Melody based Music Information Retrieval

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Swapnil Sarda
Roll No: 06305025

under the guidance of

Prof. Preeti Rao

and

Prof. Rushikesh Joshi

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Dissertation Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

The dissertation entitled **"Melody based Music Information Retrieval"**, submitted by **Swapnil Sarda** (Roll No: **06305025**) is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology, Bombay**.

Prof. Preeti Rao, EE, IITBombay
Supervisor

Prof. R. K. Joshi, CSE, IITBombay
Co-Supervisor

Prof. G. Sivakumar, CSE, IITBombay
Internal Examiner

Prof. Bikash Dey, EE, IITBombay
External Examiner

Prof. S. A. Khaparde, EE, IITBombay
Chairman

Place: IITBombay, Mumbai
Date:

Acknowledgements

I would like to thank **Prof. Preeti Rao** for her guidance and support towards this project. I would like to thank **Prof. Rushikesh Joshi** for his guidance in this project. I also wish to express my thanks to **Prof. G. Sivakumar** for his advice towards the project.

Swapnil Sarda,
I.I.T. Bombay,
July 27^{th}, 2008.

Abstract

Music retrieval systems of today use bibliographic information like singer, genre, song lyrics etc to retrieve a piece of music from a collection of music pieces. This bibliographic information is recorded manually and is not always available. Intelligent Music Information Retrieval help in automatically detecting and filling the missing information like singer or genre of a song removing the need of manually recording this information. Music data can be indexed and retrieved using different categorizations. It enables searching music using intuitive methods like querying by musical phrases, humming, singing or searching for music with similar genre, raga, patterns and so on.
This work focuses on some of the components of Music Information Retrieval based on melody. Raga is a melodic framework of categorization of music in Indian Classical Music. One major component of this work is classification of music based on this raga framework. Second component includes automatic music transcription which converts music from audio to symbolic form i.e. extraction the sequence of notes from the music in audio form. Another component finds similarity between two pieces of music and searches for music based on this similarity.

Contents

1 Introduction

Music retrieval systems of today use bibliographic information like singer, genre, song lyrics etc to retrieve a piece of music from a collection of music pieces. This bibliographic information is recorded manually and is not always available. Intelligent information retrieval techniques remove the need to manually record these attributes of the music. Music Information Retrieval systems help detecting and filling the missing information like singer or genre of a song. They enable the categorization the music data under various frameworks and attributes. Music data can be indexed and retrieved using different categorizations. This can be used in searching music using intuitive methods like querying by musical phrases, humming, singing. Songs can be searched with similar raga, genre as that of the queried song. Downie[4] gives a starting point information on Music Information Retrieval and variou4s interesting problems associated with it.

Indian Classical Music relies on melody along with rhythm. The framework to structure and categorize different kinds of melodies is called raga. Any musical performance in ICM is performed in one or the other raga. For any given raga there are rules and constraints under which a performer has to create his or her musical expressions. The rules include the allowed and disallowed notes, special recurring patterns or motifs like ascent, descent, catch-phrases, certain notes with special importance, etc. A raga signify a particular mood or emotion. This well-founded framework of raga can be used for intelligent MIR. For this we need raga detection. Raga detection in a music performance is automatic categorization of the musical piece with its associated raga. After detecting ragas for a collection of songs in a music database, they can be indexed so as to retrieve them based on melody.

Second component includes automatic music transcription. It involves converting music from audio to symbolic form i.e. extraction the sequence of notes from the music in audio form. The resulting output is used for various purposes including similarity matching or string based classification. Also the output of this step is itself an useful information for music performers, students or music theorists.

Another component of this work finds the similarity between two pieces of music based on approximate string matching of two sequences of notes. This similarity is then used for searching for melody similar to the queried melody.

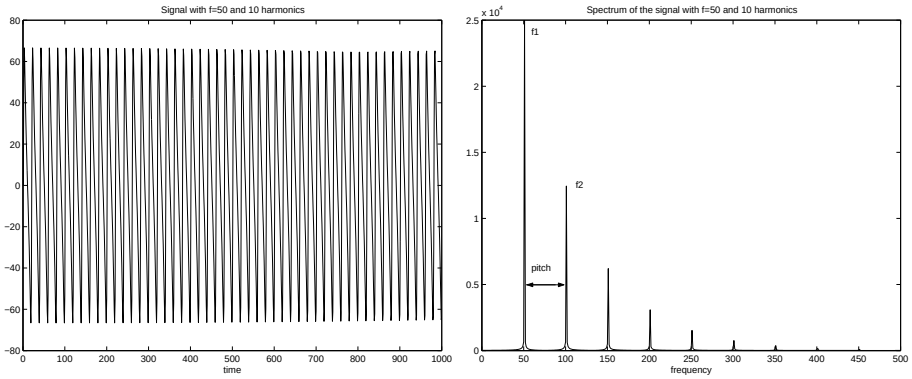
2 Preliminaries

2.1 Attributes of Sound and Music

Sound is a disturbance of mechanical energy that propagates through matter as a wave.

Physical attributes of sound :

- **Amplitude :** It is the peak deviation of a particle from its mean position on a wave.
- **Frequency (f) :** It is the number of vibrations a particle completes around its mean position in unit time.
- **Wavelength (λ) :** It is the smallest distance between any two points of a wave in phase with each other.
- **Speed (v) :** Speed means the velocity at which the sound waves or energy travels. Speed of sound is 340 m/s in air at. Speed is related to wavelength and frequency by : $speed = frequency \times wavelength$.
- **Harmonics :** The integral multiples of a frequency are called harmonics. e.g. f, 2f 3f are the first three harmonics of the frequency f.



A signal with fundamental frequency of 50 and 10 harmonics, its spectrum, fundamental frequency, pitch and harmonics are shown in the figure.

Perceptual attributes of sound:

- **Pitch :** Pitch is the perceived fundamental frequency of a sound. It is the difference between the consecutive harmonics in found in the spectrum of the sound. Though presence of higher harmonic is not necessary for the perception of pitch, all the naturally occurring sounds like voice and musical instruments have many harmonics. Even the fundamental frequency may be absent from the spectrum and it can still be perceived from the difference between the harmonics. The fundamental frequency of pitch is denoted by f0.

- **Loudness :** Loudness is the intensity at which we perceive sound. Though it is proportional to the amplitude of the sound wave, human perception of amplitude is not linear but logarithmic. So a sound with double amplitude is not perceived with doubled loudness but just +3db louder.

- **Timbre :** Timbre is what distinguishes one instrument from another. It depends on the relative strengths of harmonics and the formant (described below) structure of the sound.

- **Voice quality :** Quality of human voice can be considered analogous to timbre of an instrument.

  Source-Filter model of voice production : Vocal chords (source) produce the sound with the intended pitch. This sound is then filtered by the vocal tract (can be considered as a tube). A tube is a filter with resonances at some particular frequencies which depend on the length of the tube alone. At these resonating frequencies the original sound energy is enhanced, other frequencies are damped. These frequency bands are called **formants**. These formant are what gives rise to different voices and different timbres. Formants are labeled as f1, f2, etc.

