TOWARDS EXPRESSIVE MELODIC ACCOMPANIMENT USING PARAMETRIC MODELING OF CONTINUOUS EXPRESSIVE ELEMENTS OF MUSIC IN A MULTI-ATTRIBUTE TRIE FRAMEWORK

A Thesis
Presented to
The Academic Faculty

by

Trishul Mallikarjuna

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Music Technology in the
Georgia Tech Center for Music Technology

Georgia Institute of Technology
17th December 2010

TOWARDS EXPRESSIVE MELODIC ACCOMPANIMENT USING PARAMETRIC MODELING OF CONTINUOUS EXPRESSIVE ELEMENTS OF MUSIC IN A MULTI-ATTRIBUTE TRIE FRAMEWORK

Dedicated to my father

Śrī. Mallikārjuna B. S.

* * *

I hope music technology will soon enable musicians like him

to make music without needing to use technology.

ACKNOWLEDGEMENTS

This thesis draws from various associated pieces of my work over several semesters at the Georgia Tech Center for Music Technology (GTCMT), and would not have been possible without the inspiration, support and advice from my faculty, colleagues, friends and family.

I'm indebted to Dr. Parag Chordia for the inception of this work and for advising me throughout it, and also for his efforts in keeping me on the better part of the track. This thesis is a culmination of a major proportion of the work I did under him at GTCMT. I extend my thanks to Dr. Gil Weinberg for his advice and for giving me a chance to experiment with various platforms of music technology available at GTCMT; the computational framework used in this work was conceived and developed for use with the musical robots at the center under his guidance. I'm grateful to Dr. Jason Freeman for his advice and for encouraging me to experiment with various techniques of interactive music while introducing me to the 'what' and 'why' of music technology; many of the Max/MSP modules used in this work were conceived and developed for the projects under his guidance.

I thank my colleagues Mr. Brian Blosser, Mr. Avinash Sastry, Mr. Aaron Albin, Mr. Ryan Nikolaidis and Dr. Guy Hoffman for their contributions at various stages. The preliminary simulations were built on earlier work by Brian. Avinash and Aaron have worked on sections of the framework dealing with trie modeling, as parts of a bigger project. Work by Guy & Ryan with the robot *Shimon* has inspired some of the techniques, while the web scripts for the evaluation survey were derived from those used by Guy.

I also feel grateful to all other colleagues for their support in various related projects through the years, and to my other friends and family for keeping me on my toes and for keeping their expectations up. I hope that the work, the thesis, the further developments with the framework and the musical results will meet at least a part of the expectations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Elements of continuous variation such as tremolo, vibrato and portamento enable dimensions of their own in expressive melodic music in styles such as in Indian Classical Music. There is published work on parametrically modeling some of these elements individually, and to apply the modeled parameters to automatically generated musical notes in the context of machine musicianship, using simple rule-based mappings. There have also been many systems developed for generative musical accompaniment using multi-order probabilistic models of discrete musical elements such as MIDI notes and durations, many of them inspired by computational research in linguistics. There however hasn't been a combined approach of parametrically modeling expressive elements in a multi-order probabilistic framework. A real-time computational framework is accordingly presented here, which uses a multi-attribute trie / *n*-gram structure to model parameters like frequency, depth and/or lag of the expressive variations such as vibrato and portamento, along with conventionally modeled elements such as musical notes, their durations and metric positions in melodic audio input. This work proposes storing the parameters of expressive elements as metadata in the individual nodes of the traditional trie structure, along with the distribution of their probabilities of occurrence. During automatic generation of music, the expressive parameters as learned in the above training phase are applied to the associated re-synthesized musical notes. This modeling scheme is aimed at being used to provide automatic melodic accompaniment in a performance scenario. The parametric modeling of the continuous expressive elements in this form is hypothesized to be able to capture deeper temporal relationships among musical elements and thereby is expected to bring about a more expressive and more musical outcome in such a performance than what has been possible using other works of machine musicianship using only static mappings or randomized choice. A system was developed on Max/MSP software platform implementing this framework, which takes in a pitched audio input such as human singing voice, and produces a pitch track which may be applied to synthesized sound of a continuous timbre. A rule-based and highly user-customizable sub-system that seamlessly integrates the processes of segmentation of notes and extraction of their parameters using a real-time state-machine based approach was also developed. The system was trained and tested with several vocal recordings of North Indian Classical Music, and a subjective evaluation of the resulting audio was made using an anonymous online survey. The results of the survey show the output tracks generated from the system to be as musical and expressive, if not more, than the case where the pitch track generated from the original audio was directly rendered as output, and also show the output with expressive elements to be perceivably more expressive than the version of the output without expressive parameters. The results further suggest that more experimentation may be required to establish the efficacy of the framework employed against using randomly selected parameter values for the expressive elements. This thesis presents the scope, context, implementation details and results of the work, suggesting future improvements.

CHAPTER I

INTRODUCTION

Elements of continuous variation, such as tremolo, vibrato and portamento, enable dimensions of their own in expressive melodic music. Vibrato refers to the microtonal oscillation of the pitch of a voice about the mean pitch level of a musical note, and tremolo refers to a similar oscillation of the energy level or loudness of the voice. Portamento refers to the continuous gliding shift of the pitch of a voice from the vicinity of one musical note to that of another, used instead of a sudden jump between the two levels. Glissando, another such expressive variation, refers to a shift of pitch from one note to another, similar to a portamento, but with discrete steps of pitches in between instead of a continuous glide – practiced with instruments with discrete note levels such as a piano. These expressive variations, mainly the tremolo, vibrato and portamento, are produced as a natural affect of physical characteristics of the singing voice or the instrument, or are voluntarily imparted by the singer or the player of the instrument. These are extensively used in many styles of music such as the Indian Classical Music (ICM), as tools of improvisation [49][34][19] and thus to aid in imparting the psychoacoustic and emotional effects of music [12].

With the advent of digital computers, there have been many systems developed over the past few decades to make machines understand music, improvise in various styles and interact musically with fellow humans [45][46]. An example of such a system, with a computational design similar to that described in this thesis is the Continuator [40]. Such systems typically involve generative musical accompaniment using computational models of discrete musical elements such as MIDI notes and durations. These models typically use probabilistic data structures such as Markov chains [2], which can mathematically predict the occurrence of a certain discrete musical event with an associated measure of likelihood, given the history of a number of events until a point of time, thereby being able to model deep temporal dependencies between the events.

To develop or extend such a system to styles of music like Indian Classical Music involving extensive use of continuous expressive elements, the continuous elements need to be modeled parametrically, i.e., quantitative parameters that succinctly describe the elements, such as the frequency of vibrato or the functional coefficients of a portamento curve, need to be determined in the same way the discrete elements are modeled as mentioned above. There have been works of research to parametrically model some of these continuous elements individually; for example, there is published work in modeling of vibrato/glissandi [30] and their removal from audio [43] in the context of audio analysis. There have also been works using the models limitedly in rule-based generative accompaniment [48], where the parameters are extracted from audio recordings in an offline manner and are later applied to machine-generated musical notes in a rule-based fashion, involving only a set of simple static dependencies between the parameters and the discrete elements such as note numbers and durations. There however hasn't been a system with a combined approach – trying to integrate the modeling of parameters of continuous expressive elements into a framework of probabilistic modeling of discrete musical elements such as notes and durations, thereby allowing the parameters to be modeled in a close context of the deep temporal dependencies among the discrete musical events.

A candidate for such a combined approach is presented by this thesis – a system with a real-time computational framework that extracts and seamlessly embeds parameters (like frequency, depth, duration and/or shape) of vibrato and portamento inside an n-gram Markov-based model of musical notes, their durations and metric positions, alternatively termed a multi-attribute prediction suffix tree framework, or a multi-attribute trie framework. The system can in turn generate musical audio output that is stylistically similar to the source audio used to generate the model, aimed at being used to enable a computer to provide automatic melodic accompaniment in a performance scenario. This method of parametric modeling of the continuous expressive elements is intended to bring about a more compelling and expressive musical outcome in such a performance than what has been possible using other works of machine musicianship. The system takes in a pitched audio input such as human singing voice, and analyzes it in real-time to determine the parameters of vibrato and portamento as well as the discrete notes and durations. It then integrates this information into the model, so that the model can further be used to generate notes and expressions in the same style.

Audio recordings of North Indian Classical Music (NICM) performances are taken as the source material for training, development and evaluation of the system as ICM demonstrates a good use of vibrato and portamento [34][49][19] and thus would serve as a suitable example of a style of music that would be reasonably modeled using the approach. However, the same framework can be applied to model and generate music in other styles of music that use continuous expressive variations such as vibrato and portamento. While vibrato and portamento have been treated as the principal expressive elements for the purposes of developing the system, the modeling scheme developed for vibrato in the musical pitch domain can equivalently be applied to the sound velocity domain to model tremolo.

The end goal of this work is split in two ways. Within the context of this document and the work done in it, the goal is to present and evaluate a new approach to model expressive music in the field of Music Information Retrieval (MIR). In a longer term and a broader context, the goal is to design an interactive music system for live performance that incorporates expressive elements of music and improves upon previous such systems.

This document presents the context of the research, the design and development of the system, and the initial experiments with the system including its evaluation. Chapter 1 (this chapter) has provided an introduction to the context and motivation for the development of the system. Going ahead, Chapter 2 briefs on some of the related work that the development of the system was based on. Chapter 3 gives an overview of the system architecture. Chapter 4 presents the techniques used in tracking the pitch of an incoming audio stream and in segmenting it into various stable notes and other regions, along with an introduction to the parameters defined in the system and explaining how they were extracted. Chapter 5 explains the *n*-gram modeling structure used in the system and how the above parameters were embedded in it. Chapter 6 makes brief notes about how the audio was synthesized and what interface was given to the user in the developmental stage of the system. Chapter 7 briefs over some of the initial evaluation experiments conducted with the system and discusses their results, and Chapter 8 presents some of the improvements that may be considered for the system in the future, before Chapter 9 concludes. The system developed in the work described in this thesis is named ‘PortaVibra’, and is referred henceforth in the document as the ‘current system’ or as the system in the ‘current work’.

CHAPTER II

RELATED WORK

The system described in this thesis draws inspiration and techniques from two streams of related work: one dealing with the ways to parametrically model continuous elements of expressive variations such as vibrato and portamento, and the other dealing with ways to probabilistically model discrete elements of music such as notes and durations.

2.1 Parametric modeling of expressive musical elements

There have been many works on parametric modeling of expressive musical elements, most of them predominantly dealing with vibratos.

Schoonderwaldt et al. developed a generative, rule-based model for expressive violin vibrato [48]. Arguing that a mere regular periodic modulation of pitch generally yields unsatisfactory results sounding both unnatural and mechanical, and that an appropriate control of vibrato rate and vibrato extent is a major requirement of a successful vibrato model, they used measurements of vibrato as performed by professional violinists to generate vibrato rate and extent envelopes consisting of a limited number of linear parts (Figure 1), which were in turn used to control a sampled violin synthesizer. They found that the note-by-note values of vibrato extent and sound level were positively correlated for long notes in the analyzed performances, and that sound level tended to increase until vibrato started on the within-note level, and included these observations in their model. This work served as an initial guide for the current work, by introducing a system that involved parameter extraction and applied those parameters to a synthesis engine. This system however extracts the parameters using offline analysis of performances, while in contrast the current system was developed to extract the parameters in real-time as the audio input is fed into the system. Further, the above work applies the parameters to machine-generated notes in a rule-based

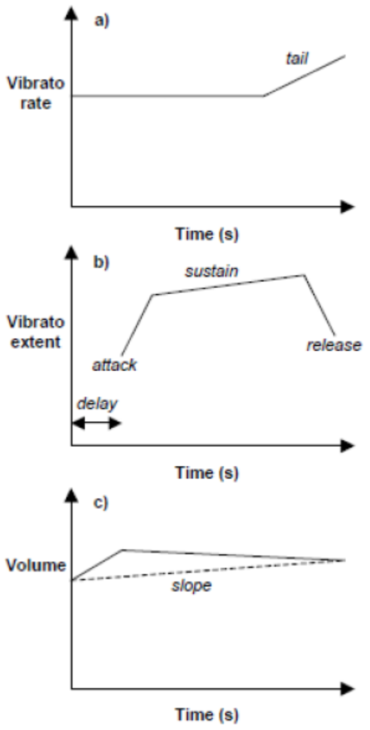


Figure 1: This figure illustrates the pitch envelopes derived and used by Erwin Schoonderwaldt (Reproduced from Schoonderwaldt [48])

fashion, involving only a set of simple static dependencies between the parameters and the discrete elements such as note numbers, while the current system was developed to model dynamic and deeper temporal dependencies between the parameters and the discrete elements such as the notes, their durations and their positions in the metric grid.

