *rāga* that needs to be verified, templates for the *rāga* and its cohorts are required.

2.1 Extraction of pallavi lines

A composition in Carnatic music is composed of three parts, namely, *pallavi*, *anupallavi* and *caranam*. It is believed that the first phrase of the first *pallavi* line of a composition contains the important movements in a *rāga*. A basic sketch is initiated in the *pallavi* line, developed further in the *anupallavi* and *caranam* [21] and therefore contains the gist of the *rāga*. The algorithm described in [21] is used for extracting *pallavi* lines from compositions. Details of the extracted pallavi lines are given in Table 1. Experiments are performed on template and test recordings, selected from these pallavi lines, as discussed in greater detail in Section 5.

2.2 Selection of cohorts

Wherever possible 4-5 *rāgas* are chosen as cohorts of every *rāga*. The cohorts of every *rāga* were defined by a professional musician. Professionals are very careful about this as they need to ensure that during improvisation, they do not accidentally sketch the cohort. Interestingly, as indicated by the musicians, cohorts need not be symmetric. A *rāga* A can be similar in movement to a *rāga* B, but *rāga* B need not share the same commonality with *rāga* A. The identity of *rāga* B may depend on phrases similar to *rāga* A with some additional movement. For example,

[3] http://www.charsurartsfoundation.org

to identify the *rāga* Indolam, the phrase G2 M1 D1 N2 S is adequate, while Jayantashree *rāga* requires the phrase G2 M1 D1 N2 S N2 D1 P M1 G2 S.

3 Longest common segment set

In *rāga* verification, matching needs to be performed between two audio clips. The number of similar portions could be more than one and spread across the entire clip. Therefore, there is a need for a matching approach that can find these similar portions without issuing large penalties for gaps in between them. In this section, a novel algorithm called Longest Common Segment Set is described which attempts to do the same.

Let $X = \langle x_1, \cdots, x_m;\ x_i \in \mathbb{R};\ i = 1 \cdots m\rangle$ be a sequence of m symbols and $Y = \langle y_1, \cdots, x_n;\ y_j \in \mathbb{R};\ j = 1 \cdots n\rangle$ be a sequence of n symbols where x_i and y_j are the tonic normalized pitch values in cents [9]. The similarity between two pitch values, x_i and y_j, is defined as

$$sim(x_i, y_j) = \begin{cases} 1 - \frac{|x_i - y_j|^3}{(3s_t)^3} & if \ \mid x_i - y_j \mid < 3s_t \\ 0 & otherwise \end{cases} \quad (1)$$

where s_t represents a semitone in cents. Due to different styles of various musicians, an exact match between two pitch values contributing to the same *svara* cannot be expected. Hence, in this paper a leeway of 3 semitones is allowed between pitch values. Musically two pitch values, 3 semitones apart, cannot be called similar but this issue is addressed by the cubic nature of the similarity function. The function reaches its half value when the difference in two symbols is approximately half a semitone. Therefore, higher similarity scores are obtained when the corresponding pitch values are at most half a semitone apart.

A common subsequence Z_{XY} in sequences X and Y is defined as

$$Z_{XY} = \begin{cases} \langle (x_{i_1}, y_{j_1}), \cdots, (x_{i_p}, y_{j_p}) \rangle \\ 1 \le i_1 < \cdots < i_p \le m \\ 1 \le j_1 < \cdots < j_p \le n \\ \underset{k=1,\cdots,p}{sim} (x_{i_k}, y_{j_k}) \ge \tau_{sim} \end{cases} \quad (2)$$

where τ_{sim} is a threshold which decides the membership of the symbol pair (x_{i_k}, y_{j_k}) in a subsequence Z_{XY}. The value of τ_{sim} is decided empirically based on the domain of the problem as discussed in Section 5. An example common subsequence is shown with red color in Figure 1.

3.1 Common segments

Continuous symbol pairs in a common subsequence are referred to as a segment. Two different types of segments are defined, namely hard and soft segments.