- **Note :** Note is the basic unit of melody. Pattern or sequence of notes makes the melody. Notes can be considered as to be the discretization of pitch. Pitch takes all the continuous values. Notes take some disrate pitch values. Notes are named as C, D, etc. or Sa, Re, etc.

- **Octave :** Octave is the pitch or frequency interval where the higher frequency is double of the lower frequency. There are twelve semitones in an octave. As with the perception of loudness, human perception about pitch is also logarithmic. Therefore the notes in an octave are marked logarithmically.

$$Note_i = 12\ log_2\left(\frac{Freq_i}{Freq_{low}}\right) \text{ or}$$
$$Freq_i = 2^{\left(\frac{Note_i}{12}\right)} \times Freq_{low}$$

Here $Freq_i$ is the frequency of the i^{th} note from note with frequency $Freq_{low}$.

Though all the twelve semi-tones are (logarithmically) almost equidistant, human perception doesn't accept them as equidistant. Therefore only seven tones are perceived as full-tones.

- **Musical Scales :** A musical scale tells the way in which an octave is divided in notes. The octave division mentioned above is an example of **equal tempered scale** in which all the notes are equidistant (in logarithmic sense) in throughout the pitch range. Western music uses this scale and is increasingly finding popularity as it does not need retuning for different instruments or while changing the key of performance etc. Other examples of scales are the **just intonation scale** and the **pythagorean scale**. In just intonation scale, the pitches of all the notes are the ratios of small numbers. That means the notes are consonant. Indian Classical Music employs this scale. Pythagorean scale is derived from a stack of fifth notes (e.g. C and G) tuned in the ratio of 3:2. This scale produces pitch ratios of high integers and therefore sounds bad and not used today.

2.2 Melody and its representation

2.2.1 Music representation :

- **Raw format :** In the raw form music is represented as an audio waveform. It is the representation of the sound energy variations in a medium.

- **Symbolic form :** Symbolic form representation is not the output as sound waves. It is the musical essence of the sound. In the symbolic form melody can be stores as a sequence of notes.
  Specification of the instrument can be stored as changing magnitudes of various harmonics w.r.t. time.

Musical Instruments Digital Interface (**MIDI**) standard is used to represent music in symbolic form.

2.2.2 Melody representation :

- **Pitch Contour :** Pitch contour represents the variation in pitch with respect to time.
- **Pitch Histogram :** It is the distribution of occurrence of pitch over the entire range of pitch. This is called the unfolded pitch histogram. If we map all the pitches on a single octave, then it is called a folded pitch histogram.
- **Chroma Vector :** It is the 12-dimensional vector representing the distribution of occurrences of notes in the given piece folded on a single octave of 12 notes. Note that chroma vector is still different from the folded pitch histogram. Pitch takes all the values in the range of one octave in folded pitch histogram whereas in chroma vector, pitch takes only twelve values. This means chroma vector is distribution of occurrences of notes within one octave.

2.3 Indian Classical Music

Melody(raga) and Rhythm(tala) are the two basic foundations of Indian Classical Music. Indian Classical Music performance consists mainly of two tracks. One is pitched - vocal or instrumental - consisting melody. Other track is the accompaniment by unpitched percussion instrument which gives the rhythmic support to the melody.

Raga : Raga is a melodious framework which a performer sets for his performance and improvises over it with his own style. A raga specifies a set of allowed notes (minimum five) that can be played in that raga. It has certain patterns of notes like ascent, descent, catch-phrase associated with it. There is one note with special importance called sonant (vaadi). Sonant can be the most played note or it can be the note ending most phrases or has some special significance in the motifs. Another note with second importance is called consonant (samvaadi). A raga signify a particular mood. Different ragas evoke different emotions.

Tala : Tala is to rhythm, what raga is to melody. There are several talas. Each one has its own number of divisions, number of beats, stress places (sam), release places (khaali) and so on. Like raga, tala can also be

improvised. Improvisation of tala depends upon the instruments.
Examples of talas : tintaal, dadara.

Indian Classical Music solfege called **sargam** consists of these seven notes: sa re ga ma pa dha ni sa.

Indian Classical Music is generally played in three octaves (**saptak**) - mandra, madhya and taar and in three tempos (**laya**) - slow (Vilambit), medium and fast (drut).

3 Raga Detection

The raga detection can be used in MIR for retrieving songs based on raga-similarity, in recommendation systems, for creating catalogs of songs and so on.

Two methods for raga detection are developed in this work. Both method use the pitch contour as their initial melody representation. First method converts this pitch contour in pitch distribution while the other method converts it to a sequence of notes.

For Indian Classical Music, variations in pitch are very important to distinguish between ragas. In the transcribed note sequence these minor variations of pitch are lost. The first method retains and uses this finer information while this information is discarded in the second method. This section describes the first method, second method is described in section 5.6.

3.1 Previous work

Chordia[3] uses pitch-class and pitch-class dyad distributions (PCD and PCDD) for the task. From a given note transcription, a folded pitch histogram is generated. This gives 12 dimensional feature vector. It is augmented by a pitch dyad (two consecutive notes) histogram. This is a 144 dimensional vector, totaling 156 dimensional feature vector. This is reduced by dimensionality reduction technique like PCA to 56 dimensions. This feature vector is then subjected to four different classification methods:

1. **Multi Variate Gaussian :** Here PCDs of various songs from a single raga are considered as generated from a single Multi Variate Gaussian Distribution. The parameters of these MVGs for each raga are learned. This distribution is the likelihood function. The prior is also calculated from the data as the proportion of the songs from the particular rag in the total songs. The prediction is done as the raga with the highest value of $prior \times likelihood$. Accuracy reported was 94%.

2. **Neural Network :** The feature vector acts as the input, and ragas as outputs. One hidden layer with 100 nodes gave accuracy of 75%.

3. **kNN :** k Nearest Neighbors method with k=1 was used. Accuracy was 67%.

4. **Decision Tree :** Decision trees go on splitting the data depending on a question at each stage. Minimal entropy strategy was used as the split criterion. Accuracy was 50%.

Constraints on dataset: The database of songs used was without any background music or noise. The tonic was know beforehand. The number of different ragas were 17 for sarod database and 11 for vocal database.

Raga is modeled as finite automaton by Sahasrabuddhe[8]. In this method there are various states in the progress of music and state transition depends on the latest notes. Note that the latest notes can be a finite sequence of notes (of very short length). The problem with method for using it for classification is that the music knowledge needs to be coded manually (in the form of automata states and transitions). It is difficult to learn the states from the data. An extension to this model could be probabilistic finite automaton or Hidden Markov Model where state transition occurs with some probability instead of depending on the latest notes. This gives the model some randomness.

