SEGMENTATION AND RECOGNITION OF TABLA STROKES

Parag Chordia
CCRMA, Stanford University
660 Lomita Dr.
Stanford CA 94305
pchordia@ccrma.stanford.edu

ABSTRACT

A system that segments and labels tabla strokes from real performances is described. Performance is evaluated on a large database taken from three performers under different recording conditions, containing a total of 16,834 strokes. The current work extends previous work by Gillet and Richard (2003) on categorizing tabla strokes, by using a larger, more diverse database that includes their data as a benchmark, and by testing neural networks and tree-based classification methods. First, the time-domain signal was segmented using complex-domain thresholding that looked for sudden changes in amplitude and phase discontinuities. At the optimal point on the ROC curve, false positives were less than 1% and false negatives were less than 2%. Then, classification was performed using a multivariate Gaussian model (mv gauss) as well as non-parametric techniques such as probabilistic neural networks (pnn), feed-forward neural networks (ffnn), and tree-based classifiers. Two evaluation protocols were used. The first used 10-fold cross validation. The recognition rate averaged over several experiments that contained 10-15 classes was 92% for the mv gauss, 94% for the ffnn and pnn, and 84% for the tree based classifier. To test generalization, a more difficult independent evaluation was undertaken in which no test strokes came from the same recording as the training strokes. The average recognition rate over a wide variety of test conditions was 76% for the mv gauss, 83% for the ffnn, 76% for the pnn, and 66% for the tree classifier.

Keywords: instrument recognition, tabla, automatic transcription, timbre.

1 INTRODUCTION

Labeling tabla strokes is a timbre recognition problem that is very similar to instrument recognition. There is a growing body of work dealing with recognizing isolated percussion timbres (Herrera et al., 2003, 2002; Sandvold et al., 2004; Sillanpaa et al., 2000; Tindale et al., 2004). These studies have several motivations. Isolated percussion timbre recognition is useful for automatically searching or categorizing large collections of sounds, which can be used for multimedia search or organizing sound libraries in sample collections. It can also be used to label monophonic audio. Beyond that, it is hoped that the isolated tone recognition problem will give insights into identifying components of sound mixtures. There are many applications, such as searching and editing audio files, where we would like to identify and label percussion sections in polyphonic music.

Tabla stroke transcription is one of the few real-world transcription tasks where monophonic timbre recognition is genuinely useful. Although there are simultaneous strokes, these are conceptualized as single entities. Sometimes, notes will overlap because of ringing, but tabla is essentially a single stream of constantly changing timbres. In most music, we are faced either with a solo passage from one instrument, or a mixture of instruments.

2 INTRODUCTION TO TABLA

Tabla is comprised of a pair of drums, a treble drum, referred to as the tabla or *dayan*, and the bass drum called the *bayan*. The *bayan* is made of copper and sometimes terracotta. The right-hand *dayan* is a tapering cylinder carved from a block of dense wood. Each of the drums is covered with goatskin. A unique feature of the tabla is the tuning paste called the *syahi*, which is applied to the center of the *dayan*, and off-center on the *bayan*. When a membrane stretched over a resonating body is struck, there is generally no clear sense of pitch because the sound produced is rich in inharmonic overtones. When properly applied, the *syahi* causes the alignment of some of the inharmonic partials, giving the *dayan* a clear sense of pitch if struck correctly (Fletcher and Rossing, 1998).

The two drums of the tabla produce many different timbres. Many of these sounds have been named, forming a vocabulary of timbres. The naming of strokes has facilitated the development and transmission of a sophisticated solo repertoire. In addition to the rhythmic complexity of tabla music, it is its timbral beauty and diversity that distinguish it from other percussion instruments. Tabla sounds can be roughly classified into the following three major sound groups:

1. Ringing bell-like tones played on the treble drum. These tones are distinguished by a clear sense of pitch, sharp attack, and long sustain. *Ta*, *tin* and *tun* are examples.

2. Resonant bass strokes played on the bass drum. This is the stroke *ghe*. Unlike other sounds, *ghe* is the only sound that is continuously varied. The tabla player modulates the pitch by controlling the tension on the skin of the bass drum using the base of his palm.

3. Closed, crisp sounds. These sounds have sharp attacks and decays and sound damped. *Kat* played on the *bayan*, and *te*, *tak*, *dhe*, and *re*, which are played on the *dayan*, are examples of this family.

In addition to these, strokes can be played simultaneously forming a compound stroke. Typically, a stroke from the *dayan* is combined with a bass tone (*ge*) or a closed tone (*ke*) on the *bayan*. The most common compound strokes are *dha* (*na* + *ghe*), *dhin* (*tin* + *ghe*), and *dun* (*tun* + *ghe*). These are conceptualized as distinct strokes and not as mixtures.