Hard segment is a group of common subsequence symbols such that there are no gaps in between as shown in green color in Figure 1. Then a hard segment, starting with

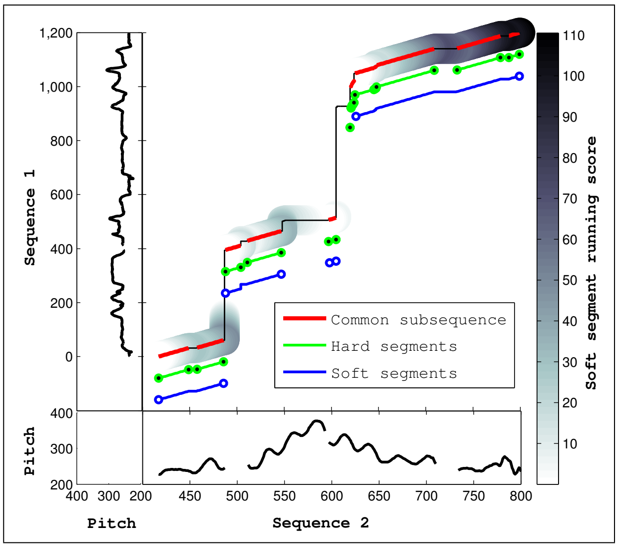


Figure 1. An example of a common segment set between two sequences representing the real data

a symbol pair (x_i, y_j), must be of the form

$$H^l_{X_i Y_j} = \begin{cases} \langle (x_i, y_j), (x_{i+1}, y_{j+1}), \cdots, (x_{i+l}, y_{j+l}) \rangle \\ 1 \le i < i+1 < \cdots < i+l \le m \\ 1 \le j < j+1 < \cdots < j+l \le n \end{cases} \quad (3)$$

where $l + 1$ represents the length of the hard segment. The score of the k^{th} hard segment $H^l_{X_{i_k} Y_{j_k}}$ is defined as

$$hc\left(H^l_{X_{i_k} Y_{j_k}}\right) = \sum_{d=0}^{l} sim\left(x_{i_k+d}, y_{j_k+d}\right) \quad (4)$$

Soft segment is a group of common subsequence symbols where gaps are permitted with a penalty. Therefore, a soft segment consists of one or more hard segments (shown with blue color in Figure 1). The gaps between the hard segments decides the penalty assigned. Thus, the score of the k^{th} soft segment $S_{X_{i_k} Y_{j_k}}$, consisting of r hard segments, is defined as

$$sc\left(S_{X_{i_k} Y_{j_k}}\right) = \sum_{s=1}^{r} hc\left(H^l_{X_{i_k} Y_{j_k}}\right) - \gamma\rho \quad (5)$$

where γ is the total number of gaps between r hard segments and ρ is the penalty for each gap. The number of hard segments to be included in a soft segment is decided by the running score of the soft segment. The running score of the soft segment increases during the hard segment and decreases during the gap due to penalties as shown in gray-scale in Figure 1. During a gap, if the running score decreases below a threshold τ_{rc} (or becomes almost white in Figure 1) then that gap is ignored and all the hard segments, encountered before it, are included into a soft segment.

3.2 Common segment set

All segments together correspond to a segment set. The score of a segment set (ss) is defined as

$$score\left(ss_{XY}\right)=\frac{\sum_{k=1}^{p} c\left(Z_{X_{i_k}Y_{j_k}}\right)^2}{\min(m,n)^2} \tag{6}$$

where p is the number of segments, c refers to the score computed in either (4) or (5) and Z refers to a segment (hard or soft). This equation gives preference to longer segments. For example, in case 1, there are 10 segments each of length 2 and in case 2, there are 4 segments each of length 5. In both the cases the total length of the segments is 20 but in (6), case 1 is scored as 0.1 and case 2 is scored as 0.25 when the denominator is taken to be 20^2. Longer matched segments could be considered as a phrase or an essential part of it. Whereas, shorter matched segments could generally mean noise. Therefore, there is a heavier penalty for shorter segments.