Pandey[7] uses two stage approach to classification. The first stage uses a Hidden Markov Model. The HMM consists of the $12 \times 3 = 36$ states corresponding to 36 notes in 3 octaves. The note emitted in a state is the state itself, so there is no separate emission probability matrix. The state transition matrix is learned by Baum-Welch learning algorithm. The second stage consists of string matching. Raga-specific catch-phrases (pakads) are used as the reference strings. Following score is used to classify the song:

$$score_I = \sum_n \sum_j freq_{n,j,I}$$

where $score_I$ = number of times the j^{th} n-gram of the pakad of raga I is found in the input.

80% to 87% accuracy was reported for classification between two ragas.

Constraints on dataset: The database of songs used was without any background music or noise. The tonic was known beforehand. The number of different ragas were 2.

3.2 Feature Selection

Deciding which feature to use for the classification is a nontrivial problem. The features should be capable of distinguishing between the ragas and should be immune over other categories. For example values of features extracted from two performances in a same raga by two different singers should be as similar as possible. The features should be invariant to any categorization except raga-class. This include different performers, different styles of performance, key (or tonic or reference pitch), tempo, rhythm and so on.

The features selection should also take into account the background noise and accompaniments like tabla (percussion) and drone (which plays only the tonic)

The features mostly used for music information retrieval tasks include Cepstral Coefficients, MFCCs (Mel Frequency Cepstral Coefficients), brightness, bandwidth, loudness, pitch, spectral centroid, zero crossing rate. McKinney[5] gives comparative study of some of the features.

For our purpose, folded pitch distribution (FPD) is used for raga classification. FPD is the normalized folded pitch histogram. It is a probability distribution over the pitch. The intuition behind using this feature for raga classification is like this: first, in ICM the difference between octave notes hardly matters, so we can fold the pitch histogram on a single octave. Second in ICM pitch moves more continuously over the pitch range rather than being discretized into the fixed note pitches. Though there are the stable pitches corresponding to the notes, the variations around these notes is what distinguishes various ragas. So discretizing the pitch histogram into notes

loses this finer pitch information.

Also use of this feature relieves us from detecting the tonic as we have a method of aligning the FPDs (explained later) and removing the effect of the songs being sung in different keys.

The intuition is also verified by this experiment: FPDs of different songs are extracted and compared. FPDs of different songs with similar raga are found to have matching shapes (though they were circularly shifted because of the mismatch between tonics). While songs from different ragas have non-matching FPD shapes.

3.3 Feature Extraction

3.3.1 Pitch Contour Extraction

Pitch tracking means extracting the pitch of the sound-wave. Input to this is the raw wave audio signal. The output of this process is the variation of pitch w.r.t. time. This output is called the pitch contour. In ICM there is only one pitched track along with a un-pitched (or fixed pitched) percussion track. This pitched track can be a singing voice or an pitched instrument. So we need to track a single pitch track i.e. monophonic pitch tracking.

There are various pitch tracking algorithms, some sample of which are summarized here:

- Autocorrelation based method finds the autocorrelation of the windowed time domain wave signal for the lags starting from one up to the size of the window. The peak for the lowest lag corresponds to the pitch of the current window signal. The ¡lag¿ ÷ ¡sampling rate¿ gives the current period and its inverse gives the current pitch.

- A crude approach to measure pitch is to find the zero crossing rate. ZCR is the distance between two zero crossing points.

- Harmonic Product Spectrum: This method finds pitch in the frequency domain unlike last two methods which are time domain approached. The windowed time domain signal is converted to frequency domain using fourier transform. For a frequency f, the magnitudes of this transform at all the f's integral multiple frequencies are multiplied. This is done for all the relevant frequency range. The lowest frequency

for which there is a peak in the above calculated function of f gives the current pitch.

- Cepstrum is a widely used tool for speech and music analysis. It is the inverse DFT of logarithm of DFT of the original signal. The logarithm in the frequency domain decouples the effects of source and filter in the source-filter model. It can be used for our purpose of finding the pitch. The first peak in the cepstrum gives the pitch duration, its inverse gives us the pitch.

In the current method, simple autocorrelation based monophonic pitch tracker is used for pitch contour extraction. The range of allowed pitch is set to 75Hz to 800Hz - comprising four octaves. A single song will not have this full pitch range but different songs with different singers have different pitch ranges. Hundred pitch values are obtained per second (i.e. at each 10 ms). One pitch value was estimated from a window of three periods of the lowest allowed pitch (75Hz), i.e. $3 \div 75\text{Hz} = 40$ ms.

The autocorrelation based methods is prone to octave errors, but as our method uses pitch folding to a single octave, these errors do not have any effect.

3.3.2 Perceptual Pitch Transform

The pitch contour obtained is not linear in terms of human perception or in terms of musical scale. Therefore it is transformed to the musical scale by the following logarithmic transform. The transformed values are the analog version of the MIDI note numbers.

$$midipitch = 69 + 12\ log_2\ \frac{pitch}{440.0}$$

3.3.3 Pitch Distribution

The pitch values from the pitch contour are divided in pitch-bins of equal size. This size called pitch resolution is a parameter of the system and can be adjusted. Pitch resolution of 10 cents (1 cent = 1/100 note = 1/1200 octave) is used for the classification. We get the **pitch histogram** over the pitch range of 75 to 800.

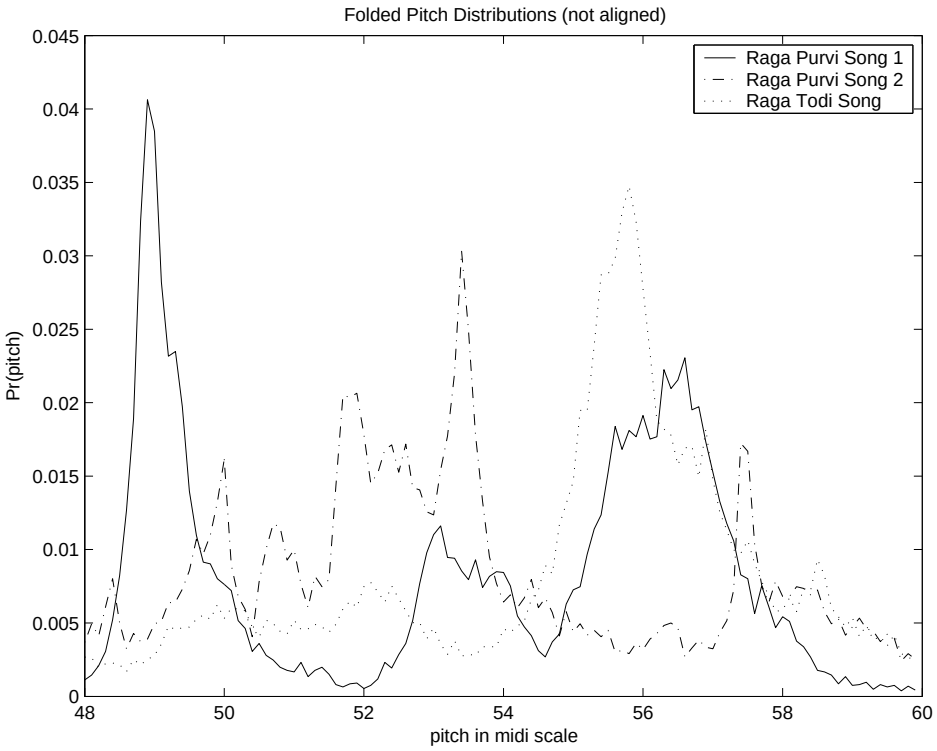


Figure 1: non-aligned folded pitch distributions

The next step is the folding which just adds all the pitch frequencies of the pitches with octave differences together and gives the 120-bin **folded pitch histogram**.