In total there are approximately twenty different timbres. Isolated examples of each sound can be found at http://ccrma.stanford.edu/ pchordia/ismir05/. In most compositions, a smaller subset of the total number of available timbres is used. This is similar to tonal music in which certain notes are emphasized to create a tonal framework. The mapping between timbres and symbols is not one-to-one: one name can refer to several timbres depending on the context, and a single timbre can be referred to by more than one name. In this work, each distinct timbre is given a unique identifying label.

Individual strokes are combined into larger phrases, such as *terekite*, *denegene*, *kitetake*, *gegenage*. These combinations are akin to words in language; they combine hierarchically to make larger phrases. Because of these building blocks, certain strokes tend to follow other strokes. This sequential structure can be modeled as a Markov process.

3 RELATED WORK

The closest work is that of Gillet and Richard (2003) who describe a tabla transcription system. Segmentation was done by simple amplitude thresholding. They reported 98.3% segmentation accuracy, with the majority of errors coming from false negatives. In their system each stroke is parametrized by the mean, variance and relative weight of four frequency peaks. This was done by modeling the magnitude spectrum using a Gaussian mixture model (gmm). The gmm was used not as a probability distribution, but rather as a simple way to parametrize the shape of the frequency spectrum: the position of the peaks, their bandwidth, and their relative weight. In this way, each stroke was represented by a 12-dimensional feature vector.

Classification was done using nave Bayes, 5-nearest neighbor (5-nn), or by kernel density estimation. In the first, the feature vector was modeled as being drawn from a multivariate gaussian distribution. In a 10-fold cross-validation experiment the best reported accuracy was 83% using 5-nn in a 10 category problem. By performing language modeling, using a hidden Markov model (hmm), they were able to attain a 93.6% recognition rate. This was done by treating the stroke labels as hidden states, and the feature vectors as emissions from those states. Transitions between strokes were estimated directly from the training data. For the sequence of feature vectors the Viterbi path was calculated giving the sequence of strokes. In a more difficult generalization test, in which training and test samples are from different conditions, performance dropped to 79.8% without language modeling, and 90.2% with language modeling.

Early work in percussion transcription was done by Schloss (1985) who described an amplitude-based thresholding algorithm for onset detection. A method to classify several different conga strokes based on the relative energy in different frequency bands was described. Recognition rates were not reported.

Sillanpaa et al. (2000) attempted to identify random mixtures of up to 3 sounds from five different drum classes. A total of 129 drum sounds were used. An iterative template matching scheme was used in which the nearest neighbor was found and subtracted. The procedure was repeated until the energy fell below some threshold. For isolated notes, the correct label was given 87% of the time, with extra drums (false positives) occurring 32% of the time. In two drum mixtures, both drums were found 49% of the time, and at least one drum 100% of the time. For mixtures of three, the recognition rates were 8% for all three, 60% for two, and 100% for at least one.

Recent work in percussion transcription has been done by Herrera et al. (2003, 2002). In the first work, isolated percussion notes from 9 different instruments were classified. A wide variety of spectral and temporal features were used, including spectral centroid, mel-frequency cepstral coefficients (mfccs), log-attack time and zero-crossing rate (zcr). Recognition of 90% using a k-nearest neighbor (k-nn) classifier was reported, however the database only included 634 examples from commercial sample CDs. In the second study, the number of timbres was increased to 33 and the total number of samples to 1976. A recognition rate of 85% was attained using a k-nn algorithm in a 10-fold cross-validation test.

Tindale et al. (2004) described a system that classified sounds produced by striking different areas of a snare drum, in total producing seven distinct timbres. Using a database of 1260 strokes recorded, under very uniform recording conditions, a recognition rate of 95% was achieved with a feed-forward neural network. Testing was done using a leave-one-out protocol. The high recognition accuracy was likely dur to the similarity of many of the strokes in the database.

A more difficult percussion recognition task was undertaken by Sandvold et al. (2004), who collected 1136 training samples from 25 different recordings and 1419 test samples from 17 different recordings. The challenge came from the diversity of the database, and the fact that real performance data were used. Sandvold showed that performance could be dramatically improved (20-30%) by manually labeling a small subset of the tones in the test set.

van Steelant et al. (2004) dealt with mixtures of either the snare drum or the bass drum with three other percussive sounds, leading to six possible timbres. The system attempted to detect the presence or absence of the bass and snare drum in each of the six possible mixtures. Using support vector machines they reported accuracies of 94% and 96% for the bass and snare drum respectively. When evaluating these results, it is important to note that only six timbres are used, and that only a decision about the presence or absence of the bass and snare drums is taken. Nevertheless, for popular music, identifying bass and snare drum patterns can reveal much about the structure of the piece. Features such as the relative energy in different frequency bands, mfccs, temporal centroid, and measures of the shape of the frequency spectrum were used.