3.3 Longest common segment set

Longest common segment set (lcss) is a segment set with maximum score value as defined in (7).

$$lcss_{XY}=\underset{ss_{XY}}{argmax}\left(score\left(ss_{XY}\right)\right) \tag{7}$$

Therefore, lcss can be obtained by maximizing score in (6) using dynamic programming.

3.4 Dynamic Programming algorithm to find longest common segment set

The algorithm for finding the optimum soft segment set is given in Algorithm 1. Optimum hard segment sets are found similarly. In the algorithm, tables *c* and *s* are used for storing the running score and the score of the common segment sets, respectively. Table *a* is used for storing the partial scores from *s*. Table *d* is maintained for backtracking the path of the LCSS. The arrows represent the subpath to take while backtracking (up, left or cross). Input sequences to function LCSS are appended with symbols ϕ_x and ϕ_y such that their similarity with any symbol is 0. This is mainly required to compute the last row and column of score table. On similarity, line 8 updates the running score with a value based on the similarity, whereas line 9 updates the score using the previous diagonal entry. When symbols are dissimilar a gap is found. Lines 12 and 19 are used to penalize the running score. If it is an end of the segment then line 14 and 21 updates score as per (6). Line 26 updates table *a* with the score value of the current segment set when the beginning of a new segment is encountered. When a gap is encountered line 28 updates it to -1. To find the longest common segment set, backtracking is performed to obtain the path in table *d* that has the maximum score as given by table s. The boundaries of soft segments can be found using the cost values while tracing the path.

Algorithm 1 Algorithm for Soft-Longest Common Segment Set

Data:
c - table of size $(m+2)\times(n+2)$ for storing running score
s - table of size $(m+2)\times(n+2)$ for storing score
d - table of size $(m+2)\times(n+2)$ for path tracking
a - table of size $(m+2)\times(n+2)$ for storing partial scores.

1: **function** LCSS $(\langle x_1,\cdots,x_m,\phi_x\rangle,\langle y_1,\cdots,y_n,\phi_y\rangle)$
2: Initialize 1^{st} row and column of *c*, *s*, *d* and *a* to 0
3: $p \leftarrow \min(m,n)$
4: **for** $i \leftarrow 1$ to $m+1$ **do**
5: **for** $j \leftarrow 1$ to $n+1$ **do**
6: **if** $sim(x_i,\,y_j) > \tau_{sim}$ **then**
7: $d_{i,\,j} \leftarrow$ "$\nwarrow$"
8: $c_{i,\,j} \leftarrow c_{i-1,\,j-1} + \left(\frac{sim(x_i,\,y_j)-\tau_{sim}}{1-\tau_{sim}}\right)$
9: $s_{i,\,j} \leftarrow s_{i-1,\,j-1}$
10: **else if** $c_{i-1,\,j} < c_{i,\,j-1}$ **then**
11: $d_{i,\,j} \leftarrow$ "$\uparrow$"
12: $c_{i,\,j} \leftarrow \max(c_{i-1,\,j}-\rho,0)$
13: **if** $d_{i-1,\,j} =$ "$\nwarrow$" **then**
14: $s_{i,\,j} \leftarrow \frac{a_{i-1,\,j}*p^2+c_{i-1,\,j}^2}{p^2}$
15: **else**
16: $s_{i,\,j} \leftarrow s_{i-1,\,j}$
17: **else**
18: $d_{i,\,j} \leftarrow$ "$\leftarrow$"
19: $c_{i,\,j} \leftarrow \max(c_{i,\,j-1}-\rho,0)$
20: **if** $d_{i,\,j-1} =$ "$\nwarrow$" **then**
21: $s_{i,\,j} \leftarrow \frac{a_{i,\,j-1}*p^2+c_{i,\,j-1}^2}{p^2}$
22: **else**
23: $s_{i,\,j} \leftarrow s_{i,\,j-1}$
24: $q \leftarrow \max(a_{i-1,\,j-1},\,a_{i-1,\,j},\,a_{i,\,j-1})$
25: **if** $q=-1$ and $d_{i,\,j} =$ "$\nwarrow$" **then**
26: $a_{i,\,j} \leftarrow s_{i-1,j-1}$
27: **else if** $c_{i,\,j} < \tau_{rc}$ **then**
28: $a_{i,\,j} \leftarrow -1$
29: **else**
30: $a_{i,\,j} \leftarrow q$