We then normalize the histogram by the sum of all its members which gives us the probability distribution of the pitches in one octave with the specified pitch resolution. This we call as **folded pitch distribution (FPD)**.

Non-aligned and aligned FPDs for two songs from the same raga Purvi and for one song from another raga Todi is shown in the figure 1 and figure 2 respectively. The similarity between songs from same raga is visible.

3.4 Classification

3.4.1 Similarity Measure

We have our input features are probability distributions. We need a similarity (or dissimilarity or distance) measure between them, so that we can

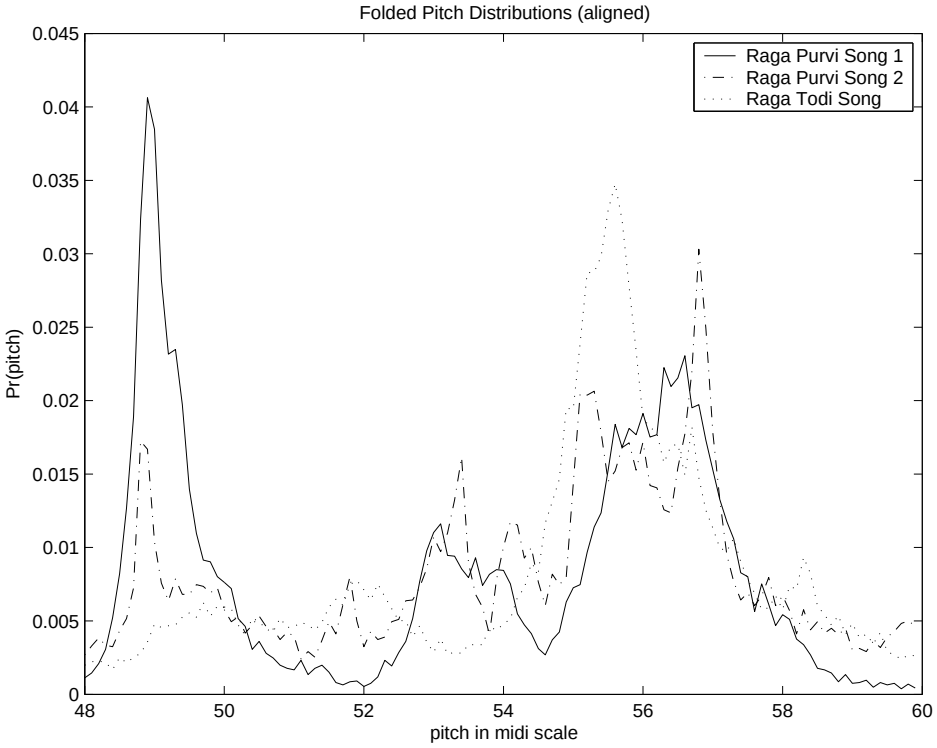


Figure 2: aligned folded pitch distributions

understand how close or far the given songs or the raga-classes are. KL divergence (Kullback-Leibler divergence) is used as a measure of dissimilarity between the probability distributions. It is given as,

$$KL(p||q) = \Sigma_f p(f) log_2 \frac{p(f)}{q(f)}$$

KL divergence is never negative. KL divergence means the extra information needed to represent a symbol drawn from distribution p while coding (compression) method assumes the distribution to be q. Thus if q is the pitch distribution of a raga and p is the pitch-distribution of a song, $KL(p||q)$ gives the inefficiency (or number of bits wasted) in coding the given song with the model based on the given raga. If they have the same distribution, this wastage is zero.

But KL divergence is not symmetric. To make it symmetric we sum the KL-divergences both ways.

$$d(p, q) = KL(p||q) + KL(q||p)$$

We will call this KL distance.

Another variant of symmetric KL-divergence is also used. In this the symmetric KL-divergence is normalized by the sum of the entropies of the two distributions. This is necessary only when we are using FPDs with different pitch-resolution (and thus different number of pitch values in the FPDs). In current implementation the pitch-resolution is kept constant, so normalizing by entropy is not necessary.

$$dn(p, q) = \frac{d(p, q)}{H(p) + H(q)}$$

Where entropy, $H(p) = \sum_f p(f) log_2 p(f)$

KL-distances between two songs from the same raga (Purvi) and for songs from different ragas (Purvi and Todi) are shown in the figure 3. The minimum of the former curve case is smaller as expected than the minimum of the later one.

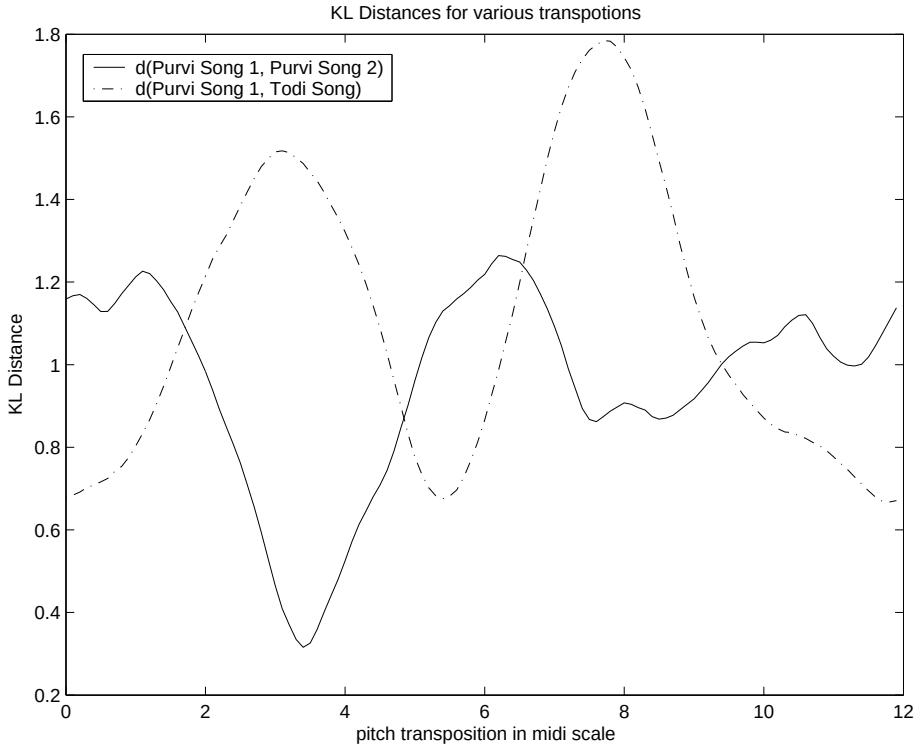


Figure 3: KL distances

3.4.2 Classifier Training

Folded pitch distributions, which are 120-valued vectors, for each song are extracted and stored.