4 TABLA STROKE DATABASE

Original recordings were made of two highly skilled tabla players[1] using different drums, with a good quality condenser microphone on a DAT recorder. The recordings were made in a small room, about 13X13, which was mildly reverberant. Each player played several *qaidas*[2] at slow and fast tempos. This was a qualitative term that players were free to interpret, although constrained by typical performance practice. This was done to ensure that the system was evaluated on the full range of durations that occur in real performances. *Qaida* was used because of its central role in tabla repertoire, because it can be extensively elaborated, and because it is simpler to annotate than fully improvised sections. Most *qaidas* emphasize a subset of all possible strokes, so an attempt was made to chose *qaidas* that taken together included a wide variety of strokes. Each recording was then manually segmented and annotated. Annotation was very time consuming and limited the total amount of data that could be gathered.

Table 1: Stroke Database

Source	id #	# strokes	# targets
Gillet	1	1678	10
	2	1821	10
	3	2216	10
Tanmoy	4	2475	16
	5	2612	16
Ravi	6	3035	9
	7	2997	9
Total		**16834**	

The annotation also made clear some of the difficulties of evaluating the automatic transcription systems. Even within a well-established musical tradition naming conventions differ. As we noted, different names are given to the same timbre and the same name can be given to different timbres. Unless one has manually auditioned each stroke this can introduce structural errors that cannot be improved by improving the classification system. Also, even for experienced musicians, it can sometimes be difficult to assign the label from the sound alone (i.e. without any visual reference). Perhaps the most subtle and subjective issue is how to deal with playing errors. When should a stroke be considered an error, and when should it be considered a plausible variation of a particular stroke? If we discard too many mishit strokes we lose the ability to learn these variations. The intuitive answer is to consider a stroke a mishit, rather than a variation, if it is unlikely to occur in the future. If it is a common error, then it should be included. This sort of judgment requires extensive domain knowledge by the annotator.

In addition to these data sets, a data set that was used previously in Gillet and Richard (2003) was obtained from the authors, hereafter referred to as the Gillet data set. The goal was to use the data to benchmark the current system. The Gillet data set is comprised of three data sets recorded under different conditions. The first data set uses a low quality tabla under low fidelity recording conditions, the second set uses a higher quality tabla and better microphone, while the last set is less closely miked, leading to some reverb in the recording.

In general, the Gillet data are recorded in lower fidelity than the previous data sets, and the player is clearly an amateur. While low fidelity is potentially an interesting challenge for the classification algorithm, the tabla players level produces some obvious mistakes that degrade the quality of the data set. Two problems emerged. First, the played stroke does not always sound like the intended stroke, whereas the database annotation records the intended stroke label. Second, the consistency of sound production for an amateur player is significantly less than a professional, leading to greater acoustic variation within the same stroke class. However, since these problems were also faced by Gillet and Richard, it is still valid to use their data as a benchmark. It is, however, an important factor in evaluating the absolute error rate.

In addition to labeled strokes, the Gillet data contained onset location information and recorded the order of strokes. Table 1 summarizes the data sets. Example phrases from each of the sources can be found at http://ccrma.stanford.edu/ pchordia/ismir05/.

5 METHOD

The system works in the following three stages: 1) the signal is segmented by finding onset points 2) the segmented strokes are parametrized by computing features that summarize the timbre 3) a classifier that has been trained on a subset of the data is used to label the strokes. Each step is described in detail below.

5.1 Segmentation

Successful transcription depends on accurate segmentation. Because the system makes decisions at the note level, and not at the frame level, it is important that each stroke is

[1]The tabla players were Tanmoy Bose and Ravi Gutala. Tanmoy Bose is an internationally known artist, who performs with Pandit Ravi Shankar. Ravi Gutala is a senior disciple of Pandit Swapan Chaudhari.

[2]A theme and variation form

segmented at the correct point. For tabla music, the segmentation only needs to find onset points. Offset points are much less salient, and the inclusion of some tail silence will not negatively impact the classifier. False positives (detecting onsets that are not there) will result in the stroke being divided in unexpected places, possibly leading to the omission of a critical portion of the stroke. False negatives (failing to detect onset points) will lead to segments that contain more than one stroke. For these reasons, even if the classifier is working well on isolated notes, inaccurate segmentation will lead to an increased error rate. Although not discussed here, accurate detection of onsets is also important for beat and tempo detection.