Desain and Honing worked on systematically modeling continuous aspects of music performance, specifically vibrato and portamento, using methods of machine learning such as simulated annealing for pattern recognition while deriving the parameters of continuous aspects [17]. Simulated annealing is a field-proven technique often used in pattern recognition tasks where an attempt is made to find out the parameters of a pattern function that matches well with the observed data, by initially starting with a pseudo-random search for the optimal values in the parameter space and then slowly converging on to an optimal point by adjusting a control variable for ‘randomness’. While simulated annealing and similar methods can work as good techniques for pattern recognition in an offline scenario such

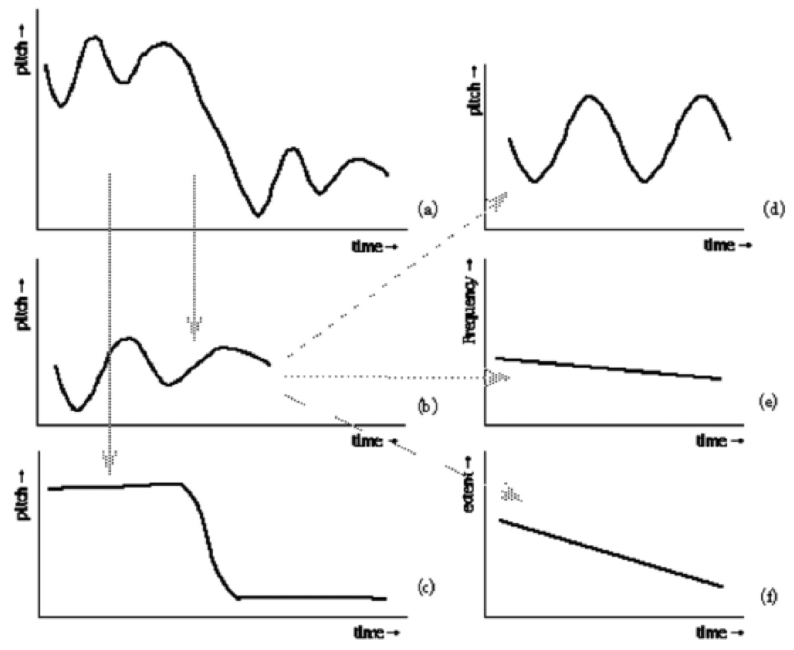


Figure 2: This figure shows the breakdown of a pitch track proposed and used by Desain in an early work. (Reproduced from Desain [16])

as in the above work, being able to find good matches for the parameters of vibrato and portamento curves, it can be computationally expensive and thus inefficient in a real-time scenario. Hence, several state-machine based heuristics that are more straightforward – and thereby faster in performance – were adopted in the current work for segmenting the pitch track into note regions and to extract the values of the parameters.

Desain and Honing further presented some of the classical hypotheses on the question of how vibrato is adapted to global tempo [17]. Some of the hypotheses are that vibrato is not adapted at all – that it just continues on across note boundaries without being adjusted in any way, or that it is only in the note onset that vibrato is controlled and restarted at a specific phase. Some other hypotheses are that the duration of the notes are adapted to the vibrato rate through delaying of the offset such that a whole number of vibrato cycles fit in the note duration, or that it happens the other way around – with the vibrato rate being adapted to the note duration such that a whole number of cycles fits in. Yet another hypothesis is that the adaptation is only done towards the end of the note.

Following up on the above, Desain et al. investigated the rhythmic aspects of vibrato across different musicians and tempi [18]. In their attempt to answer the question whether vibrato rate is adapted to tempo, they established a confirmed relation between tempo and vibrato rate for instruments with a low vibrato rate, and found an increasing vibrato rate towards the end of notes except in case of a very high vibrato rate. They found meter to be target of synchronization for the instruments that showed some scaling of vibrato rate with tempo. They concluded the temporal aspects of vibrato to be important aspects of vibrato, and to be under considerable control of the performer, being important in their relation to tempo and to the length of notes of the music, but being insignificant in their relation to meter. Some of the above conclusions were taken into consideration while designing the current system. The system stores parameters of vibrato and portamento as a metadata packet along with each note in a manner that establishes a direct relation between the parameters and duration of each note. Further, several parameters related to portamento in the current system are expressed in terms of beat durations, thereby allowing them to scale with tempo, similar to the scaling of vibrato rate with tempo as suggested above. Scaling of vibrato rate with tempo in the current system was also considered, but was not used in the initial implementation as that particular aspect required a separate line of analysis of the exact relation between tempo and vibrato rate, which was left for future improvements. Also, in the current system, even though the rate of vibrato for each note is quantized and stored in the model with a value very close to that extracted from the source audio, during generation of output notes the vibrato rate is adjusted to the nearest convenient value such that the appropriate number of cycles of vibrato fit within the duration allocated for vibrato, with an appropriate phase and direction in line with the adjacent events in the output audio. This was done following up on one of the above hypotheses, while variation of vibrato rate towards the end of each note was left as a future improvement.

Further, Timmers and Desain describe a series of experiments dealing with the key questions about vibrato in performance of different instruments and present the qualitative opinions from musicians about their performance of vibrato as well as quantitative scientific measurements about the same, while discussing the relation between them [52]. The musicians were reported to suggest variations in vibrato in relation to their explicit expressive intentions, such as phrasing, contrasting first and second half, accentuating of certain notes, and tension and relaxation of the music. Analysis of vibrato in the performances was reported to have shown a general, considerably strong effect of musical structure on amplitude, vibrato rate and extent, a general consistency of vibrato characteristics over repetitions implied by the strong effect, and a limited relatedness between amplitude, vibrato rate and extent. Further, a significant relation between metrical stress and vibrato rate, and that between phrase position and amplitude was reported. Inspired by the above observations, in the current system, the vibrato rate and depth were modeled independent of each other, while the system easily supports modeling them together in case such a modeling is desired. The system was also designed such that the metadata containing all the parameters are stored in the model in direct relation to the position of the notes in the metrical grid, similar to the relation between phrase position and amplitude reported above.

Elsewhere, Herrera and Bonada describe their method for vibrato extraction and parameterization in the Spectral Modeling Synthesis framework [43], which is another notable related work.

The above works involving parametric modeling of vibrato try to address some of the key questions about the physical attributes of vibrato and its practice: Is there a specific vibrato before and/or after a note transition? Does global tempo influence vibrato rate? How does vibrato develop during a note? Is the vibrato rate adapted to the meter of the music (coordinated with tempo and note timing)? The answers and conclusions presented in the above works establish vibrato as an important expressive element in music in several contexts of western music performance – defined by the type of instrument, style of improvisation and the choices of the composer and the performer. The experiments have found that the physical attributes of vibrato vary by player, instrument, style/genre, tempo, composition, and other parameters, and that the vibrato waveforms have various shapes. The experiments have also found that there is a confirmed relation between tempo and vibrato rate for instruments with a low vibrato rate, that the vibrato rate increases towards the end of notes, and that the temporal aspects are important in their relation to tempo and to the length of notes of the music, but insignificant in their relation to meter. Some of these results are used to make practical working assumptions in the system described in this thesis, as briefed in individual paragraphs above.

While the related works described above were primarily conducted in the context of western music and while the above hypotheses and observations may not hold good in the context of Indian Classical Music, owing to the absence of similar research and experiments in ICM, the observations and hypotheses mentioned above were taken as the starting point in the implementation of the current system, and it is hoped that they would get refined and adjusted appropriately for ICM in future experiments.

2.2 Probabilistic modeling of musical melody

While the systems described above dealt with parameterization of expressive elements, on the other hand, there have been many systems developed for computational melodic modeling and for interacting with fellow humans through automatic generative music[45][46]. A good example with a computational design similar to the system described in the thesis is the Continuator [40] built by François Pachet, a system enabling musical Interaction between machine and a human with the ability to model the musical style of the human player. The system would break down a melodic sequence into short phrases, learn the temporal relations between the notes by calculating probabilities of occurrence of specific sequences of notes and storing them in a tree-like structure, and try to continue a test sequence using the structure built. The system used discrete raw MIDI symbols – but did not use derived types or more abstract relations such as transpositions; the learning was limited to short sequences of notes – such as phrases of 8 notes; further, the system did not have a way of generically learning the structure of metrical timing in the phrases.

Inspired by systems such as the Continuator and by related mathematical methods such as the Markov chains, a simple initial implementation of the system described in this thesis was made [56] at Georgia Tech Center for Music Technology (GTCMT) in order to be used with *Shimon* – the marimba-player robot of Georgia Tech designed to be perceptual, improvisational and socially interactive[58]. The system enabled the robot to analyze incoming MIDI notes from a human performer playing a piano, and to respond with stylistically similar melodic improvisations in real-time[55]. This implementation used a 2^{nd} order Markov chain – in which a table of probabilities of occurrence of the notes was developed based on the observed occurrences of those notes in the input stream – in relation to two previous notes, and this table of probabilities was used to generate the notes statistically during improvisation. This implementation however was not able to model note relationships that spanned over more than two previous notes, and thus a framework that supported larger and variable order was desired. Further, even though this initial implementation could imitate the note durations and timings of the human performer and scale them with tempo, it did not have a statistical scheme for modeling the note timings, and needed a framework that can effective model them along with their relation to the note values themselves.

Several systems that expand upon the concept of Markov models and prove to be useful in contexts similar to the above have been proposed. Pearce describes the construction and evaluation of several statistical models of melodic structure in music perception and composition [42], discussing the variable length *n*-gram modeling approach to music as well as entropy-based evaluation of a model's performance. He establishes a high correlation between such a model and human judgements of melodic continuation – particularly dealing with western melodies. Rodriguez and Buenos describe a tree-like modeling structure called Multi-Attribute Predictive Suffix Graph (MPSG) as an extended form of Probabilistic Suffix Tree (PST), wherein each node is labeled with a tuple consisting of multiple symbols or indices for different musical attribute [53].

The current system tries to expand upon the techniques in the above works, by adopting a similar framework of variable-length *n*-gram modeling with multiple attributes, to also train on continuous elements. This results in a modeling data structure shaped like a tree, with discrete elements stored as symbols in each node of the tree along with their probabilities and also their count of observations – essentially a prefix tree – also termed a 'trie' as derived from the word 'retrieval' due to the data retrieval properties of the structure, while the parameters of continuous elements are stored in each node as attributes – alternatively called metadata. Thus the structure can be aptly described as being a multi-attribute trie framework. The parameters in the metadata packet in each node of the trie were further designed to be modeled as a 1^{st} order Markov chain – enabling the distribution of the probability of their occurrence to be effectively captured. In the context of a larger project, parts of the framework developed were also used at GTCMT to analyze and model North Indian tabla compositions [11].

Conklin and Witten describe certain multiple viewpoint systems for music prediction [14], in which predictions resulting from different viewpoints consisting of models built for different semantic representations of musical elements (such as discrete pitch classes versus melodic intervals) are compared, combined and weighted against each other to make the final prediction. By relying on multiple viewpoints, such systems have been shown to result in better predictions than those possible with individual viewpoints in several musical contexts [14][11]. An example for a multiple viewpoints system in the context of musical melody is one using two viewpoints to model musical notes: one expressing notes in terms of discrete pitch classes and another in terms of melodic intervals in comparison to the previous notes. Such a multiple-viewpoint system was also integrated into the current framework. However, this part of the system incorporating multiple viewpoints is not used for the initial experiments with the system, and is hence not described in detail in this thesis.

Elsewhere, Dubnov et al. describe Audio Oracle: a new algorithm for fast learning of audio structures [20], building upon the concept of suffix trees and suffix links that are extensively used in language modeling, with a focus on fast retrieval of melodic associations and fast recombination of the melodic elements in a real-time performance scenario. In the initial experimentation of the current work, an exhaustive variable-length *n*-gram modeling approach with a 'lossless' representation of melodic associations was adopted, while leaving the focus on fast retrieval and recombination such as with Audio Oracle to future work.

Thus, in essence, this thesis proposes and describes a system of modeling parameters of continuous expressive elements of music such as vibrato and portamento along with the discrete elements such as notes and their durations and positions, as applied to, developed with, and tested in the particular context of Indian Classical Music. While various parts of the system are derived from earlier implementations as referenced above, the novelty of the system lies in the way the parameters of continuous expressive elements are embedded in the trie, encapsulated as a packet of metadata in each node of the trie, with each parameter in turn provisioned to be modeled probabilistically. The forthcoming chapters describe the architecture, design and functionality of the system.