In addition a summary FPD for each raga is also stored. It is calculated as follows:

Due to the difference in the tonic pitches in two FPDs, even if their shape is same, they are not necessarily aligned. We have this method for aligning the two FPDs : the KL distances between two FPDs are calculated by shifting one of the distribution by k = 1 to 120. The value k which gives the minimum KL distance is the correct alignment for the two FPDs.

All the FPDs of the songs from the same raga are aligned to a reference FPD selected from the same raga. These aligned FPDs are then added together with proper weights given to them according to the length of their respective songs. The weighted sum is then normalized to sum to one to make it probability distribution. And the result is stored as class FPD (cfpd) for the raga.

Correction to fpd by peak thresholding: For many songs, the pitch contours was found to latch onto the drone present in the song. The fpd with such erroneous pitch contour gives a large peak for the drone and the rest of the fpd is reduced to very small values reducing the discriminating power of the fpd. In order to counter this problem of the drone, peak thresholding was employed. The highest pitch peak in the fpd is downsized to ¡some threshold¿*¡second peak in the fpd¿ The excess removed from this peak is distributed amongst the rest of the fpd. This heuristic reduces the drone error if it is present. And it does not have any side effects if no drone is present.
Peak thresholding using threshold = 2.0 improved accuracy to 38% while threshold = 1.5 gave 36% accuracy.

3.4.3 Classification of song

Folded pitch distribution is extracted for the query song as it was done for training data. This fpd is aligned with cfpd's of each raga using the KL-divergence metric and their distances(symmetric KL-divergences) are calculated. The class with the smallest distance is declared to be the class of the song.

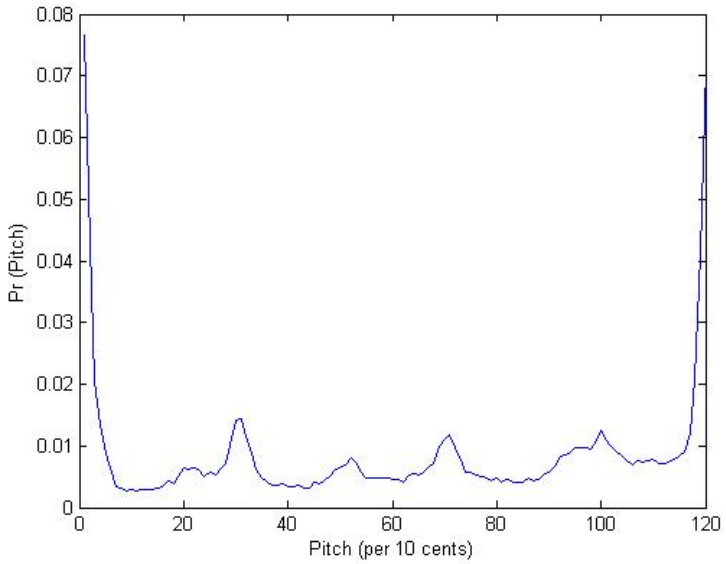


Figure 4: uncorrected fpd

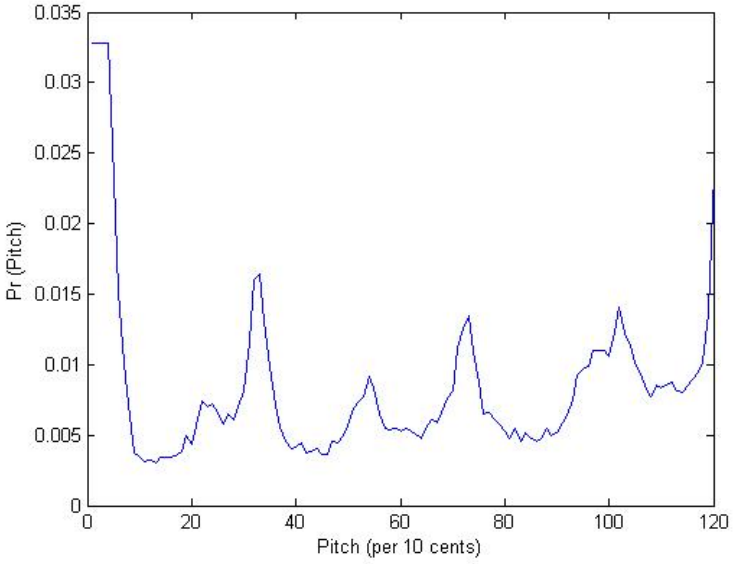


Figure 5: fpd corrected by peak thresholding

The problem with this method is that when the number of songs for a given raga are high, the variations across the FPDs of the songs collectively damage the shape of the summarized FPD which bear no resemblance to the true distribution and classification accuracy suffers. Therefore instead of comparing the query fpd with class cfpd's, it is compared with all the fpd's in the model. The class of the fpd for which the query fpd has the lowest KL distance is declared as the class of the query song.

3.5 Experiments and Results

3.5.1 Dataset

The main dataset used was a collection of recordings of vocal performances in ICM over last 100 years. It contained about 250 performances by about 35 performers in more than 100 ragas. The total duration of the recordings was around 800 minutes, each performance of about 3 to 4 minutes on average while some ranging from 2 mins to more than 10 mins. Most of the performances were recorded at 11 kHz. These recordings were obtained from http://homepage.mac.com/patrickmoutal/macmoutal/rag.html. The distribution of songs per raga is given in table 1

Constraints on dataset: As the dataset is not recorded for classification purpose but is sampled from the real recording, it is only constrained under the framework of Indian Classical Music. We don't know the tonic (or key) in which the song are sung. All the songs have the background background accompaniment of tabla and tanpura. The number of different ragas are 106.

3.5.2 Results and Comparison

The classifier takes four to five seconds to classify a song after its pitch is extracted. Time taken by pitch tracking is ten time less than the playback time of the song e.g. pitch is extracted from a three mins song in about twenty seconds.