4 Raga Verification

Let $\mathrm{T}_{r\bar{a}ga}=\left\{t_1,t_2,\cdots,t_{N_{r\bar{a}ga}}\right\}$ represent a set of template recordings, where '*rāga*' refers to the name of the *rāga* and $N_{r\bar{a}ga}$ is the total number of templates for that *rāga*. During testing, an input test recording, X, with a $claim$ is tested against all the template recordings of the claimed *rāga*. The final score is computed as given in (8).

$$\text{score}\left(\mathrm{X},\text{claim}\right)=\max_{Y\in \mathrm{T}_{claim}}\left(\text{score}\left(\text{lcss}_{XY}\right)\right) \tag{8}$$

The final decision, of accepting or rejecting the claim, directly based on this score could be erroneous. Score normalisation with cohorts is essential to make a decision, especially when the difference between two *rāgas* is subtle.

4.1 Score Normalization

LCSS scores corresponding to correct and incorrect claims are referred as true and imposter scores, respectively. If the imposter is a cohort *rāga*, then the imposter score is also referred as cohort score. Various score normalization techniques are discussed in the literature for speech recognition, speaker/language verification and spoken term detection [1, 17].

Zero normalization (Z-norm) uses the mean and variance estimate of cohort scores for scaling. The advantage of Z-norm is that the normalization parameters can be estimated off-line. Template recordings of a *rāga* are tested against template recordings of its cohorts and the resulting scores are used to estimate a *rāga* specific mean and variance for the imposter distribution. The normalized scores using Z-norm can be calculated as

$$\underset{norm}{\text{score}}\,(\text{X}, \text{claim}) = \frac{\text{score}\,(\text{X}, \text{claim}) - \mu_I^{\text{claim}}}{\sigma_I^{\text{claim}}} \quad (9)$$

where μ_I^{claim} and σ_I^{claim} are the estimated imposter parameters for the claimed *rāga*.

Test normalization (T-norm) is also based on a mean and variance estimation of cohort scores for scaling. The normalization parameters in T-norm are estimated online as compared to their offline estimation in Z-norm. During testing, a test recording is tested against template recordings of cohort *rāgas* and the resulting scores are used to estimate mean and variance parameters. These parameters are then used to perform the normalization given by (9).

The test recordings of a *rāga* may be scored differently against templates corresponding to the same *rāga* or imposter *rāga*. This can cause overlap between the true and imposter score distributions. T-norm attempts to reduce this overlap. The templates that are stored and the audio clip that is used during test can be from different environments.

5 Performance evaluation

In this section, we describe the results of *rāga* verification using LCSS algorithm in comparison with Rough Longest Common Subsequence (RLCS) algorithm [15] and Dynamic Time Warping (DTW) algorithm using different normalizations.

5.1 Experimental configuration

Only 17 *rāgas* out of 30 were used for *rāga* verification as only for 17 *rāgas* sufficient number of relevant cohorts could be obtained from the 30 *rāgas*. This is due to non-symmetric nature of the cohorts as discussed in Section 2. For *rāga* verification, 40% of the pallavi lines are used as templates and remaining 60% are used for testing. This partitioning of dataset is done into two ways, referred as D1 and D2. In D1, the variations of a pallavi line might fall into both templates and test though it is not necessary. Variations of a pallavi line are different from the pallavi line due to improvisations. In D2, these variations can either belong to template or they all belong to test but strictly not present in both. The values of thresholds τ_{sim} and τ_{rc} are empirically chosen as 0.45 and 0.5, respectively. Penalty, ρ, issued for gaps in segments is empirically chosen as 0.5.