The simplest and most common approach to onset detection is some form of amplitude thresholding. The gist of these algorithms is to formalize our visual intuition that onset points correspond to regions where the amplitude envelope suddenly rises. A version of this approach that was used by Schloss (1985); Gillet and Richard (2003). A problem that is often encountered in these approaches is false positives caused by ringing. To correct this, the threshold change value has to be set correctly, and a rest parameter, which disallows onsets that are too close together, has to be used. Generally, adjustment of these parameters varies with the recording and has to be done manually. More recently, this approach has been refined by breaking the signal into various frequency bands and then combining the amplitude thresholding information from each band to take a decision. This is described in Klapuri (1999), and a frequency-domain implementation of this is found in Goto (2001).

The amplitude thresholding methods described above were implemented but did not yield satisfactory results. At the optimal point on the ROC curve, false negatives and false positives each exceeded 3%. Duxbury et al. (2003) propose a system that combines amplitude and phase information. The algorithm works by using the steady-state assumptions to predict the modulus and argument of the complex amplitude. Consider the Fourier transform of the signal: for the k-th bin, the modulus gives the amplitude of the sinusoid, and the argument gives the phase. By definition, in a steady-state region, the phase changes at a constant rate that is given by radian frequency. So at the next time frame, we expect the amplitude to be the same and the phase to equal the phase at the previous time frame plus the radian frequency multiplied by the time step. The expected change can be estimated by taking the phase difference between the previous two frames. So $\phi_k(m) = \phi_k(m-1) + [\phi_k(m-1) - \phi_k(m-2)]$ in steady-state regions.

The deviations in each bin of the complex amplitude from the expected complex amplitude at time m are summed to compute a new detection function. The new detection function is less noisy, more clearly showing onset points. This leads to more accurate segmentation, without requiring significant parameter optimization. This system had a false negative rate of less than 2% and a false positive rate of less than 1%.

5.2 Feature Extraction

Most current instrument recognition studies use a large number ($>$ 100) of temporal and spectral features Essid et al. (2004); Herrera et al. (2003, 2002); Livshin and Rodet (2004); van Steelant et al. (2004). Commonly used spectral features include spectral centroid, mfccs, measures of spectral shape such as skewness and kurtosis, spectral rolloff and others. In addition to features that are calculated over the whole note, some features are calculated in frames so that their evolution over time can be used. Because most classification techniques require feature vectors of equal length, we cannot simply append the values of the features in each frame to form a large feature vector since notes vary in duration. Because of this, the temporal evolution of features is often summarized by the average amount of change and the standard deviation of that change. Alternatively, feature vectors from consecutive frames can be modeled using hidden Markov models, but this method has been used infrequently to date.

In order to avoid overfitting, dimension reduction is usually performed on the full feature set. We limited the initial set of features by carefully examining the magnitude spectra and amplitude envelopes of tabla strokes from each category. The main temporal features were temporal centroid (which gives the balancing point of the amplitude envelope), attack time, and zero-crossing rate. The spectral features used were spectral centroid, skewness (of the magnitude spectrum), kurtosis, and thirteen mfccs. In total, thirty-one initial features were calculated for each stroke. These were further reduced using principal component analysis.

5.3 Classification

Four different classifiers were trained as described below. After being presented with the feature vectors of each test sample, the classifiers returned the corresponding labels.

5.3.1 Multivariate Gaussian (mv gauss)

The feature vector was modeled as being drawn from a mv gauss distribution. The labeled samples from the training data were used to estimate the mean and covariance of the likelihood distribution for each class, i.e. $Pr(data|class_i)$. The covariance matrix was computed on the pooled data, rather than being estimated separately for each class. The prior probability of each class was determined by calculating the relative proportion of strokes in the training database. Using Bayes rule, the likelihood and prior probabilities were multiplied, and divided by the evidence, to calculate the posterior probability for each class: $Pr(class_i|data)$. The label was selected according to Bayes rule; the class that had the largest posterior probability was chosen.

5.3.2 Feed-forward neural network (ffnn)

Neural networks, rather than explicitly estimating the likelihood and prior distributions, use training examples to compute a non-linear mapping from the feature space to categories. The great advantage of such architectures is their ability to non-linearly combine features to represent complex decision boundaries. The decision boundaries need not be convex or contiguous.