CHAPTER III

SYSTEM ARCHITECTURE

The system developed in the work is designed to take in a pitched audio input such as human singing voice, and to produce a synthesized continuous pitch track that can be mapped to the sound of a continuous-timbre instrument such as violin/saxophone. It can be broadly divided into 3 parts as below; each is described in detail in the upcoming chapters. Figure 3 shows a block diagram of the functional units in the system, along with the user-settable variables and the flow of variables among the units. A functional description and the purpose of most of the units and variables can be derived from the forthcoming chapters. Other units and variables are presented in the illustration to assist in understanding and using the Max/MSP objects that are put up at the project website [35].

3.1 Pitch tracking, note segmentation and feature extraction

On the input side, incoming audio is processed to extract the instantaneous pitch level, using audio signal processing objects and several steps of pre-processing. This pitch track is then segmented into stable, transient and rest regions, which denote the regions of stable notes, transitions from one note to another, and rests, respectively. This information, along with the shape of the pitch track is further used to extract the key expressive parameters, such as the frequency and amplitude of vibrato, the duration of a transient/portamento, etc.

3.2 Multi-Attribute Trie Modeling

The result of pitch track analysis is used to build a variable-length n-gram model of the input audio. The notes of the stable regions, their durations and locations are used as the core symbols for the nodes in the *n*-gram trie model, while the parameters of expressive variations are added as independent metadata at each node of the trie. Each parameter is further stored as an array of quantized values along with the corresponding probability distribution. In the preliminary implementation of the system, the system can continuously receive the incoming audio, generate the pitch track from it, analyze the pitch track and train the computational model with parameters for every new note detected.

3.3 Sound Synthesis, interaction and interface

This involves using the *n*-gram model built above to make predictions of a continuing sequence of melodic notes along with parameters for expressive variations, which are in turn fed into a real-time sound synthesizer that produces voiced audio output. In the preliminary implementation of the system, the user indicates – through a soft switch on the interface – when the system may continue the sequence using the model developed so far, and the system then generates stylistically similar musical output using the voice chosen in the synthesizer. This output can potentially be used to make the computer system accompany a human musician in other modes of improvisation, involving directly imitating the human, embellishing upon what the human performs, harmonizing or improvising differently – all in the style perceived using the *n*-gram modeling. These improvisations may be done in many temporal modes of interaction: either in synchrony with the human, or with a delay, or in call-and-response manner, or as a solo playing. Adopting the system for a performance involving such improvisations and temporal modes of interaction is left for future work.

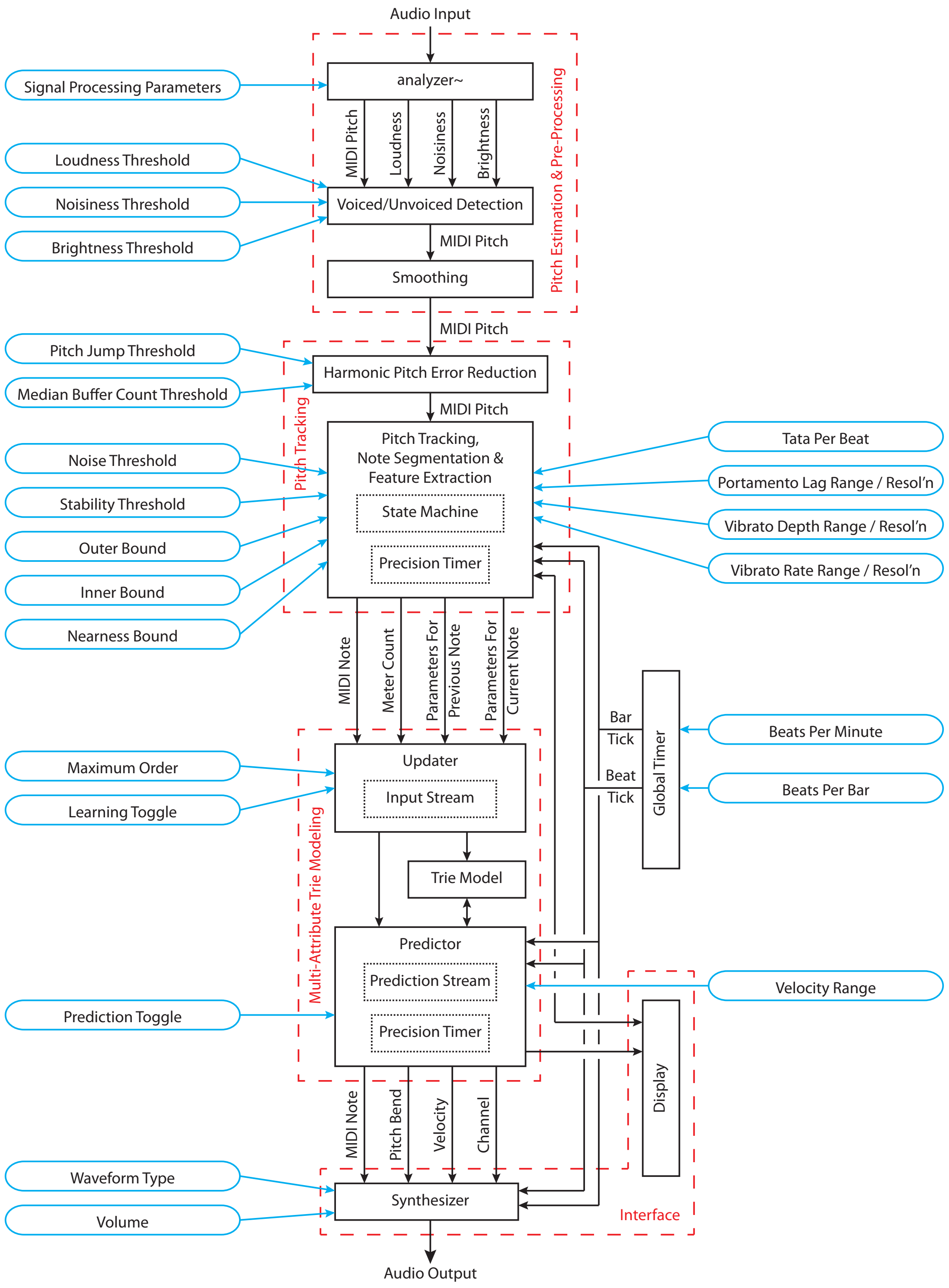


Figure 3: A block diagram illustrating the system architecture; red dashed blocks represent the high-level functional units; blue rounded elements represent user-settable variables; black solid blocks represent low-level functional units; black arrows with corresponding text represent signals and variables flowing across functional units

CHAPTER IV

PITCH TRACKING, SEGMENTATION AND FEATURE EXTRACTION

This chapter introduces the techniques used to extract the pitch from input audio and to determine the stable and transitional regions inside the pitch track. It further introduces the parameters of expressive variation that were chosen for the system, and discusses how they were extracted from the pitch track.

4.1 Preliminary work on MATLAB

As the first step, a preliminary non-real-time analysis was conducted on the MATLAB software platform to make a visual observation of the pitch tracks and to determine the feasibility of parametric modeling of expressive parameters. Several recordings of performance by a few vocalists of North Indian Classical Music were analyzed and pitch tracks were extracted in non-real-time. SWIPE′, an algorithm developed by Arturo Camacho [7] was used for pitch estimation. SWIPE′, expanded as Sawtooth Waveform Inspired Pitch Estimator, is an algorithm that tries to compare the frequency spectrum of the incoming audio to that of a sawtooth waveform in order to estimate its pitch level. Since the frequency spectrum of a human voice – especially when producing vowel sounds – is shaped similar to that of a sawtooth waveform, this algorithm is expected to yield good results when analyzing voices such as the human voice, as in the case of current work. The SWIPE′ algorithm additionally uses several other techniques of Digital Signal Processing (DSP) and digital arithmetic, including decomposition of the spectrum into Mel-Frequency Cepstral Coefficients (MFCCs) which have the affect of making the analysis similar to that done by human auditory system [38][15][39]. The pitch tracks were segregated into regions of

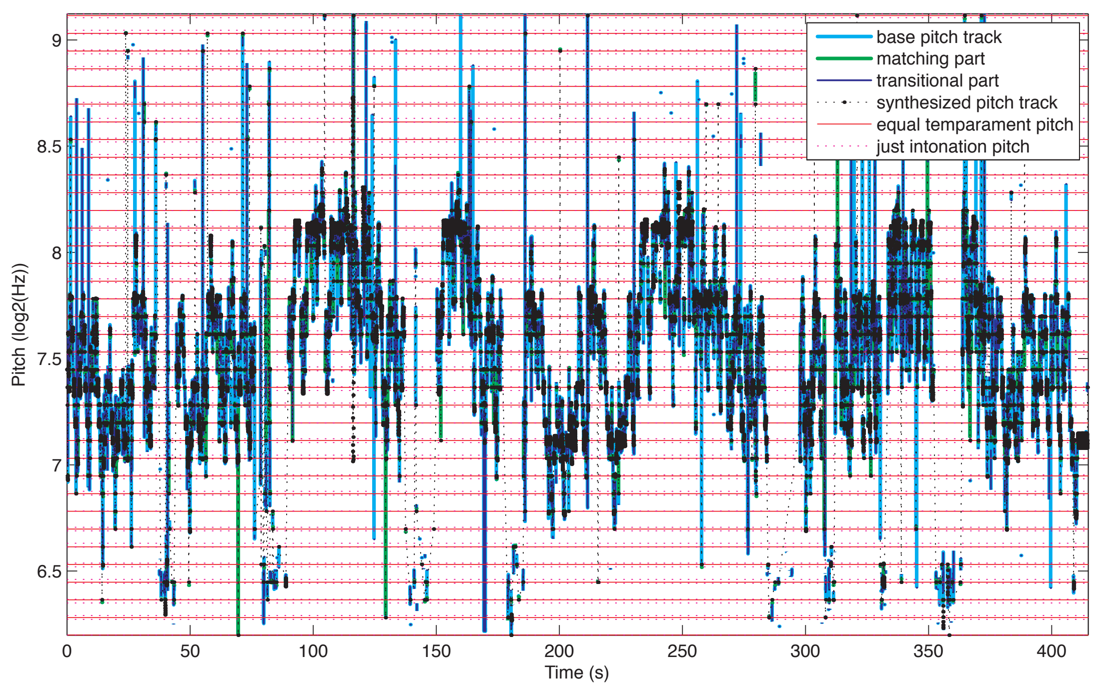


Figure 4: Pitch track of a full input audio track – from the preliminary MATLAB simulation

various stable notes and transitions, and parameters for a small number of expressive parameters including vibrato were extracted, using the techniques outlined in the following sections. The tracks were re-synthesized using simple waveforms such as a sinusoid and sawtooth as base. Expressive parameters were added to the synthesis, such as a sinusoidal wave for vibrato at the stable notes, using the vibrato depth and rate as determined earlier. An example of the original and synthesized pitch tracks of an audio track is shown with varying levels of zoom in Figure 4, Figure 5 and Figure 6; bluish lines are the original pitch tracks as output by SWIPE′, while the lines with black dots represent the synthesized pitch track. Example audio tracks are included in the project media [35]. The resulting synthesized audio tracks were found to be convincing representations of the original recording, as per the opinions of several colleagues at the Georgia Tech Center for Music Technology, and inspired the building of the real-time system for modeling expressive elements in melody and open evaluation of the system.