5.2 Results

Table 2 and Figure 2 show the comparison of LCSS with DTW and RLCS using different normalizations. Equal Error rate (EER) refers to a point where false alarm rate and miss rate is equal. For T-norm, the best 20 cohort scores were used for normalization. LCSS (soft) with T-norm performs best for D1 around the EER point, and for high miss rates and low false alarms, whereas it performs poorer than LCSS (hard) for low miss rates and high false alarms. This behavior appears to be reversed for D2. The magnitude around EER is much greater for D2. This is because, none of the variations of the pallavi lines in test are present in the templates. It is also shown that RLCS performs poorer than any other algorithms for D2. The curves also show no improvements for Z-norm compared to baseline with no normalization. This can happen due to the way normalization parameters are estimated for Z-norm. For example, some of the templates, which may not be similar to the test, can be similar to some of the cohorts' templates, resulting in higher mean. This would not have happened in T-norm where the test itself is tested against the cohorts' templates.

Algorithm	Dataset	No Norm	Z-norm	T-Norm
DTW	D1	27.78	29.88	17.45
	D2	40.81	40.03	35.96
RLCS	D1	24.43	27.22	14.87
	D2	41.72	42.58	41.20
LCSS (hard)	D1	29.00	31.75	15.65
	D2	40.28	40.99	34.11
LCSS (soft)	D1	21.89	24.11	12.01
	D2	37.24	38.96	34.57

Table 2. EER(%) for different algorithms using different normalizations on different datasets.

6 Discussion

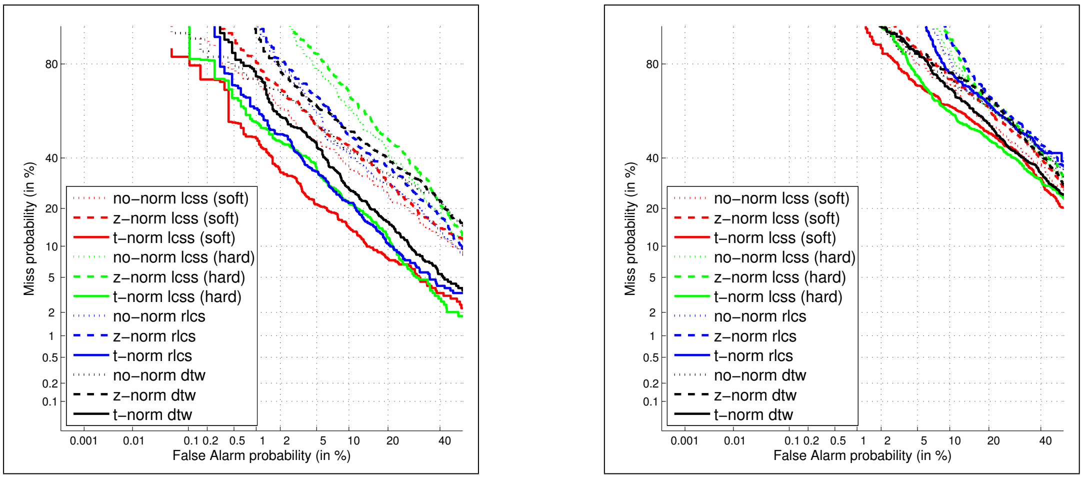
In this section, we discuss how LCSS (hard) and LCSS (soft) can be combined to achieve better performance. We also verify that T-norm reduces the overlap between true and imposter scores.

6.1 Combining hard-LCSS and soft-LCSS

Instead of selecting a threshold, we will assume that a true claim is correctly verified when its score is greater than all the cohort scores. Similarly, a false claim is correctly verified when its score is lesser than atleast one of the cohort scores. Table 3 shows the number of claims correctly verified only by hard-LCSS, only by soft-LCSS, by both and by neither of them. It is clear that there is an overlap between the correctly verified claims of hard-LCSS and soft-LCSS. Nonetheless, the number of claims distinctly verified by both is also significant. Therefore, the combination of these two algorithms could result in a better performance.