Neural networks are comprised of nodes and associated weights. Inputs to each node are multiplied by a weight and fed to a non-linear function that emits a value close to one if the input exceeds some threshold, and close to zero otherwise. In a ffnn, the input layers consists of as many nodes as there are features. For every node, each feature is multiplied by a weight and summed before being fed to the non-linear function. Each node in the input layer is connected to each node in a hidden layer, where the process is repeated. The output layer has as many nodes as there are classes. In the ideal case, when presented with a sample from a given class, the network outputs a value of one in the corresponding output node, and zero elsewhere.

The ffnn is essentially a collection of weights and non-linear functions. It turns out that the particular choice of non-linearity is usually not critical, leaving the central question of how to set the weights. This is done by the back-propagation algorithm. Weights are initially assigned randomly. During training, a sample is presented to the network, and each output node emits a value. Since we are aiming for a one at one of the nodes, and zeros at the other nodes, we can calculate the difference between the outputted values and the desired values. The square of this is our error function. The weights are adjusted to minimize the error function. In particular, we perform a gradient descent in the weight space. Details of the back-propagation algorithm can be found in Duda et al. (2001).

Architecture plays a crucial role in the network's ability to generalize. The number of hidden layer nodes must be selected so that the ffnn is sufficiently expressive without overfitting the data. In the ffnn, the main architectural decisions concern the number of hidden layers and the number of nodes in each hidden layer. In this study, ffnns with 1 and 2 hidden layers were tried. Both worked well, the first with 50 hidden nodes and the second with between 10 and 20 nodes in each layer.

5.3.3 Probabilistic neural network (pnn)

A pnn is a neural network that is used to non-parametrically learn a probability density function. In classification problems, training samples are used to learn the posterior distribution of each class. The pnn is a neural network implementation of Parzen windows density estimation (Duda et al., 2001).

A pnn consists of input units, pattern units, and output units. For classification problems, the number of input units is equal to the dimension of the feature vector. A pattern unit consists of weights that are the normalized feature values of a training sample. Pattern units are connected to each of the input units. Each output unit represents a category and a pattern unit connects to the one output unit of the same category. Training consits of setting the weights of each pattern unit equal to the feature values of the corresponding training sample.

When a feature vector is presented to the network, the inner product of the feature vector and the weights of a given pattern unit are calculated: $\mathbf{w^t x}$. If the sample is similar to the sample represented by the pattern unit, this value will be large. This is repeated for each of the pattern units that emit a non-linear function of this value: $f(\mathbf{w^t x})$. Each category unit sums the outputs of the pattern units that connect to it. The test sample is given the label of the category unit that emits the largest value. It can be shown that if $e^{(\mathbf{w^t x}-1)/r^2}$ is used as the function, where r is a constant the controls the spread, then the pattern unit will emit a value that is equal to the probability that the test point was generated from a Gaussian centered on the training example (Duda et al., 2001).

5.3.4 Tree classifier

A binary tree based classifier was constructed. Tree based classifiers work by considering questions that divide the training data. The tree consists of nodes and branches. At each node a question is posed that divides the data. They are essentially formalizations of a series of questions, giving them a certain intuitive appeal. At the end, we would like to have all the examples at a terminal node belong to the same class. This is called a pure node. At each node the goal is to split the data in a way that maximally purifies the data. A common way of doing this is by using an entropy criterion. A pure node has zero entropy, while a node that has several classes uniformly distributed will have normalized entropy value of one. Splits are chosen that maximally reduce entropy. Although purity can generally be increased by further splitting, overly complex trees will lead to overfitting. To avoid this, traning is usually halted when the error rate begins to increase on an independent verification set.

5.3.5 Hidden Markov model

An attempt was also made to improve performance using language modeling. As noted before, tabla strokes have a sequential structure. We can ask what the probability of the next stroke is given the previous stroke, or the previous several strokes. These transition probabilities can be estimated from training data in which the sequence of strokes is known. For example, if we are estimating the transition probabilities from *dha*, we simply look at the relative number of times it continues to other strokes. Based on these transition probabilities, we can calculate the likelihood of a given sequence of strokes. Most sequences will be highly unlikely.

In our problem, however, we cannot directly observe the stroke labels; we only see a sequence of feature vectors. The goal is to decide what sequence of strokes was most likely to have produced the feature vectors we observed. For each stroke class, there is a certain probability that the feature vector was drawn from it; this is simply the likelihood probability, i.e. $Pr(data|class_i)$. We need to combine the transition probabilities with the likelihood probabilities in order to calculate the most likely path. This is done by the Viterbi algorithm. For the sake of brevity, the reader is referred to Duda et al. (2001); Rabiner and Juang (2003) for an explanation. Language modelling was used with the mv gauss classifier and the ffnn. In the latter, the values of the output nodes were normalized and treated as estimates of the posterior probabilities.