It may also be noted that the above pitch tracks contain a number of very high or very

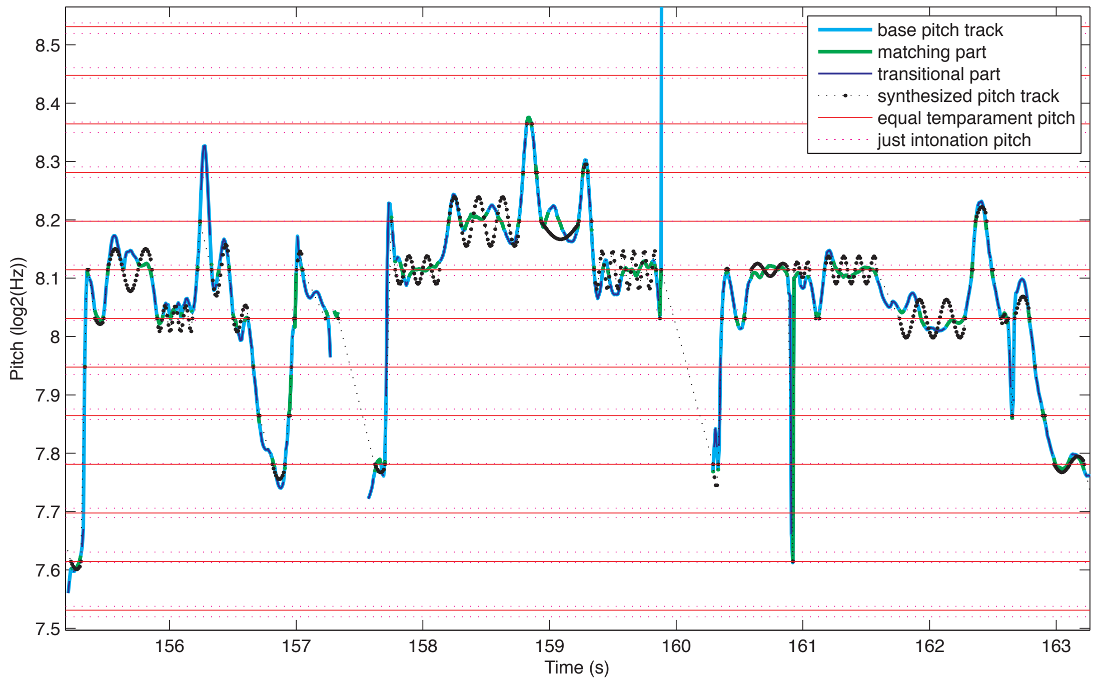


Figure 5: A zoomed-in region of the pitch track in Figure 4 – from the preliminary MATLAB simulation

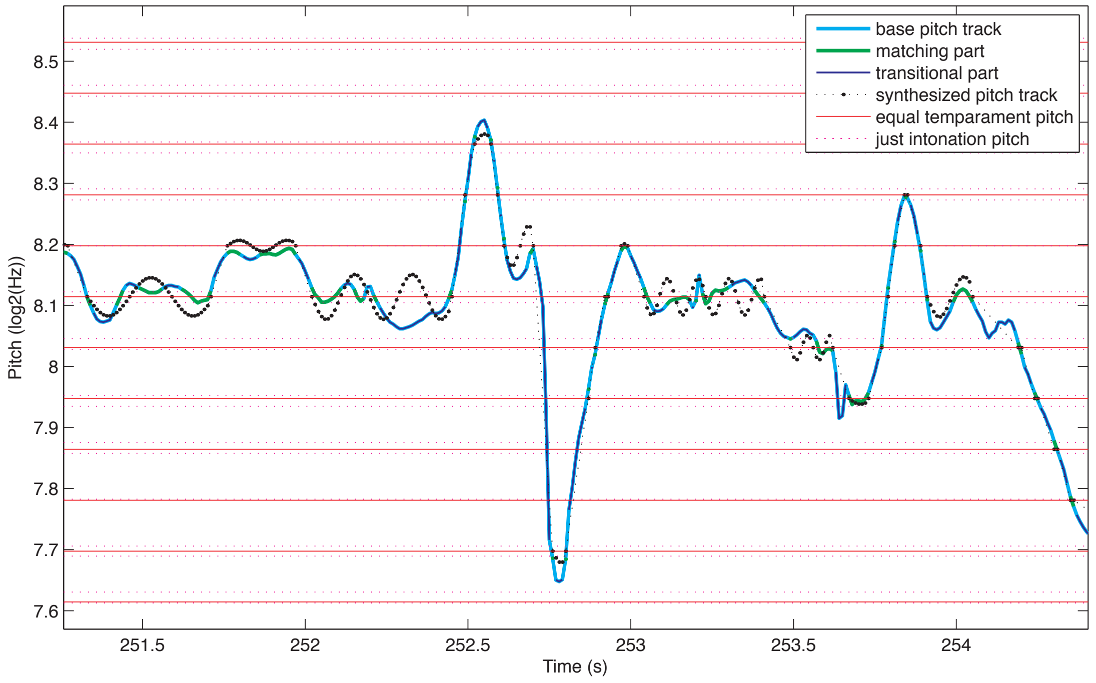


Figure 6: A closer zoomed-in view of a region of the pitch track in Figure 4 – from the preliminary MATLAB simulation

low impulsive notes – which occur due to errors in the pitch tracking process such as boundary errors or octave errors. Octave errors and similar pitch errors are where the pitch tracking algorithm outputs a pitch that is harmonically related as an integral multiple or submultiple, but not the same as the intended original pitch. Boundary errors are erroneous pitches that result due to the algorithm processing partial frames along the boundaries between different notes and those in between a note and an adjacent rest. To work around these errors, a heuristic was adopted in the current system, wherein each incoming pitch value is approved to be valid or rejected as an erroneous note based on how it compares to the median of a recent history of pitch values. A certain user-definable number of pitch values from the most recent past is kept in a running buffer, and its median is calculated. If a new incoming pitch differs from this median by more than a user-settable pitch limit (such as 9 or 12 semitones), the pitch is rejected as an erroneous one and not considered for further processing, nevertheless being put into the above queue of recent pitch values. This technique was found to avoid a majority of incoming erroneous pitches in the current system, if the above limits were tuned appropriately to each scenario depending on the quality of incoming audio, processing power or frame rate, and the quality parameters set in the pitch tracking algorithm.

4.2 Pitch Tracking

While the SWIPE′ algorithm used in the preliminary offline processing yielded generally good results, no real-time implementation of the algorithm was found during the development of current work. Since such an implementation of a pitch tracker was not the primary focus of the current work, in the real-time system on the Max/MSP platform, a Max/MSP object made by Tristan Jehan [29] was used alternatively on the input side. This object determined the pitch of incoming audio, along with several other attributes such as loudness, noisiness and brightness. The object, named analyzer~, based on an earlier MaxMSP object fiddle~ made by Miller Puckette [44], analyzes the frequency spectrum of frames of

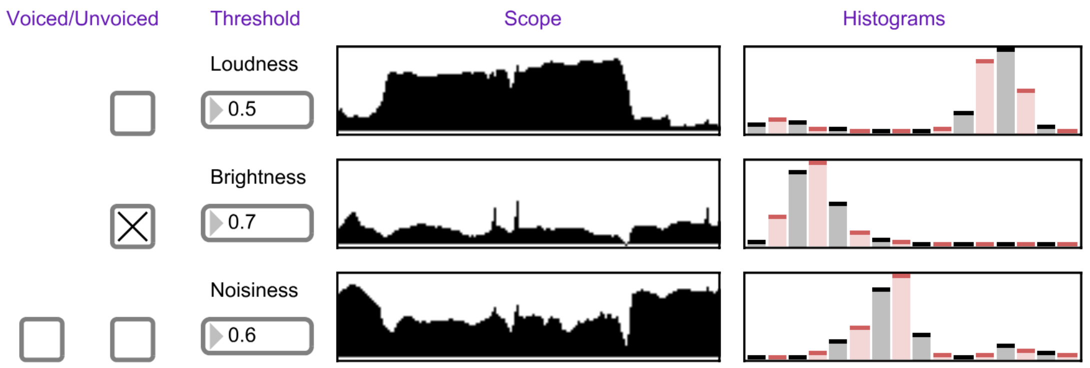


Figure 7: Plots and histograms of timbral quality outputs of analyzer~ Max/MSP object: Loudness, Noisiness and Brightness.

incoming audio to estimate the fundamental harmonic, or pitch, of the particular frame of audio, and outputs the pitch as a MIDI note number whenever successive frames result in a relatively stable pitch. Among a number of other parameters, the object allows the user to set the period (in milliseconds) over which the raw pitch may not deviate more than a specifiable interval in half-tones from the average pitch, to report the pitch as a new entry at its outlet; these parameters are fine-tuned in the current system to suit the precision of pitch and vibrato detection required from the input audio stream. The object also outputs quantitative measures of loudness, noisiness and brightness, normalized to a value between 0 and 1. Figure 7 shows a running graph of these measures for a section of recorded vocal audio track input, along with their normalized histograms. A video of the same is included in the project media [35]. These measures of timbral qualities are used in our system to determine if a region of audio is voiced or unvoiced, by setting practical thresholds to these measures. The pitch values from the voiced regions of audio were further fed to a custom-written Max external for the next stages of analysis.

Most of the source audio recordings used in development of the system exhibited a just intonation scale, in which pitch levels of different notes in the *raag* or scale form simple integral ratios such as 8/5, 5/3 etc. with one another, owing to such a practice in Indian

Classical Music and to the fact that the recordings were of oral vocal performance. However, equal temperament scale was chosen for implementation in the system, due to its scalability and ease of technical implementation, owing to the property that adjacent notes in this tuning always maintain a constant ratio of $2^{1/12}$. This choice could potentially affect the output of the system – in cases such as that involving a very deep vibrato wherein the difference in mean pitch position between the source note and perceived note could induce errors in pitch tracking. However, this effect can be countered in most practical cases by setting the bound limits around a note (a user-settable variable introduced later in Section 4.3) to sufficiently large values such that both equal temperament and just intonation levels are included deep within the bounds. Pitch lines in Figure 6 portray the differences between the two tunings for some of the notes.

4.3 Note Segmentation

The pitch track is segmented next automatically into various regions belonging to one of the three types: stable note region, transition region, and rest region. While the user is given an option through the interface (Figure 19) to select the notes in the scale or *raag* being used in the performance or interaction, the system was developed to detect and segment the individual notes from the input audio automatically, thus paving way for real-time interaction in a performance scenario. While there are methods developed and systems designed elsewhere to segment input audio or pitch tracks [41][23][5], and those to provide a feedback on the detected notes to the user through real-time interface [24], the current framework required a note segmentation system that is closely associated with the rest of the framework at programming level and that would support real-time feedback, while also recovering a custom-defined set of parameters of expressive elements (as will be described in Section 4.4) simultaneously. Hence, such a system was freshly developed for the current framework, with custom-written C++ code in the Max/MSP platform.

In this system, the process of segmentation of pitch tracks into notes and that of extraction of parameters happen in parallel to each other and in a dependent manner, with the detected boundaries of note segments in time affecting the metrics used to determine the values of the parameters – such as the number of waves of vibrato, and the Portamento Lag (explained in Section 4.4). The system was designed keeping fast/real-time execution in an embedded hardware environment such as a mobile environment in view, in lieu of future implementation plans and the recent trend of many interactive musical applications being developed on such platforms [50][54][57]; it employs a state machine that is quickly and completely updated at the arrival of each pitch value from the input audio – or equivalently – from the pitch track generated from the audio, and that supports immediate output of detected notes and parameters, instead of a buffer-based approach in which the pitch values are accumulated in a buffer with delayed analysis, calculations and output. The methods employed leverage on simple graphical points of interest in the input pitch track such as local maxima and minima, points of zero-crossing and points of inflection, as described further in this section as well as in Section 4.4.

The method of note segmentation employed in the framework, by itself, is not claimed to be a novel contribution in the current work and is developed just as a tool towards supporting the extraction of parameters of expressive elements. However, when combined with the methods employed in extracting the parameters, it forms a novel rule-based and highly user-customizable part of the overall framework that seamlessly integrates the processes of extraction of the parameters and segmentation of the notes, thereby trying to result in a faster, more efficient and more accurate outcome than that of the alternative, typical, buffer-based approach. Further, this segmentation and parameter extraction framework is an integral and essential part of the overall system framework leading to the modeling of the parameters in a novel multi-attribute trie fashion. A detailed quantitative evaluation of the performance of the segmentation and parameter extraction methods is left for future work.

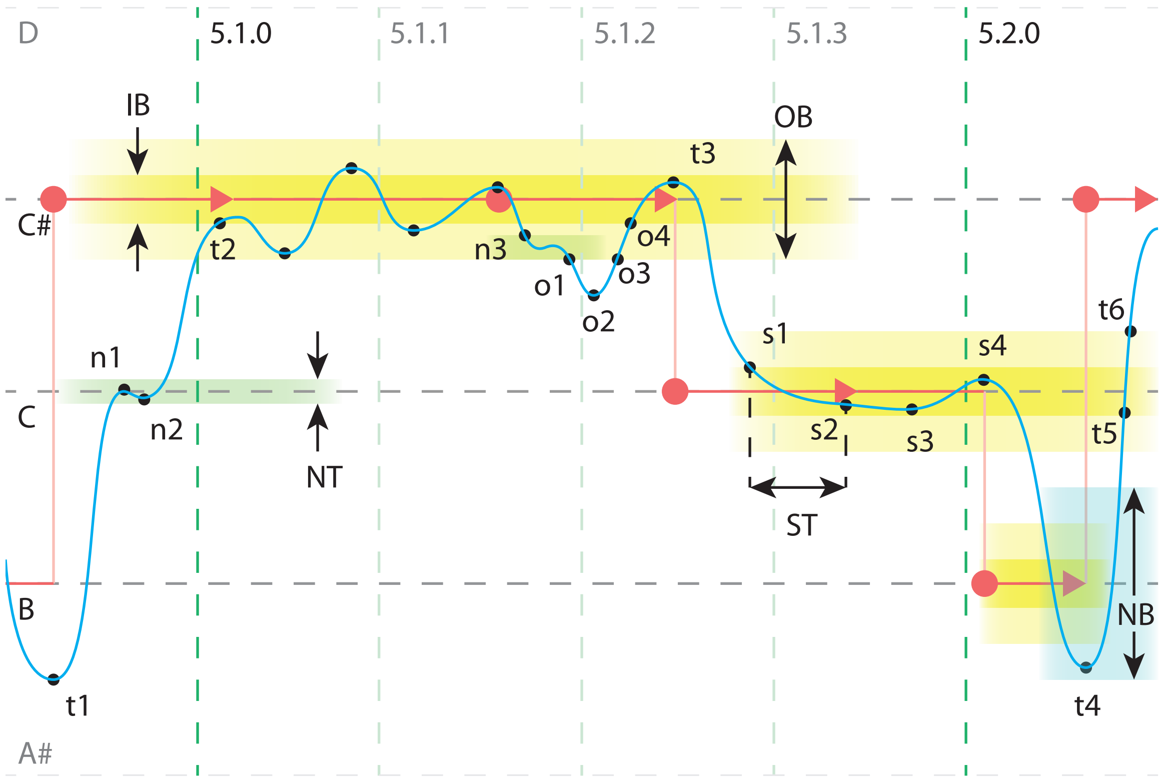


Figure 8: An illustration of a part of a pitch track showing salient user-settable variables used to segment the track into regions of transitions and stable notes; IB = Inner Bound (in pitch); OB = Outer Bound (in pitch); NT = Noise Threshold (in pitch); ST = Stability Threshold (in time). Indexed black dots mark salient points along the pitch track.