6.2 Reduction of overlap in score distribution by T-norm

Figure 3 shows the effect of T-norm on the distribution of hard-LCSS scores. It is clearly seen that the overlap, between the true and imposter score distributions, is reduced



a) DET curves for dataset D1 b) DET curves for dataset D2

Figure 2. DET curves comparing LCSS algorithm with different algorithms using different score normalizations

Dataset	Claim-type	Hard-only	Soft-only	Both	Neither
D1	True	23	55	289	77
	False	46	78	1745	54
D2	True	47	23	155	220
	False	99	75	1585	168

Table 3. Number of claims correctly verified by hard-LCSS only, by soft-LCSS only, by both and by neither of them for D1 and D2 using T-norm

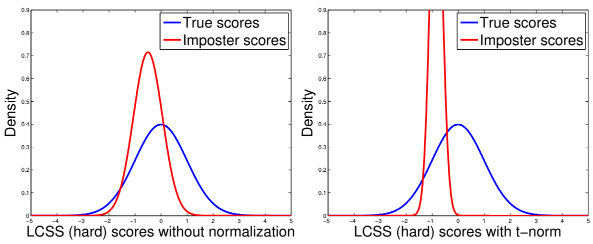


Figure 3. Showing the effect of T-norm on the score distribution

significantly. For visualization purposes, the true score distributions are scaled to zero mean and unit variance and corresponding imposter score distributions are scaled appropriately.

6.3 Scalability of *rāga* verification

The verification of a *rāga* depends on the number of its cohort *rāgas* which are usually 4 or 5. Since it does not depend on all the *rāgas* in the dataset, as in *rāga* identification, any number of *rāgas* can be added to the dataset.

7 Conclusion and future work

In this paper, we have proposed a different approach to *rāga* analysis in Carnatic music. Instead of *rāga* identification, *rāga* verification is performed. A set of cohorts for every *rāga* is defined. The identity of an audio clip is presented with a claim. The claimed *rāga* is verified by comparing with the templates of the claimed *rāga* and its cohorts by using a novel approach. A set of 17 *rāgas* and its cohorts constituting 30 *rāgas* is tested using appropriate score normalization techniques. An equal error rate of about 12% is achieved. This approach is scalable to any number of *rāgas* as the given *rāga* and its cohorts need to be added to the system.

8 Acknowledgments

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). We are grateful to Padmasundari for selecting the cohorts.

9 References

[1] R. Auckentaler, M Carey, and H Lloyd-Thomas. Score normalisation for text-independent speaker verification systems. *Digital Signal Processing*, 10:42–54, 2000.

[2] P Chordia and A Rae. Raag recognition using pitch-class and pitch-class dyad distributions. *In Proceedings of International Society for Music Information Retrieval Conference*, pages 431–436, 2007.

[3] Pranay Dighe, Parul Agarwal, Harish Karnick, Siddartha Thota, and Bhiksha Raj. Scale independent raga identification using chromagram patterns and swara based features. In *Proceedings of IEEE International Conference on Multimedia and Expo Workshops*, pages 1–4, 2013.

[4] Pranay Dighe, Harish Karnick, and Bhiksha Raj. Swara histogram based structural analysis and identification of indian classical ragas. In *Proceedings of 14th International Society for Music Information Retrieval Conference*, pages 35–40, 2013.

[5] Shrey Dutta and Hema A. Murthy. Discovering typical motifs of a raga from one-liners of songs in carnatic music. In *Proceedings of 15th International Society for Music Information Retrieval Conference*, pages 397–402, 2014.

[6] D.P.W. Ellis and G.E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages IV–1429–IV–1432, 2007.

[7] S Arthi H G Ranjani and T V Sreenivas. Shadja, swara identification and raga verification in alapana using stochastic models. *In IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 29–32, 2011.