6 EXPERIMENTAL RESULTS

6.1 Test Conditions

To better understand the generalization abilities of the system, two testing methods were used. The first method used 10-fold cross-validation in which each experiment was repeated ten times and the average error rate reported. In each trial, 90% of the samples were randomly selected for training, and the remaining 10% were used for testing.

The second method tested the ability of the classification algorithm to generalize to completely new data. In this protocol, which we will refer to as novel generalization, training samples and test samples were taken from different data sets. In cross-validation tests (including leave-one-out), it is likely that for each category an example that is very similar to each test example will be found in the training database. For example, the tabla stroke database is composed of strokes that have been taken from various tabla compositions. Because phrases are often repeated, there is a high likelihood of getting several very similar strokes in a given database, that is, strokes that are played on the same instrument, by the same player, under the same recording conditions, and with nearly the same articulation. This is partially responsible for the high recognition rates that are reported in studies using this test method. Nevertheless, this does not necessarily invalidate this test method, as the method gives us the best-case performance scenario and is useful in predicting recognition rates in situations in which we know that the training and test sets are similar.

Table 2: Recognition accuracy (10-fold CV)

Database	mv gauss	pnn	nnet	tree
1	0.86	0.90	0.93	0.79
2	0.86	0.95	0.96	0.84
3	0.86	0.93	0.95	0.80
1 2 3	**0.92**	**0.93**	**0.93**	**0.80**
4	0.86	0.90	0.90	0.74
5	0.86	0.89	0.87	0.64
4 5	**0.86**	**0.89**	**0.87**	**0.72**
6	0.94	0.94	0.95	0.84
7	0.91	0.94	0.94	0.82
6 7	**0.92**	**0.94**	**0.94**	**0.84**
average	**0.89**	**0.92**	**0.92**	**0.78**
1 2 3 4 5	0.79	0.90		
1 2 3 6 7	0.83	0.77		
4 5 6 7	0.85	0.93		

Novel generalization is significantly harder because the classification algorithm has to generalize across different instruments, players, and recording conditions. This point has been made in many previous instrument recognition studies (Essid et al., 2004; Gillet and Richard, 2003; Livshin and Rodet, 2004; Sandvold et al., 2004).

6.2 Recognition accuracy

Results using the cross-validation experiments are given in Table 2 and from the novel generalization experiments in Table 3.

The recognition rate for the combined Gillet data was 93% for both the ffnn and the pnn, and 92% for the mv gauss. This was significantly higher than the Gillet and Richard results (83%) using no language modeling, and nearly equal to the 93.6% recognition rate obtained with language modeling. The best recognition rate for the Ravi data was 94% using the ffnn and pnn. On the Tanmoy data, which contained the most number of classes (16), the recognition rate was 89%. The recognition rate averaged over several experiments was 92% for the mv gauss, 94% for the ffnn and pnn, and 84% for the tree based classifier. In the cross-validation experiments, all the classifiers performed similarly, except for the tree, which did 10% worse on average.

Table 3 make it clear that the two testing paradigms lead to different recognition rates. Recognition rates for the novel generalization experiments ranged widely, from a low of 15% to a high of 95%. The average recognition rate over a wide variety of test conditions was 76% for the mv gauss, 83% for the ffnn, 76% for the pnn, and 66% for the tree classifier. Even when taking into account the widely varying results, the ffnn seems to generalize better. When Gillet sets 1 and 2 were used for training, and 3 for testing, the accuracy was 88% for both the mv gauss and the nnet. This compares with 80% in the Gillet and Richard study without language modelling and is similar to their figure of 90% accuracy using language modeling. When Gillet sets 2 and 3 were used for training, and 1 for testing, the accuracy was 88%, as compared with 78% reported by Gillet and Richard without language modeling, and 88% with language modeling.

One clear trend that emerges from the data is that training on Tanmoy and Ravi and testing on the Gillet data led to poor results, ranging from 29-45% using the mv gauss classifier. On the other hand, performance ranged between 60-80% when the Tanmoy and Ravi sets were used to test each other. The best results were when the data came from the same recording conditions. For example, the recognition accuracy reached a high of 95% when the short duration Ravi strokes (database 7) were used to train a ffnn classifier that was used on the long duration Ravi strokes (database 6).

An interesting point is that, although a large and varied database is important for generalization, what is more important is having strokes in the training database that are similar to strokes in the test database. Increasing the size of the database can decrease performance. For example, using the Tanmoy data to classify samples from the Ravi database gave a 65% recognition rate. Adding the Gillet data to the traning set decreased the accuracy to 56%.