Figure 8 shows illustration of a hypothetical pitch track with several of the regions defined above. The horizontal gray dashed lines are the levels of discrete equal temperament notes, the dark gray ones being the notes in the current scale while the light gray ones are the ones outside the scale. The reddish line shows the hypothetical pure notes without vibrato and portamento, while the bluish line is the actual pitch track with the expressive variations. The reddish circular markers indicate the start of pure notes, while the reddish arrowheads show the location on the pure track that corresponds to the time instance when the actual expressive track is deemed by the algorithm to have attained stability for the current note, equivalently the end of portamento leading into the note. The sequence of notes illustrated is B-C#-C#-C-B-C#.

The segment between $t1$ and $t2$, that between $t3$ and $s1$ and that between $s4$ and $t4$ are examples of transition regions – where the pitch gradually transitions from one note level to another. The segment between $t2$ and $t3$ and that between $s1$ and $s4$ are examples of stable note regions – where the pitch level relatively stays close to a note level. The region around $t4$ is an example of a momentary jump to a pitch level.

To determine the pure base notes corresponding to each of these regions, and to simultaneously divide the pitch track into regions as mentioned above, the algorithm uses a state machine. It keeps track of various states of the pitch track, such as the direction in which the pitch track is proceeding in the current instance ($PTState$), whether the algorithm is currently looking to find a crest (local peak – or a point of maximum value in the vicinity) or a trough (local valley – a point of minimum value in the vicinity) in the pitch track ($PTPhase$), and whether the pitch track is already stable or about to be stable around a note or if it is in transition from a note to note ($PTStabilityState$). To determine these states in turn, various measurements, points and variables are defined and tracked along the pitch track.

Algorithm (1), Algorithm (2) and Algorithm (3) show reduced versions of pseudo-codes depicting major state variables as above and their transitions in the pitch tracking process

Algorithm 1 A reduced version of pseudo-code depicting the state variable *PTStabilityState* and its transitions in the pitch tracking process

```
if $PTStabilityState = RESET$ then
    {The following is the ‘usual transient region check’}
    if $Pitch$ is within $InnerBound$ of a note then
        $StableNote \leftarrow$ nearest note in scale
        $PTStabilityState \leftarrow BECOMING$
    else
        $PTStabilityState \leftarrow TRANSIENT$
    end if
else if $PTStabilityState = BECOMING$ then
    if $Pitch$ is within $OuterBound$ of $StableNote$ then
        if it has been so more than a set number of times then
            Stable note detected!
            $ReturnVariables \leftarrow StableNote$, appropriate meter count, & parameters
            $PTStabilityState \leftarrow STABLE$
        end if
    else
        Usual transient region check as above
    end if
else if $PTStabilityState = STABLE$ then
    if $Pitch$ is not within $OuterBound$ of $StableNote$ then
        Going out of stable state
        Usual transient region check as above
    end if
else if $PTStabilityState = TRANSIENT$ then
    Usual transient region check as above
end if
```

– particularly dealing with note segmentation. It should be noted that these do not show many of the intermediate steps and states, and some of the noise threshold considerations. It should also be noted that these do not show comparisons with previous note to avoid re-triggering of the same note – especially in presence of vibrato, and they do not show the determination of meter count values and of parameters of expressive variations. Names of variables and states in these pseudo-codes are self-explanatory in the context of note segmentation and pitch tracking as presented in this chapter. As mentioned before, these algorithms by themselves are not novel in the current work, and are simple heuristics assisting towards extraction of parameters of expressive elements. The following paragraphs portray some of the example cases and how the various states are used to determine regions in those cases.

Algorithm 2 A reduced version of pseudo-code depicting the state variable *PTPhase* and its transitions in the pitch tracking process

```
if $PTPhase = RESET$ then
  if $StartingPitch - Pitch > NoiseThreshold$ then
    $PTPhase \leftarrow FINDING\_TROUGH$
    $TroughPitch \leftarrow StartingPitch$
  else if $Pitch - StartingPitch > NoiseThreshold$ then
    $PTPhase \leftarrow FINDING\_CREST$
    $CrestPitch \leftarrow StartingPitch$
  end if
  $StartingPitch \leftarrow Pitch$
else if $PTPhase = FINDING\_CREST$ then
  if $CrestPitch - Pitch > NoiseThreshold$ then
    Crest and new note detected! (in absence of vibrato)
    $ReturnVariables \leftarrow$ note & meter count nearest to $CrestPitch$, & parameters
    $PTPhase \leftarrow FINDING\_TROUGH$
  end if
else if $PTPhase = FINDING\_TROUGH$ then
  if $Pitch - TroughPitch > NoiseThreshold$ then
    Trough and new note detected! (in absence of vibrato)
    $ReturnVariables \leftarrow$ note & meter count nearest to $TroughPitch$, & parameters
    $PTPhase \leftarrow FINDING\_CREST$
  end if
end if
```

To determine closeness of the pitch level to a note level, two bound regions limited by

Algorithm 3 A reduced version of pseudo-code depicting the state variable $PTState$ and its transitions in the pitch tracking process

```
if $Pitch > PreviousPitch$ then
  if $PTState = GOING\_DOWN$ then
    if $PTPhase \in \{FINDING\_TROUGH, RESET\}$ then
      if $PreviousPitch < TroughPitch$ then
        $TroughPitch \leftarrow PreviousPitch$
      end if
    end if
  end if
  $PTState \leftarrow GOING\_UP$
else if $Pitch < PreviousPitch$ then
  if $PTState = GOING\_UP$ then
    if $PTPhase \in \{FINDING\_CREST, RESET\}$ then
      if $PreviousPitch > CrestPitch$ then
        $CrestPitch \leftarrow PreviousPitch$
      end if
    end if
  end if
  $PTState \leftarrow GOING\_DOWN$
else
  if $PTState = RESET$ then
    $PTState \leftarrow GOING\_FLAT$
  end if
end if
```

user-settable bound values are defined around each note – the inner bound region (colored darker yellow) inside the inner bound (IB), and the outer bound region (colored lighter yellow) inside the outer bound (OB). When the pitch track enters the inner bound around a note (point $t2$ as an example around the note C#), it is deemed to be locked to that note (C#), and is deemed to stay locked until it gets beyond the outer bound (point $o1$). This method of using two thresholds instead of one to determine stability is employed in order to avoid continuous re-triggering of the same stable note in case of vibrato or noise around the bound (as in case of the vibrato seen between points $t2$ and $o1$) – quite similar to the concept of Hysteresis [4] in Physics or Schmitt Trigger [47] in Electronics. Further, when the pitch track goes out of the outer region around the note C# at the point $o1$ and re-enters the inner region at $o4$, the note C# is deemed to have been re-triggered.

In most of the cases, decisions about stability of notes and divisions into regions are taken at the points where the pitch track changes its direction – i.e., at the crests and troughs, as these points are critical points of change on the pitch track and are easier to detect algorithmically. However, a user-settable minimum duration of time called Stability Threshold (ST) is also defined. If the pitch track enters the inner bound around a note and stays inside the outer bound around the same note for this duration ST, the pitch is immediately deemed to be stable around that note, without waiting for a subsequent crest or trough. Accordingly, in Figure 8, the pitch track is deemed to have attained stability at note C at the point $s2$, after entering the region around the note at $s1$ and staying in the same region for the duration ST, even though the trough occurs later at $s3$.

In case of the momentary jump to note B from $s4$ to $t4$, the trough is lying outside the outer bound around note B; the jump exhibits a valid transition – as the span of the jump is beyond the noise threshold (NT) and also also as the pitch goes beyond the outer bound (OB) around the previous note (C). For such cases, a region of nearness (colored blue) is defined around the note within the nearness bound (NB), typically within half a semitone below and above the note. The note nearest to the crest or trough is taken as the base note at such points. Thus, since the trough at *t*4 is closer to note B than to note A#, note B is taken as the base note.

Further, if the pitch track enters the bounds around a note but neither stays within bounds for a period greater than ST nor shows a crest or trough, such as the segment between *t*5 and *t*6, the segment is deemed to be part of a transition region.

For all the cases, a noise threshold (NT) is defined to avoid the effects of noisy low-level variations in the pitch track (such as that between *n*3 and *o*1) that may induce spurious crests and troughs that may lead to wrong decisions by the algorithm. Any change in the direction of pitch within the noise threshold above or below a trough or crest is ignored. Thus, the pitch level difference between a recorded pair of neighboring trough and crest in the pitch track is ensured to be at least NT. In the example, the pair of crest and trough at the points *n*1 and *n*2 would thus be ignored and note C is not deemed to be the base note at the points, thus making the transition from *t*1 to *t*2 a seamless one. Similarly, the pair between points *n*3 and *o*1 would be ignored in making decisions about the vibrato around note C#.

Also for all the cases, the user is given an option to specify the notes of the scale being used in the current performance through the Max interface. This is done to ensure that the system doesn't erroneously detect a note outside the scale, either due to the nature of incoming audio or due to limitations of the pitch estimation process, and to give the user a way to achieve selective modeling of melody. Thus, in the example, if the point *t*4 had been closer to the note A# than to note B, note B would still be deemed to be the base note as A# is colored light gray and thus shown not to be in the scale being used.

Figure 9 is a screenshot of the development interface of the system showing how the system segments a manually simulated pitch track. The input pitch track is smoother and slower in its variations in this case, and is thus clearer visually. Figure 10 is a similar screenshot for a real audio track of the performance of a North Indian Classical Music vocalist. The input pitch track is rougher and faster in its variations in this case. In both the screenshots, it can be seen visually that the system is following the notes, segmenting the

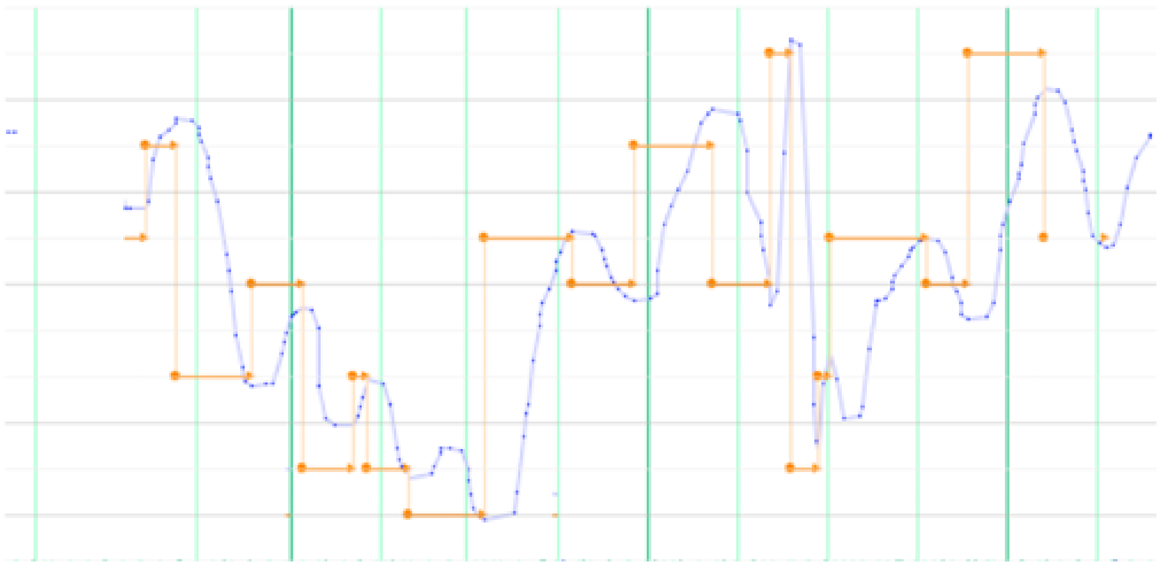


Figure 9: A screenshot of the waveform display in the development interface of the system showing how it segments a manually simulated pitch track; the input pitch track is smoother and slower in its variations in this case; the track also shows portamento lags as determined by the system for each of the notes determined; bluish line = pitch track going from note A to note B; greenish vertical lines = metric grid; reddish lines = pure note without vibrato and portamento; reddish circular markers = start of a new note as determined by the system; reddish arrowheads = end of portamento for the corresponding notes as determined by the system