[8] Vignesh Ishwar, Ashwin Bellur, and Hema A Murthy. Motivic analysis and its relevance to raga identification in carnatic music. In *2nd CompMusic Workshop*, 2012.

[9] Vignesh Ishwar, Shrey Dutta, Ashwin Bellur, and Hema A. Murthy. Motif spotting in an alapana in carnatic music. In *Proceedings of 14th International Society for Music Information Retrieval Conference*, pages 499–504, 2013.

[10] Gopala Krishna Koduri, Sankalp Gulati, and Preeti Rao. A survey of raaga recognition techniques and improvements to the state-of-the-art. *Sound and Music Computing*, 2011.

[11] Gopala Krishna Koduri, Sankalp Gulati, Preeti Rao, and Xavier Serra. Raga recognition based on pitch distribution methods. *Journal of New Music Research*, 41(4):337–350, 2012.

[12] A.S. Krishna, P.V. Rajkumar, K.P. Saishankar, and M. John. Identification of carnatic raagas using hidden markov models. In *IEEE 9th International Symposium on Applied Machine Intelligence and Informatics*, pages 107 –110, 2011.

[13] T M Krishna and Vignesh Ishwar. Carnatic music : Svara, gamaka, motif and raga identity. In *2nd CompMusic Workshop*, 2012.

[14] V. Kumar, H. Pandya, and C.V. Jawahar. Identifying ragas in indian music. In *Proceedings of 22nd International Conference on Pattern Recognition*, pages 767–772, 2014.

[15] Hwei-Jen Lin, Hung-Hsuan Wu, and Chun-Wei Wang. Music matching based on rough longest common subsequence. *Journal of Information Science and Engineering*, pages 27, 95–110., 2011.

[16] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *Proceedings of International Conference on Music Information Retrieval*, pages 288–295, 2005.

[17] Jiri Navratil and David Klusacek. On linear dets. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 229–232, 2007.

[18] Gurav Pandey, Chaitanya Mishra, and Paul Ipe. Tansen : A system for automatic raga identification. In *Proceedings of 1st Indian International Conference on Artificial Intelligence*, pages 1350–1363, 2003.

[19] P. Rao, J. Ch. Ross, K. K. Ganguli, V. Pandit, V. Ishwar, A. Bellur, , and H. A. Murthy. Melodic motivic analysis of indian music. *Journal of New Music Research*, 43(1):115–131, 2014.

[20] Joe Cheri Ross, Vinutha T. P., and Preeti Rao. Detecting melodic motifs from audio for hindustani classical music. In *Proceedings of 13th International Society for Music Information Retrieval Conference*, pages 193–198, 2012.

[21] Sridharan Sankaran, Krishnaraj P V, and Hema A Murthy. Automatic segmentation of composition in carnatic music using time-frequency cfcc templates. In *Proceedings of 11th International Symposium on Computer Music Multidisciplinary Research*, 2015.

[22] J. Serra, E. Gomez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008.

[23] Joan Serrà, Gopala K. Koduri, Marius Miron, and Xavier Serra. Assessing the tuning of sung indian classical music. In *Proceedings of 12th International Society for Music Information Retrieval Conference*, pages 157–162, 2011.

[24] Sankalp Gulati Joan Serra and Xavier Serra. An evaluation of methodologies for melodic similarity in audio recordings of indian art music. In *Proceedings of 40th IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[25] Surendra Shetty. Raga mining of indian music by extracting arohana-avarohana pattern. *International Journal of Recent trends in Engineering*, 1(1), 2009.

[26] Rajeswari Sridhar and Tv Geetha. Raga identification of carnatic music for music information retrieval. *International Journal of Recent trends in Engineering*, 1(1):1–4, 2009.

[27] M Subramanian. Carnatic ragam thodi pitch analysis of notes and gamakams. *Journal of the Sangeet Natak Akademi*, XLI(1):3–28, 2007.

[28] D Swathi. Analysis of carnatic music: A signal processing perspective. MS Thesis, IIT Madras*, India*, 2009.