Table 3 shows that language modelling did not improve classification performance in any cases where it was tried, and it marginally decreased performance in most cases.

7 DISCUSSION

The system's performance is similar to that reported by Gillet and Richard (2003), and substantially better in the case when no language modeling is used.

Table 3: Recognition Accuracy (Novel Generalization)

Train	Test	mv gauss	mv gauss hmm	nnet	nnet hmm	pnn	tree
2 3	1	0.88	0.88	0.87	0.81	0.82	0.71
1 3	2	0.88	0.89	0.89	0.86	0.85	0.73
2	3	0.86	0.84	0.82	0.78	0.86	0.73
1 2	3	0.88	0.89	0.88	0.84	0.87	0.76
5	4	0.84		0.89		0.85	0.71
4	5	0.80		0.85		0.80	0.66
7	6	0.91		0.95		0.92	0.84
1 2 3 4 5	6	0.56		0.65		0.44	0.56
4 5	6	0.65		0.84		0.76	0.59
6	7	0.84		0.90		0.87	0.78
1 2 3 4 5	7	0.58		0.63		0.41	0.45
4 5	7	0.58		0.79		0.67	0.51
4 5	6 7	0.61		0.78		0.70	0.53
average		**0.76**		**0.83**		**0.76**	**0.66**
1 2 3 4 5	6 7	0.57		0.65			0.49
4 5 6 7	1	0.37		0.45			
4 5 6 7	2	0.45		0.60			
4 5 6 7	3	0.29		0.15			
4 5 6 7	1 2 3	0.39		0.20			
4 5	1 2 3	0.26					
6 7	1 2 3	0.37					
1 2 3	6 7	0.50					

The different results in the no-language modeling case are most likely due to the incorporation of more sophisticated features. Another factor is that, in the Gillet study, timbres that were aurally the same but nominally different retained their nominal labels. For example, *ti* and *te* are almost always acoustically indistinguishable. In the current study, they were given the same label, whereas Gillet and Richard (2003) retained the original labels. Because *te* almost always follow *ti* in notation, this distinction could only be captured by the language model. In both studies, it is likely that, for the Gillet data, the upper performance bound is close to being reached. Most of the remaining errors are database issues, such as mishits and mislabeled strokes.

Table 4 shows a typical confusion matrix. This is from a cross-validation experiment using all the Gillet data with the mv gauss classifier. There are three common confusions, *na* and *dha*, *tun* and *dun*, and *kat* and *te*. The first and second are due to the fact that, in many *na* strokes, the bass stroke is still ringing from a previous stroke, because *dha* = *na*+*ge*. This makes it very easy to confuse *na* with *dha*. This is also the case with the *tun*, *dun* confusion. Because of the high prior probability of *te*, several of the other closed stokes are sometimes confused with it. Although not shown, confusion matrices for other data sets were similar. In the Tanmoy data, the most common confusion was again between strokes and their compound version (i.e. the stroke + *ge*). Again, the high prevalence of *te* in the training data led to many strokes being occasionally incorrectly classified as *te*.

The posterior probabilities generated by the mv gauss model were examined to try and understand why language modeling did not help. We would expect language modeling to have an affect where the posterior is distributed among two or more strokes. In most cases, the posterior distribution is highly concentrated on one stroke, with the probably of all other strokes being much less than one percent. Because the posterior and the transition probabilities are multiplied in the Viterbi algorithm, the total probability is dominated by the posterior in those cases. A highly concentrated posterior indicates the confidence with which the classifier is assigning a label. Examination of the posterior distributions revealed no significant difference in the entropy of correctly and incorrectly labeled strokes. In cases where the posterior distribution is spread out, the entropy will be greater. The posterior was usually highly concentrated even in cases where the classifier failed, which means that the transition probabilities of the language model did not change the classification decision.

Auditioning some of the incorrectly classified strokes revealed that many of the errors were due to ambiguities in the data. For example, the confusion matrix shows that many na strokes were classified as *te*, even though the two strokes are theoretically distinct. *Na* is a ringing, pitched stroke, wheras te is a non-resonant, non-pitched stroke. Listening to some of the misclassified *na* strokes revealed that some them were played poorly, resulting in a slapping rather than ringing sound. Although, not every error was auditioned, it is likely that language modeling did not help because many of the errors had to do with the data source, and not the accuracy of the classifier.