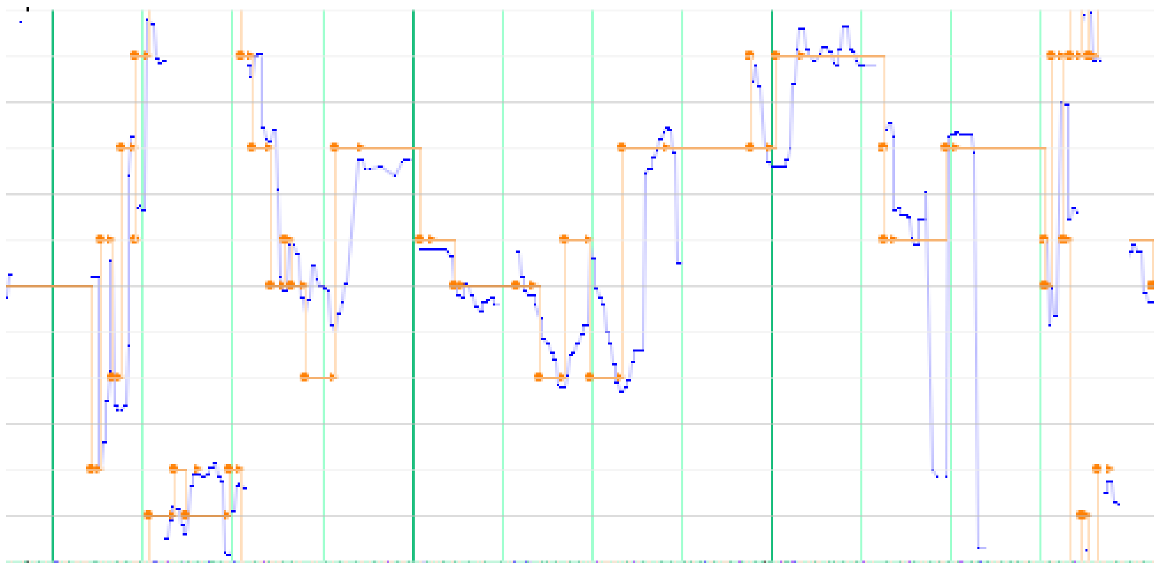


Figure 10: A screenshot of the waveform display in the development interface of the system showing how it segments a pitch track derived from an actual audio input; the input pitch track is rougher and faster in its variations in this case; the track also shows portamento lags as determined by the system for each of the notes determined; bluish line = pitch track going from note A to note B; greenish vertical lines = metric grid; reddish lines = pure note without vibrato and portamento; reddish circular markers = start of a new note as determined by the system; reddish arrowheads = end of portamento for the corresponding notes as determined by the system

pitch track into discrete notes, and marking the end of portamento for each note in the way it was designed to do.

4.4 Feature Extraction

In parallel to the segmentation of the pitch track as detailed in Section 4.3, various measurements are made through the process to extract the parameters for features such as portamento and vibrato. Figure 11 illustrates some of the key parameters involved, and the elements measured for the same. The illustration shows a pitch track (colored blue) traverse the note sequence A-B-A. The track shows a period of portamento at each transition, as well as oscillations of vibrato – particularly about note B.

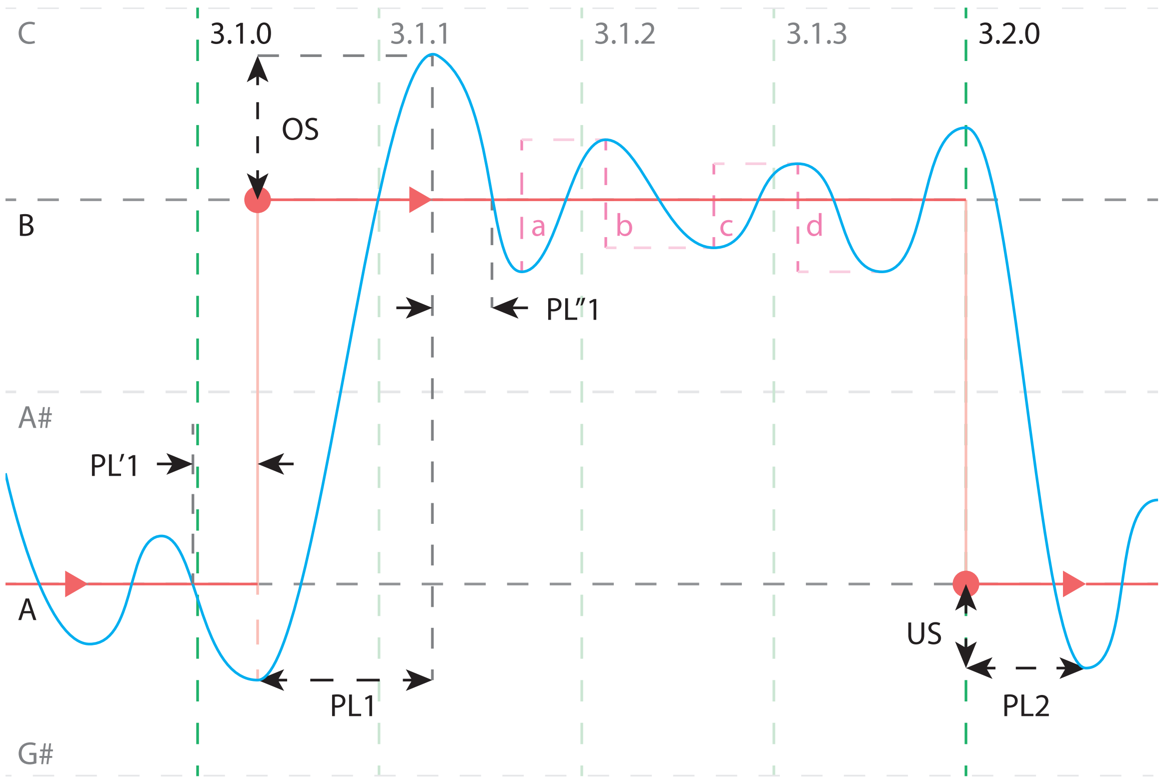


Figure 11: An illustration of a part of a pitch track showing salient measurements used to extract vibrato and portamento parameters; bluish line = pitch track going from note A to note B; greenish vertical lines = metric grid; reddish lines = pure note without vibrato and portamento; PL = Portamento Lag; PL′1 = Overshoot prep lag (for PL1); PL″1 = Overshoot recoil lag (for PL1); OS = Overshoot amount; US = Undershoot amount; pinkish vertical lines a/b/c/d = amplitude values used to determine vibrato depth.

Table 1: Parameters of expressive variations contained in a typical metadata packet

	Parameter	**Description**
1	PortamentoLag	Time between the start of a note and the point where it reaches the target, in terms of a beat duration
2	PortamentoShape	Shape of the portamento curve, as defined by 4 elements: split ratio in time axis (SplitX), split ratio in pitch axis (SplitY), first curve index (Curve1) and second curve index (Curve2)
3	TremoloDepth	Half the amplitude of the tremolo, in terms of a semitone
4	TremoloRate	Frequency of tremolo, in cycles per second
5	VibratoDepth	Half the amplitude of the vibrato, in terms of a semitone
6	VibratoRate	Frequency of vibrato, in cycles per second
7	ShootDepth	Vertical distance of overshoot or undershoot beyond the target note level, in terms of a semitone
8	ShootLag	Time taken for recoil after overshoot or undershoot, in terms of a beat period
9	PrepDepth	Vertical distance of preparatory overshoot or undershoot beyond the source note level, in terms of a semitone
10	PrepLag	Time taken before reaching the extreme point of a preparatory overshoot or undershoot, in terms of a beat period

4.4.1 Coarse Portamento Features

Several coarse parameters are considered for each portamento - dealing with the larger scale span of the portamento in the time and pitch axes: the direction, the lag, overshoot/undershoot amount, and the recoil/shoot lag, as in Table 1.

The direction of portamento refers to whether the transition is upward or if it is downward. This can indirectly be derived from the direction of change between the previous note and the current note; hence, the direction of portamento was chosen not to be included in the metadata packet in the trie model.

The Portamento Lag (PL) refers to the time measured between the last trough of the previous note to the first crest of the target note in case of an upward transition (PL1 for example), or that between the last crest of the previous note to the first trough of the target note in case of a downward transition (PL2 for example). As mentioned in Section 4.3, the local extreme positions of the pitch track are thus taken as salient reference points for measurement in most cases. However, in other special cases such as that when the pitch track starts afresh at a note without a necessary transition, or when a stable note is detected due to Stability Threshold (ST as in Figure 8) rather than an extreme position, corresponding alternate points are used to determine the Portamento Lag. In the implementation, Portamento Lags are represented as fractions of a beat duration, so that they would scale appropriately with changing tempo. Representation in terms of milliseconds of absolute time and that in terms of fraction of the note period were also considered, but were not used since they would not scale well with changing tempo in case of the model being used to generate music.

Further, during the experiments with MATLAB simulation, it was noticed that many of the transitions also had overshoots and undershoots at the beginning and the end. Though an analysis of the cause and the affect of these is outside the scope of this work and there is more detailed discussion elsewhere in literature on this and other such aspects in the context of systems using voice input and melodic transcription on it for sound synthesis and transformation [27], it was speculated that these overshoots and undershoots may be attributed to the mechanical inertia involved in generation of voice as well as to the effort on part of the singer to heighten the effect of the transitions, and thus were deemed to be important aspects of the pitch track. The corresponding parameters were thus recorded as well: the amount of overshoot (OS) or undershoot (US) – corresponding to the difference between the target pitch level and the extreme level to which the pitch track shot beyond the target level, and also the Recoil Lag (PL′′) – corresponding to the time taken for the pitch track to return to the target pitch after the undershoot/overshoot. The region similar to the Recoil Lag just before the Portamento Lag region is termed the Prep Lag (PL′) – corresponding to the time taken for a preparatory deviation from the previous note level in a direction opposite to that of the portamento.

4.4.2 Coarse Vibrato Parameters

Similarly, two parameters were recorded for vibrato: the depth and the rate. To determine the Vibrato Depth, amplitudes of individual oscillations of vibrato (except the first and the last ones corresponding to the shoots and the preps, wherever applicable) were measured separately as the difference between the extreme levels (distances a, b, c and d in the example), and then averaged over the entire note duration. To determine the vibrato rate, a count was taken of the oscillations between the end of Recoil Lag and the end of the note, and divided by the corresponding time period. The end of the note here would be the extreme point (trough or peak) corresponding to the beginning of the portamento towards the next note.

Some of the finer or more detailed parameters of vibrato, dealing with the variation of vibrato depth and that of vibrato rate along the length of a note were left for future work.

4.4.3 Fine Portamento Parameters – Portamento Curves

During the preliminary simulations with MATLAB, it was also seen that some of the rising portamentos had a shape that looked similar to a quick exponential rise, and that most of the falling portamentos had a shape that looked similar to a quick exponential fall, while others had a waveform similar to an sigmoid curve. Accordingly, to determine the sample values of the portamento region during the generation of audio by the system and to be able to generate a reasonably wide array of portamento shapes, two simple sets of wavetables with waveforms as illustrated in Figure 12 were used: one set for the rising waveforms – generated using the polynomial fraction in Eq. (1), and another one for the falling waveforms, generated using Eq. (2). While generating the wavetables, the parameter A was kept at a value of 1.0, while the parameter B was taken from the set $\{$-127/128 -63/64 -31/32 -15/16 -7/8 -3/4 -1/2 0 1 3 7 15 31 63 127$\}$. While a mathematical treatment of this choice of values for the parameter B is out of the scope of this document, it may be observed that the set of curves is symmetric about both of the diagonals in each case in Figure 12.