There are various ways to measure the accuracy of the classification system. With the entire dataset with 106 ragas, the classification accuracy is 38%. This method to calculate this accuracy corresponds to leave one out

Table 1: Distribution of songs in current dataset

raga	songs	raga	songs	raga	songs
adana	6	asavari	2	asavari komalrishabh	1
bageshri	5	bahar	3	barwa	2
basant	3	basant mukhari	1	bhairav	1
bhairav ahir	2	bhairavi	20	bhairavi sindh	1
bhatiyar	2	bhimpalasi	0	bhupali	5
bibhas	1	bihag	1	bilawal	2
bilawal alhaiya	1	bilawal yamani	1	champak	1
chandrakauns	1	chhaya	1	chhayanat	2
darbari	3	desh	3	deskar	4
dhanashri	1	durga	3	gandhari	1
hamir	2	hanskalyan	1	hindol	3
hindol-bahar	1	hori	1	jaijaiwanti	3
jaunpuri	2	jog	1	jogiya	3
jogiya-kalingada	1	kafi	4	kafi bangiya	1
kafi sindh	3	kalavati	1	kalingada	1
kalyan savani	1	kamod	3	kaushikanada	1
kedar	2	kedar-bahar	1	khamaj	1
khambavati	3	kukhubh	1	lalit	2
malawati	1	malhar	2	malhar gaud	2
malhar jayant	1	malhar miyaki	6	malhar nat	1
malhar ramdasi	1	malhar sur	1	malkauns	8
mand	2	maru-bihag	2	marwa	5
megh	1	misrapilu	1	multani	3
naiki	1	nand	1	nayakikanada	1
nilambari	1	pahadi	1	pahadi-jhinjuti	1
paraj	2	patdeep	1	pilu	4
puriya	5	puriya-dhanashri	1	puriya-kalyan	1
purvi	7	ramkali	1	sarang	2
sarang gaud	3	sarang m	1	sarang miyaki	2
sarang vrindavani	3	shankara	4	shuddhkalyan	3
shuklabilawal	1	sindhura	1	sohini	3
sohini-bhatiyar	1	sohini pancham	1	sughrai	2
suha	2	tankeshri	2	tilakkamod	5
tilang	5	todi	5	todi bhupali	1
todi desi	2	todi miyaki	1	todigujri	1
yaman	5	zila	2		

cross validation method. As the number of classes are reduced, there is less competition between the classes and accuracy improves. Using 15 raga, the accuracy reached to 65%. These results are obtained when the query song was not included in the training set. For the case when the queried song was present in the training dataset, the system always not only identified the correct raga but the exact song over entire dataset. The accuracy for this case is 100%. When the test dataset was created by taking various segments of the songs, the accuracy changed as the function of segment size. For 60 seconds segments, accuracy reached close to 100%.

For comparing the accuracy, we can note the following:

The accuracy of random guess should be around 6%, taking into account the training and test data distributions with 106 ragas. The standard classifiers give maximum accuracy of 20% for SVM, nearest neighbor, 8% for decision trees using the same features viz., fpd's.

For comparison, classification results from Chordia[3] are presented. Chordia used two databases: One sarod (Ch-Sarod) and another vocal database (Ch-Vocal). Various attributes of these datasets along with the attributes of the current dataset (Current) are shown in the table 2.

Table 2: Attributes of three different music datasets

dataset	dotal duration	no. of songs	no. of performers	no. of ragas
Ch-Sarod	72	1	1 (sarod itself)	17
Ch-Vocal	160	39	12	11
Current	750	250	35	106

The accuracy reported in Chordia[3] for different methods are as follows: 94% using Multi Variate Gaussian, 75% for Neural Network, 67% using k Nearest Neighbors, 50% using Decision Tree. The reported accuracies are for the classification of fragments (of 30 secs) of the songs. Calculating accuracy in the same way, the current system classifies almost all fragments correctly meaning close to 100% accuracy. This is tested for fragments with duration greater than 60 seconds.

Besides the measures of accuracy, the current systems handles the problem of unknown tonic which makes it more practical. Further accuracy can be improved by creating template or model fpd's from clean data i.e. from song with no drone or accompaniments.

3.6 Song Retrieval

The classification component using the nearest neighbor method described above is inherently based on finding the similar songs. Thus it is exactly the content based song retrieval. Also it can be extended to similar-songs retrieval by just using k nearest neighbors instead of one nearest neighbor.

4 Music Transcription

4.1 Introduction

Extracting the sequence of notes from a piece of music is called music transcription.

Transcription through pitch contour: For transcribing a pitch contour, we need to detect the onset time, the time at which a new note is played or sung. For stringed instruments, energy based onset detection can be used. When a note is plucked there is high energy in the sound. A detection function exploiting the energy change can be used for detecting the note onset time.

Other detection functions can also be designed. e.g. A detection function can be constructed which exploit the sudden change in the phase or amplitude. As a steady state note will not have a sudden change in the phase or amplitude, we get a new note.

After detecting the onset time, we need to determine which note is played at that time. This can be done by finding the note whose standard pitch is closest to the current pitch in the pitch contour.

Next we need to find the duration of the note. The end-point of a note can be assumed to the onset time of the next note. This is fine if the silence can also be detected as a note.

Bello[1] gives two methods for monophonic and polyphonic music transcription. Autocorrelation is used for monophonic transcription. It finds the autocorrelation of the windowed time domain wave signal for values starting from one up to the first peak encountered. This peak is the pitch for the current window. Simultaneous another module computes the energy envelope of the signal. Another collector module uses both the pitch output and energy envelope to segment the pitch signal into notes.

Cemgil[2] gives a generative model to transcribe the acoustic signal into notes. This does not extract the pitch contour. Rather a probabilistic graphical model is used to infer the likelihood of the notes that may be generating the observed acoustic signal.

Various pitch correction and smoothing techniques can be applied to take care of the errors in initial pitch detection. For example Ney[6] gives a dynamic programming based method that takes into account both the global and local costs to assign the correct pitch values.

The next sections details the method employed for transcription of pitch contour to sequence of notes:

4.2 Finding stable pitches:

The method for song transcription uses the pitch histogram. After computing the pitch contour and pitch histogram for the song, peaks in the pitch histogram are found. The peaks are sorted in order of their strengths. Peaks other than top twelve are discarded as there should be only twelve stable pitches in one octave. These twelve peaks are sorted according to their pitch value. It is possible for some pitch distributions, to have two peaks in the same pitch-class interval to be greater than the peak at other valid but small stable pitch. Therefore an additional test is made ensure the pitches are really the stable pitches as follows. For each candidate stable pitch, following error is calculated

$$error_i = \Sigma_j |(v_i - v_j) - round(v_i - v_j)|$$

where v is the array of candidate stable pitches.

The pitch v_i with the lowest error value $error_i$ is assumed to the truly correct stable pitch and is considered as reference for further computations.

4.3 Discretization in pitch value:

This converts the pitch sequence to sequence of notes:

These peaks represent the stable pitches related to the notes. At a given point in time, a note (actually pitch) is searched from the stable pitches with the closest pitch to pitch value in the pitch contour.

$$note(i) = argmin_{p \in StablePitches} |p - pitch(i)|$$

Where pitch(i) is the pitch contour and note(i) is the output sequence of notes sampled at regular rate and StablePitches are the stable pitches computed as described earlier.

4.4 Discretization in time:

This converts the sequence of notes sampled at regular rate to sequence of notes as they are played i.e. sequence of notes where a note is counted only at its onset time. A note(i) is considered a new event if it satisfies two constrains:

1. note(i) is different than the last note i.e.

   $$note(i)! = note(i-1)$$

2. note(i) is different from the median of note(i-k) to note(i+k) i.e.

   $$note(i)! = median_{j=-k}^{k} note(i+j)$$

where k is the window parameter which is adjusted according to the expectation of the number of notes per unit time.

If these two conditions are met, a new note is emitted in the output noteonset(j) which is the median as defined above. Also a time counter is kept and current note onset time and duration is output along with the new note.