8 FUTURE WORK

The current work is the start of building an automatic tabla transcription system. To accomplish this, a much larger,

Table 4: Confusion Matrix for Gillet Data

	dha	dhin	na	tin	tun	dhun	dhe	dhec	re	rec	nec	kat	ge	te	tas
dha	**357**	12	20	0	0	0	0	0	0	0	0	2	2	2	0
dhin	2	**27**	0	3	0	13	0	0	0	0	0	1	1	1	0
na	76	0	**418**	1	0	0	3	0	0	0	2	4	0	1	0
tin	0	0	0	**7**	0	0	0	0	0	0	0	4	0	1	2
tun	0	0	0	0	**106**	11	0	0	0	0	5	0	3	5	0
dhun	0	3	0	0	0	**74**	0	0	0	0	1	0	4	0	0
dhe	0	0	0	0	0	0	**27**	0	7	0	1	0	0	3	0
dhec	0	0	0	0	0	0	1	**3**	0	1	0	3	0	7	0
re	0	0	0	0	0	0	5	3	**35**	2	3	0	0	6	0
rec	0	0	0	0	0	0	2	0	0	**9**	0	0	0	2	0
nec	0	0	0	0	4	2	0	0	0	0	**39**	0	0	12	0
kat	7	0	0	5	0	2	1	0	0	0	0	**347**	3	16	0
ge	3	2	0	0	0	14	0	0	0	0	7	20	**400**	2	0
te	1	0	0	1	12	0	2	4	5	8	51	11	5	**988**	0
tas	0	0	0	0	0	0	0	0	0	0	0	0	0	1	**7**

more varied database needs to be created. The total number of strokes will likely have to be increased by an order of magnitude in order to capture the sufficient stroke variety. This will also require drawing strokes from many different players and recording conditions.

Once this is done, there will be sufficient data to develop a hierarchical Markov model. The current work does not take into account the temporal evolution of features. This could be modeled with an hierarchical hmm.

In almost all real performances of tabla solo, the tabla is accompanied by a melodic instrument such as the harmonium or *sarangi*. Future work will attempt to label strokes from recordings that contain such accompaniment. Success will likely depend on successfully expanding the database and using dynamic features.

ACKNOWLEDGEMENTS

My sincere thanks to Olivier Gillet and Gael Richard for providing their data to me.

REFERENCES

R. Duda, P. Hart, and D. Stork. *Pattern Recognition and Scene Analysis*. John Willey, 2001.

B. Duxbury, J. Bello, M. Davies, and M. Sandler. Complex domain onset detection for musical signals. In *In Proc. Digital Audio Effects Workshop (DAFx)*, 2003.

S. Essid, G. Richard, and D. Bertrand. Musical instrument recognition based on class pairwise feature selection. In *In Proc. of the 5th ISMIR Conf.*, 2004.

N. Fletcher and T. Rossing. *The physics of musical instruments*. Springer, New York, 1998.

O. Gillet and G. Richard. Automatic labelling of tabla signals. In *In Proc. of the 4th ISMIR Conf.*, 2003.

M. Goto. An audio-based real-time beat tracking system for music with or without drum sounds. *Journal of New Music Research*, pages 159–171, 2001.

P. Herrera, A. Yeterian, and F. Gouyon. Automatic classification of drum sounds: a comparison of feature selection and classification techniques. In *Proc. of Second Int. Conf. on Music and Artificial Intelligence*, pages 79–91, 2002.

P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of unpitched percussion sounds. In *Proceedings of the Audio Engineering Society*, pages 1378–91, 2003.

A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *International Conference on Acoustics, Speech, and Signal Processing*, 1999.

A. Livshin and X. Rodet. The importance of cross database evaluation in musical instrument sound classification: A critical approach. In *In Proc. of the 5th ISMIR Conf*, 2004.

L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 2003.

V. Sandvold, F. Gouyon, and P. Herrera. Drum sound classification in polyphonic audio recordings using localized sound models. In *In Proc. of the 5th ISMIR Conf*, 2004.

A. Schloss. *On the Automatic transcription of percussive music- From acoustic signal to high-level analysis*. CCRMA, 1985.

J. Sillanpaa, A. Klapuri, J. Seppanen, and T. Virtanen. recognition of acoustic noise mixtures by combined bottom-up and top-down approach. In *In Proc. of EUSIPCO*, 2000.

A. Tindale, A. Kapur, G. Tzanetakis, and I. Fujinaga. Retrieval of percussion gestures using timbre classification techniques. In *In Proc. of the 5th ISMIR Conf.*, 2004.

D. van Steelant, K. Tanghe, S. Degroeve, B. De Baets, M. Leman, and J. Martens. Classification of percussive sounds using support vector machines. In *Proc. of the annual machine learning conference of Belgium and The Netherlands*, 2004.