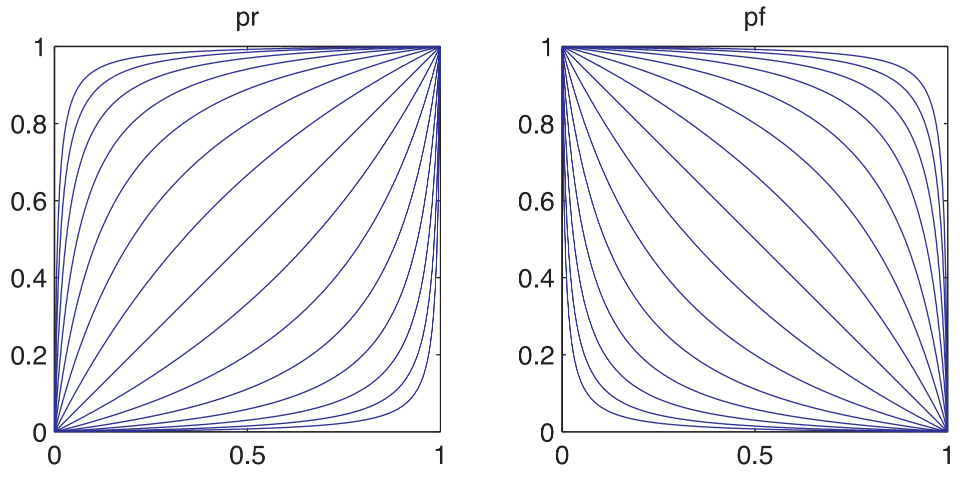


Figure 12: An illustration showing a set of superimposed curves used for modeling portamentos. These are generated using the functions in Eq. (1) and Eq. (2), with parameter *A*=1.0 and with parameter *B* taken from the set {-127/128 -63/64 -31/32 -15/16 -7/8 -3/4 -1/2 0 1 3 7 15 31 63 127}. These elementary curves are combined, scaled and offset appropriately to approximate the portamento segments of the pitch curve.

The above wavetables were scaled and offset in both pitch and time axes and in various forms during audio generation to get a waveform that was continuous and matching with the immediately neighboring points, as illustrated around a portamento region in Figure 16.

$$p_r(x) = A \times \frac{x + xB}{1 + xB} \tag{1}$$

$$p_f(x) = \frac{A(1 - x)}{1 + xB} \tag{2}$$

The above technique, being simple and fast, worked for the purpose through most of the development of the system. However, it may be noted that the resulting synthesized pitch track may not be smooth at all points. This may be seen at the joining point between segments p3 and p4, and at that between segments p1 and p2 in the black dashed line in Figure 15 – which is an example of a synthesized pitch curve. To overcome this limitation, a technique involving a 3rd degree polynomial and associated parameters that can be used to alter the shape of the polynomial, can be considered to achieve a representation that is closer to the original portamento. 3rd degree polynomials are extensively used to generate the spline curves used in a variety of modern graphics softwares and systems, due to their mathematical properties such as being able to be differentiated twice and hence the ability to maintain a smooth shape at the points of inflection or at the end points [3]. A few experiments were done in this direction with spline curves during development, and their incorporation into the framework was left as a future work.

4.4.4 Fine Portamento Parameters – Interpolation and Inflection

To fit the above portamento curves into the region of portamento defined by the Portamento Lag in the pitch track input, the region is divided into two parts separated at the inflection point – if any is present – in the region, and different curves applied to each of the two parts, as illustrated by segments p2 and p3 in Figure 12. For this, the pitch track in the region is interpolated at a number of evenly spaced points inside the region, as shown with the red markers in the left part of Figure 13. The figure shows interpolation with 8 divisions for illustrative purpose, while typically 20 divisions were used in implementation.

The interpolated pitch track is further analyzed to determine the inflection point – which is mathematically the point at which the second derivative or the curvature of a function changes direction – or in simpler non-mathematical terms, where the curve becomes convex to concave. This is a point that can serve as the anchoring point or a joining point for lower-order parts used to model a higher-order curve, and is used with a similar purpose with splines [3] in 2-dimensional computer graphics. Many methods to determine the inflection point from the above interpolated pitch curve were considered during development of the framework. Determination of the inflection point is trivial in most of the simple cases where there is no noise, where the amount of noise involved is low or when the number of divisions is low enough to counter the effect of noise. In such cases, the slope of the interpolated track in each division would be calculated and the difference between the slope values of adjacent divisions (a simple heuristic for second derivative) would be examined to find out the point at which this difference would change the sign, and that point taken as the inflection point. With this method, the point *y* in the right part of Figure 13 would be deemed the inflection point for the example curve in the illustration, which also agrees with visual observation.

While many methods more rigorous and efficient than the above heuristic have been proposed and used [8][1] for cases of noiseless signals and curves similar to the above scenario, more complex cases involving noise require other approaches like Kalman filtering [31][32] or other heuristics. A heuristic was accordingly developed and used in the framework to find the inflection point in a generic noisy case of pitch track, using simple calculations of the area under the curve facilitated by the trapezoidal rule in mathematics [6]. In this method, each point along the divisions is hypothesized to be the inflection point, and the area between the curve and a straight line joining the point to an end point of the curve is calculated, The area values on either side of the point are added with opposite signs, and the absolute value of this sum is defined as the 'area differential' at that point. The point with the highest area differential is further deemed the inflection point. In Figure 13, three such differential areas are illustrated – the area with a pattern of circles, the one with a pattern of lines and the one with a pattern of dots. In the example, point *z* would be deemed the inflection point. While this heuristic may not fetch the optimal inflection point (*y* in the example) in all cases, it yields a solution close to the optimal one in many cases, and has the advantage in presence of noise. Heuristics even better than this and which can fetch the optimal inflection point in presence of noise exist and were considered in the implementation of the framework; they are left to be discussed in future publications.

After the inflection point is determined as described above along with the offset and scaling extent for the curves, one of the curves from the set in Figure 12 is selected to represent the actual portamento curve, using a simple heuristic that selects the curve that is closest to the actual portamento curve. This process is illustrated in Figure 14. The period of the portamento curve *pqrs* is computationally cut into a number of equal divisions, the number being 3 in the illustration as well as in most of the initial experiments with

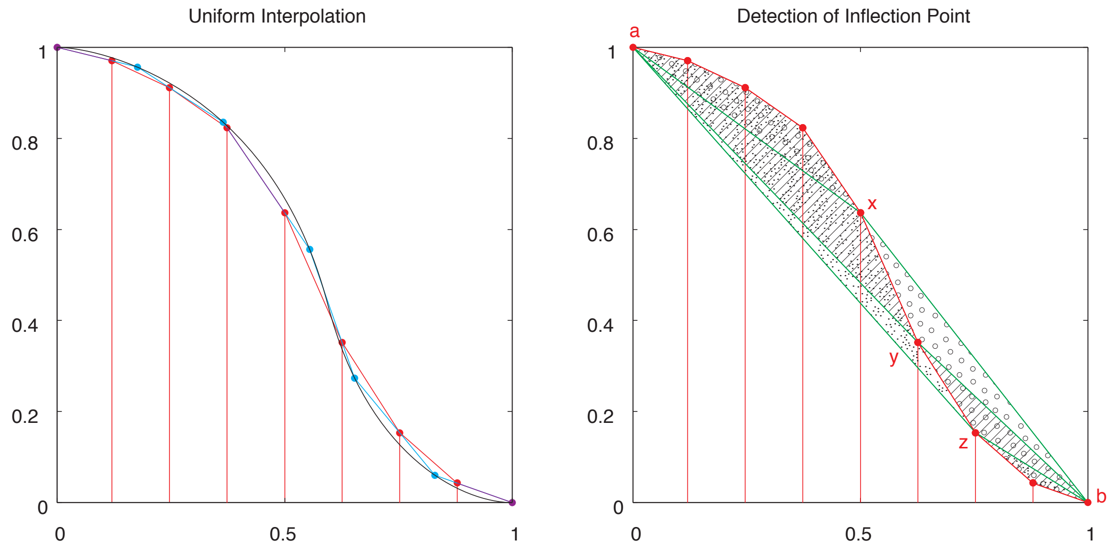


Figure 13: An illustration of uniform interpolation of a normalized portamento region in a sample pitch track (left) and determination of the inflection point from the interpolated pitch track (right); black curve ≡ original smooth pitch locus; blue dots ≡ samples received in the input pitch track; red dots ≡ interpolated pitch points; green lines ≡ demarcations of regions considered in determining the inflection point

the framework. Then, the points of intersection between the candidate as well as original portamento curves and the division boundaries are calculated. The distances between the intersection points along the original portamento curve and those along each candidate portamento curve are further taken, per division, and added up to obtain a heuristic measure of the distance between the particular candidate curve and the original curve. The candidate curve with the least distance measure is finally selected as the closest curve to represent the original curve. In the illustration, three of the closer curves have been shown as dashed black lines, with their points of intersection with the pinkish vertical division lines along the portamento period. The distance of the candidate curve $pabs$ from the original curve $pqrs$ is deemed by the heuristic to be $aq + br$. Similarly, $cq + dr$ and $eq + fr$ are deemed to be the distances of the other two close curves. Since $cq + dr$ is the lowest value among these distances, the candidate curve $pcds$ is finally deemed to be the closest curve to represent $pqrs$.

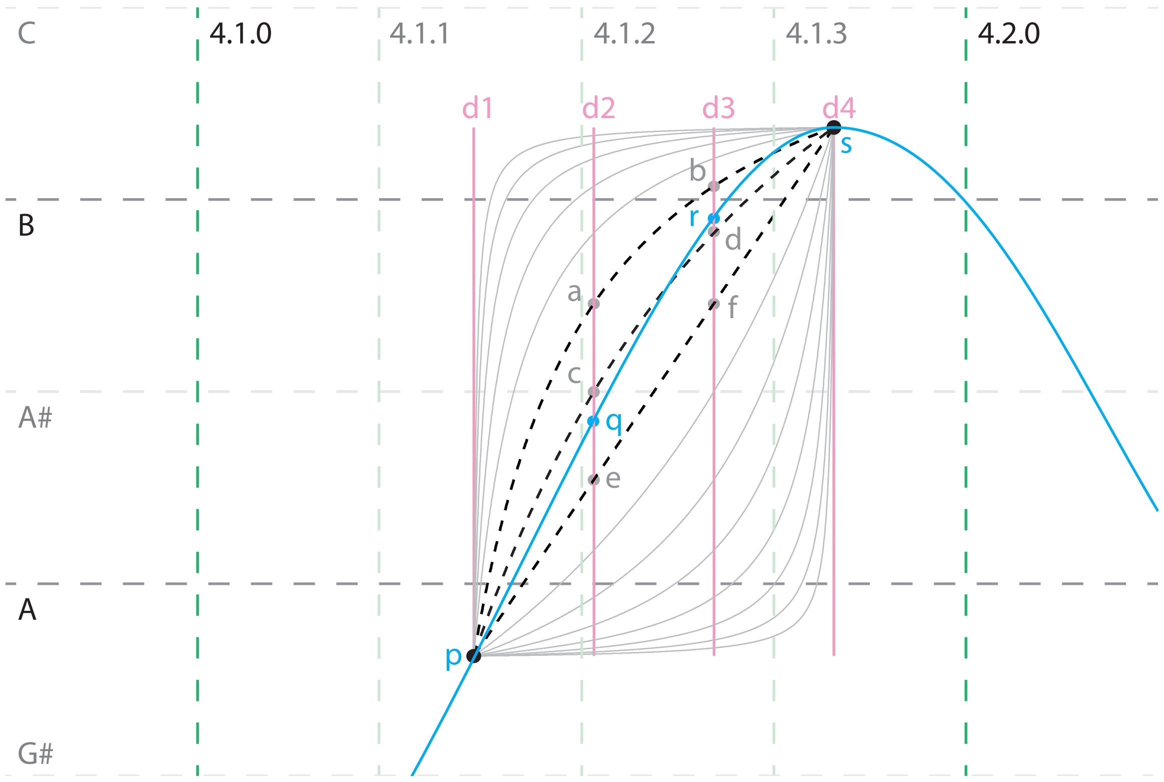


Figure 14: An illustration of selection of the closest parametric curve to represent an actual portamento curve, from a scaled and transposed group of curves; blue solid line ≡ a portamento segment of the original pitch track; gray solid lines ≡ candidate parametric portamento curves from Figure 12; black dashed lines ≡ some of the parametric curves closer to the original pitch track (curve *pcds* being deemed to be the closest one); pinkish vertical lines ≡ borders of sub-divisions of the portamento segment period; dots ≡ points of connection and intersection

4.4.5 Chapter Epilogue

Figure 15 shows the parametric curves corresponding to the parameters determined by the above algorithm for the sample pitch track shown earlier in Figure 11. The black dashed line represents the parametric curve, while the blue continuous line is the original pitch curve. The segments named ‘p’ are the portamento curves, selected from among those in Figure 12 and adjusted through scaling and offsetting. The other segments named ‘s’ are the sinusoidal vibrato curves. The reddish colored block on the pure note track represents the inflection point for the portamento, as also seen in some of the demonstration videos on the project media webpage [35], while the reddish circle and the arrowhead represent the beginning and end of portamento respectively, as mentioned before. Figure 16 shows the same curves as above, overlaid upon a few sets of scaled and offset versions of the parametric curves in Figure 12 (solid and colored gray) over a portamento region and its surrounding segments, to illustrate the selection of a suitable curve for each segment.