For pitch contour sampled per 10 milliseconds, k=4 to k=6 give good results. Figure 6 shows the process of transcription. The wavy contour is the original pitch contour. The more stable contour is the pitch-discretized note contour. The final notes in the transcription are shown with crosses.

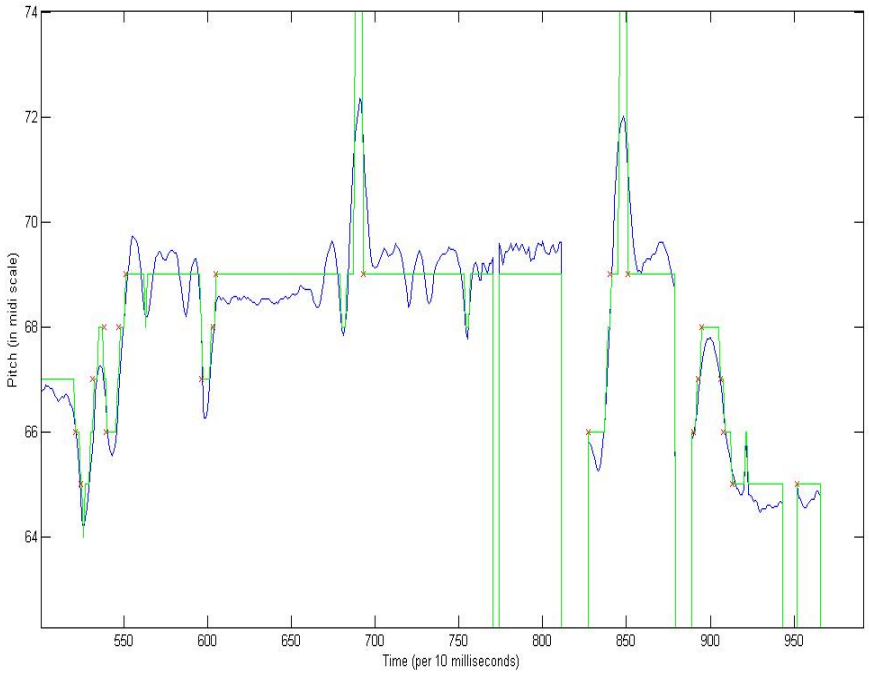


Figure 6: original pitch contour and pitch-discretized note contour. The final notes in the transcription are shown with crosses.

5 Melody based Music Search

The task involves content (melody) based searching of music. The task here involves matching the fragments of melodies, finding similarity between various songs based on melody, querying a song (or songs) with a short piece of melody. The melody is represented as a sequence of notes. We have to find the most similar subsequence to the query sequence of notes from the song database or just a single song.

5.1 Exact String Matching

Let t (text) and p (pattern) be two strings of length n and m respectively and $m \leq n$. The characters or elements of t and p come from the same finite alphabet Σ. Exact string matching problem is to find the exact occurrence of p in t. There various standard algorithms which give the exact match in time $O(mn)$ to $O(nlog(m))$. But we cannot employ exact string matching as for searching in music, we have to satisfy following two constraint:

1. **Approximate or fuzzy melody matching:** We chiefly want approximate matching as we cannot expect to have two pieces of melody to be exactly same (or one being exact subsequence of another). There are various reasons to support this assumption. As we are extracting the melody sequence from audio data and not from some transcription written by the composer (or any other human), so there are errors in the input melody sequence. Even a single such error will completely give away the matching sequences if exact matching is used. Also, even if the the input sequences were perfect, without errors, we would want to tolerate some notes to be changed, dropped or added in one melody to match it to another.

2. **Transpose invariance:** A melody can be sung in different keys. The pitch spectrum also varies between individual singers, especially between male and female voice. The range of pitch is different for different musical instruments. Also from music theory we can conclude that it is the variations in pitch (or notes) with reference to some tonic (or key note) that makes music similar or dissimilar than the absolute pitches (or notes). For these reasons it is desirable that the melody based similarity search tolerate the pitch transpositions between the two pieces of music.

5.2 Approximate String Matching

Let t and p be two strings of length n and m respectively and $m \leq n$.
The approximate string matching depends upon a metric called **edit distance**. Edit distance is defined as the number of operations required to be done to convert a sequence p to t. Edit distance is a generic concept. There are various edit distances e.g. Levenshtein distance which counts the number of deletion, insertion and substitution to be done to match the two sequences, Hamming distance which only allows substitution. A modification to edit distance is attaching different weights to different operations.

Approximate matching is done by aligning sequence p to t by performing the allowed operations which gives the lowest edit distance. Thus we get a subsequence of t, p' which is obtained by performing some deletions, insertions and substitutions in p.

Mostly Dynamic Programming is used for approximate string matching. For strings of large lengths many approximate string matching algorithms use heuristics. For example FASTA or BLAST[9] are some of the sequence alignment algorithms for aligning DNA or Protein sequences.

Dynamic Programming maintains a matrix C of size m by n, where C(i,j) represent the the minimum distance between p[1..i] and t[1..j] and thus tells the best path (operations to be done to match the two strings) between p[1..i] and t[1..j].

The minimum value in the last row of C gives the last element in t which matches with p. From this point C(m,k) we find the matching path by recursively checking whether we came to this path by insertion, deletion or substitution till the first element of p is encountered. Note the matrix C gives us the information about which operation is best locally in O(1) time. So finding path has just linear complexity i.e. it is no extra work than finding the cost matrix which is done in O(mn).

5.3 Transpose Invariance

We have another constraint to be satisfied, i.e. transpose invariance. There are standard algorithms to do the approximate string matching. Also we can easily incorporate the transpose invariance in exact string matching algorithms. But doing both things together is a non-trivial task. The standard

approximate string matching algorithms consider the string as a sequence of symbols. While here the string elements (notes) also have numeric property. Also the notes have their meaning in relation to their neighbors. So if we delete or insert or substitute a note in a string this relation should be appropriately managed. This is explained through following examples:

Let the pattern p = 40 41 43 44 47 49 46

Let dp be the difference sequence of p

$dp(i) = dp(i+1) - dp(i)$

dp = 1 2 1 3 2 -3

- **Example 1 :**

  t = 44 45 47 17 48 51 53 50

  Let dt be the difference sequence of t

  dt = 1 2 -30 31 3 2 -3

  Ideally p and t should match 7 notes with one deletion i.e. 17 i.e. dp and dt should match 6 notes with one deletion i.e. -30. Simple approximate matching matches dp and dt by 5 notes with 2 deletions which are -30 and 31. Correct match should be done by deleting note 17 in t i.e. deleting -30 and adjusting 31 to 1 in dt

- **Example 2 :**

  t = 44 45 47 17 18 21 23 20

  dt = 1 2 -30 1 3 2 -3

  dp and dt should match only first 2 notes or last 4 notes. This is shown in the figures. Simple approximate matching matches 6 notes incorrectly by deleting -30 in dt. For correct matching the adjustment should be done and propagated from -30 till the end of p. The method to do this adjustment (we call it residue) and its propagation is detailed

in the next section.

Both of the constrains described above can be implemented using standard algorithms. But doing both of them together is non-trivial as can be seen from the above examples. The next section details about the method employed to do the transpose invariant approximate string matching.

5.4 Method

First we take the difference sequences dp and dt of the input sequences p and t to be matched. The modified string matching algorithm is applied to dt and dp.

$dt(k) = t(k+1)t(k)$
$dp(k) = p(k+1)p(k)$

In Approximate String Matching using Dynamic Programming, we use a cost matrix C where C(i,j) is the cost of the best path up to dp(i) and dt(j). Likewise we use a new matrix R. R(i, j) stores the adjustment to be done to dt(j+1) for its comparison with dp(i+1). The adjustment at R(i, j) is based on the best path till (i,j). When comparing dt(j+1) and dp(i+1), the residue R(i,j) is added to dt(j+1). If note is added/deleted/substituted, appropriate changes are made in R.

The figures 7 and 8 show two cost matrices, various matching trails found, the best matching trail, the exactly matching notes and residue. The string p is along the vertical direction while t is along horizontal direction. Figure 7 matches a song against the aroha of its raga while 8 matches a song against its own modified excerpt.

5.5 Applications

1. **Searching in a song:** For a single song, the current method correctly finds the location of the matching position of the queried input.

2. **Song Retrieval:** Retrieving a song from a collection of songs matching the queried melody - the current implementation is based on matching the query with all the songs in realtime instead of indexing them. Therefore it takes impractical amount of time (linear to the size of collection) for searching a song.

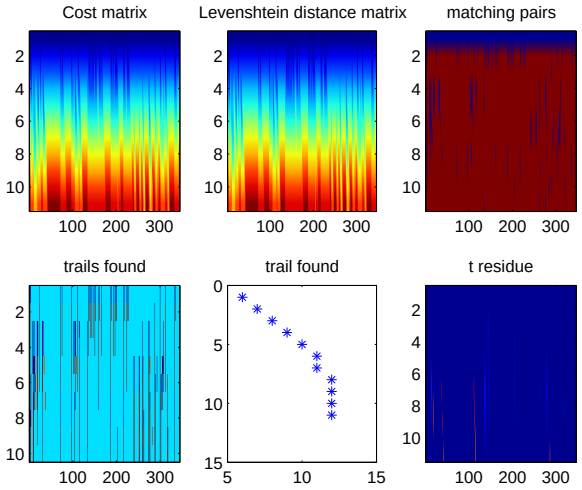


Figure 7: Pattern matching of a song with ascent of the same raga. The vertical direction represents the note of ascent string while horizontal direction represent the notes in the song.

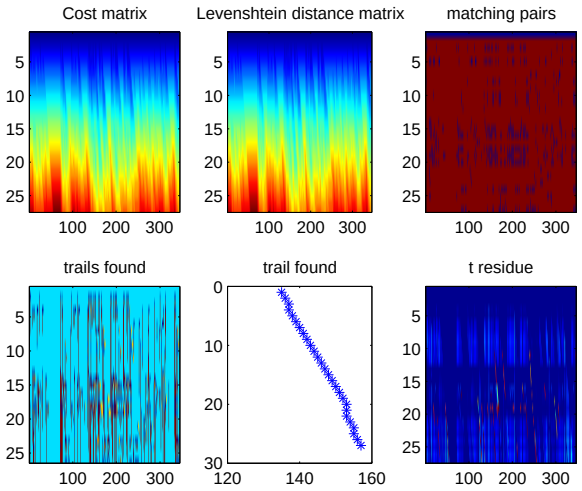


Figure 8: Pattern matching of a song with its own modified subsequence. Horizontal direction represent sequence of notes in the song. The vertical direction represents a short subsequence of the original sequence with some modifications applied to it.

3. **Song classification:** This is described in the next section.

5.6 Classification based on String Matching

This method uses the string matching component described earlier. Various patterns of a raga are collected and stored. These include the ascent (aroha), descent (avaroha) and catch phrases (pakads). Let t be the given query sequence of notes that is to be classified. Then each pattern for each raga is matched against t giving the cost (edit distance) of the match. The raga of the pattern with the lowest of these costs is declared as the raga of the query song.

The accuracy for this method is 24%. Though the accuracy is very low, it also indicates that the string matching based classification is viable. Here are some suggestions for further improving the accuracy:

- The fixed raga patterns are short and perfect while the query song t has errors in it. The errors in the present case are mostly the extra notes uniformly distributed over the entire string. Instead of penalizing these errors they should be handled using uniform time warping. This would detect the error (extra notes) rate in t and tolerate these errors without adding them to the final cost of matching as these are not the mismatch errors but noise in t.

- The cost structure should be modified. We have used the cost of matching only one, largest occurring pattern p in the query string t. Many substrings of the raga patterns occurring in large count should contribute to the overall cost (actually similarity). We already have the cost matrix C in the dynamic programming algorithm, some good measure for similarity using this matrix accounting for the substrings is easily computable, further improving accuracy. One such measure, for example, can be the sum of the number of occurrences in t of all the substrings of p.

6 Conclusion and Future Work

6.1 Conclusion

In this work, following components for the Music Information Retrieval System are implemented:

- Raga detection system based on two different methods - pitch distribution and note sequence.
- Transcription of music into sequence of notes.
- Approximate, transpose-invariant melody matching and music search.

6.2 Future Work

Further additions and modifications to this MIR system can include:

- The string matching based classification method can be improved as described in earlier sections by some modifications in string matching algorithm like uniform time warping.
- The sequences of notes should be indexed using some indexing structure like suffix trees etc for faster melody based retrieval of songs.

This work used melody as its basis. Besides melody, utilizing meter and rhythm will give another dimension to the music retrieval system.

References

[1] J P Bello, G Monti, and M Sandler. Techniques for automatic music transcription. In *International Conference on Music Information Retrieval*, 2000.

[2] A. Taylan Cemgil, Hilbert J. Kappen, and David Barber. A generative model for music transcription. In *IEEE Transactions on Speech and Audio Processing*.

[3] Parag Chordia and Alex Rae. Raag recognition using pitch-class and pitch-class dyad distributions. In *International Conference on Music Information Retrieval*, 2007.

[4] J. Stephen Downie. Music information retrieval *chapter*7. In *In Annual Review of Information Science and Technology 37, ed. Blaise Cronin, 295-340*, 2003. Available from http://music-ir.org/downie_mir_arist37.pdf.

[5] Martin McKinney and Jeroen Breebaart. Features for audio and music classification. In *International Conference on Music Information Retrieval*, 2003.

[6] H NEY. Dynamic programming algorithm for optimal estimation of speech parameter contours. In *IEEE Transactions on Systems, Man, and Cybernetics*, 1983.

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

[8] H. V. Sahasrabuddhe and R. Upadhyay. On computational model of raga music of india. In *Proceedings of Indian Music and Computer*, 1994.

[9] Altschul SF, Gish W, Miller W, Myers EW, and Lipman DJ. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.