After the note segmentation and parameter extraction is done as above, the parameters and discrete elements are embedded in a trie as detailed in the following chapter.

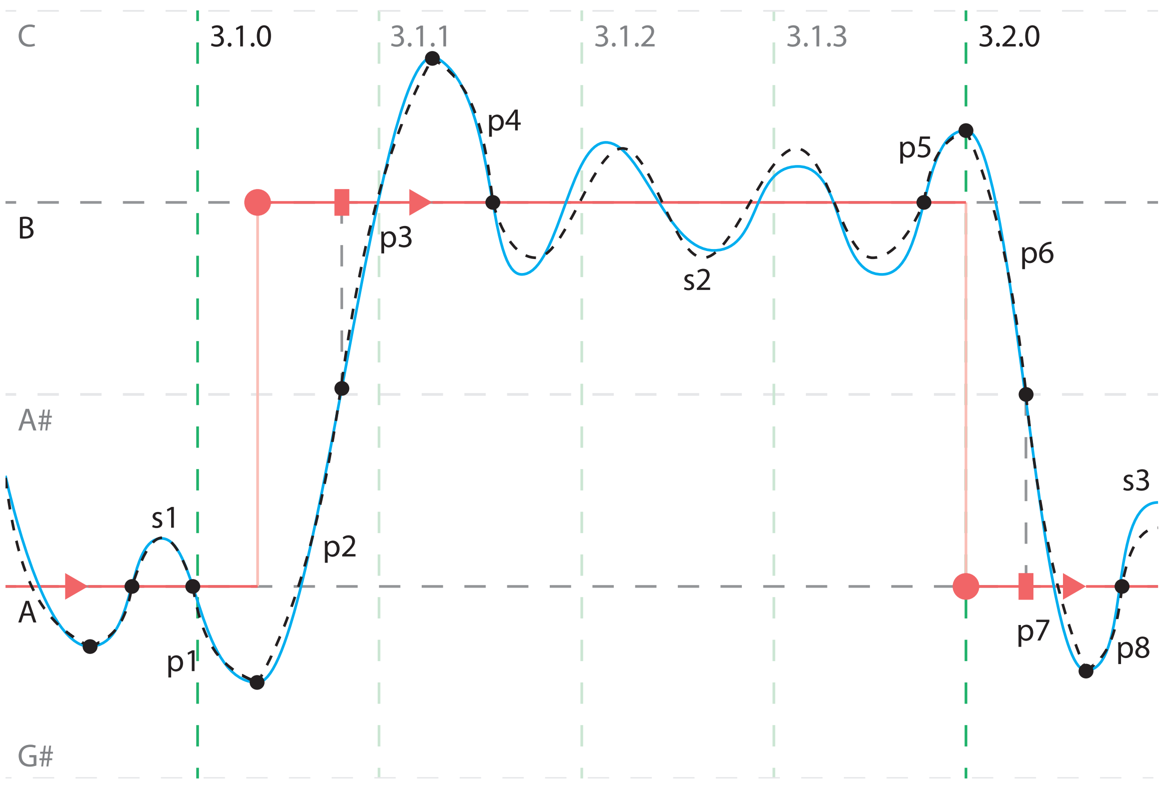


Figure 15: An illustration of parametric curves corresponding to the parameters determined by the system for the sample pitch track in Figure 11; black dashed line ≡ parametric curve; blue solid line ≡ original pitch track; black dots ≡ points of connection

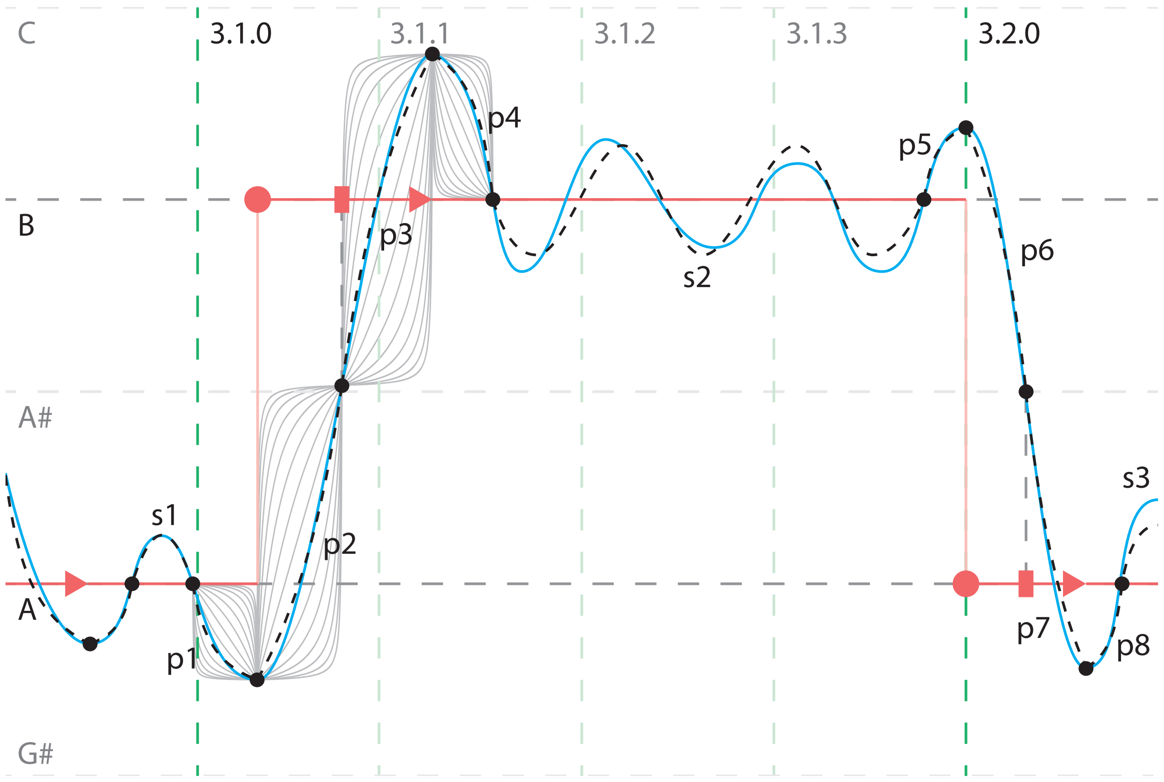


Figure 16: An illustration of scaling, offsetting and selection of curves from those in Figure 12 for a few portamento segments shown in Figure 15; gray solid lines ≡ candidate parametric portamento curves from Figure 12 for each segment; black dashed line ≡ parametric curve (selected ones for portamento); blue solid line ≡ original pitch track; black dots ≡ points of connection

CHAPTER V

MULTI-ATTRIBUTE TRIE MODELING

This chapter presents the techniques used in the work to construct a systematic multi-state model of the discrete notes, durations and continuous parameters derived from the musical input as previously detailed, and the ways in which the model is in turn used to drive machine improvisation. The modeling and evaluation framework, like most of the rest of the system, was implemented in C++ as an external object in Max/MSP along with supporting patches.

5.1 N-gram Modeling

N-gram modeling is a commonly used technique to probabilistically model sequences of elements such as phonemes in speech, letters in a word or musical notes in a phrase [37]. *N*-grams can be efficiently stored in a tree-shaped data structure commonly referred to as a 'trie' or prefix tree. Figure 17 shows the trie for the sequence ABAB+C. In such a trie, branches represent succession of certain symbols after others; and a node at a certain level of the trie holds a symbol from the sequence, along with information about the symbol such as the number of times ('count') it was seen in the sequence following the symbols above it, and the corresponding probability of occurrence. In Figure 17, the subscript below a symbol represents the symbol's probability given the context defined by the path through the trie to that node, while the superscript above it represents the 'count' value. Thus, in the topmost level, the probabilities represent the priors for the symbols.

During construction of the trie, symbols are fed sequentially into the system one-by-one. For the above example, after the sequence ABAB, the trie looks like the trie Trie1 in Figure 17. When a new symbol 'C' follows, corresponding nodes are created at all levels of the trie: 5-gram node using 'ABABC', 4-gram node using 'BABC', trigram node using

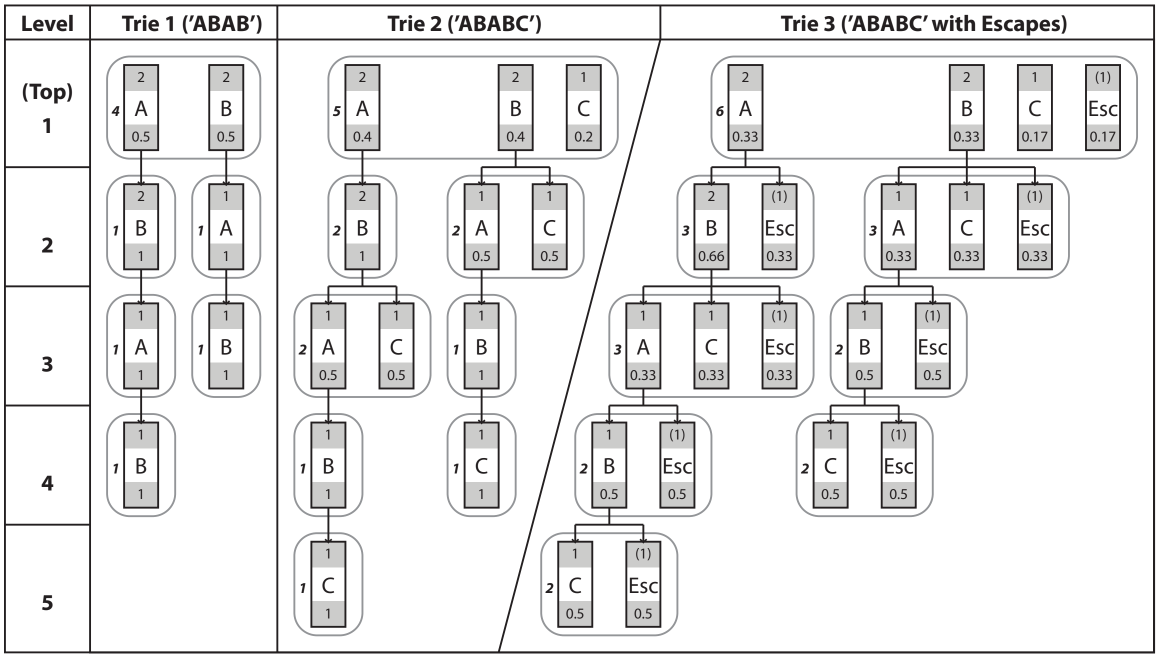


Figure 17: Illustration of tries built for the sequence 'ABAB' followed by the symbol 'C'; superscripts represent 'count' values; subscripts represent 'probability' values; rounded boxes represent sibling set; italicized number at the left of a rounded box represents the total 'count' among the siblings – used in turn to calculate the 'probability' values; the last trie is after adding the 'escape' values

'ABC', bigram node using 'BC' and a 1-gram/prior entry for 'C' at the topmost level.

The corresponding probabilities are also updated resulting in Trie 2 in Figure 17. For this updating of the probabilities, we keep track of the total 'count' (italicized number at the left of each rounded box) for each set of children from a parent node – i.e., for each set of 'siblings' (rounded box in Figure 17). In each update step at a certain level of the trie, the updated probability of a symbol is then found – in its simplest form – by dividing the updated 'count' value by the updated 'total count' value for the symbol's 'siblings' set.

After the trie has been built in this manner it can be used to predict the next symbol given a test sequence. This is done by following the nodes of the trie downwards from its top, in the order of the symbols in the test sequence, until the last symbol in the sequence (and the corresponding node in the trie) is reached. At that point, the probabilities associated with the children nodes represent the predictive distribution over the symbol set, given the observed context.

To allow for new symbols that may appear in the test sequence and to subsequently allow for a better matching of test sequences with missing or extra symbols compared to training sequences, we incorporate the concept of escape probabilities into our trie structure, as described by Witten and Bell [59]. The above example trie would then look like Trie3 in Figure 17. The use of escape probabilities is described in Section 5.2.

For long training sequences, the depth of the trie can become large and is often limited to a maximum order to limit memory usage and to speed up the prediction process given a test sequence.

5.2 Escape Probabilities

Whenever the system encounters a new symbol in the test sequence, the problem of zero frequency occurs; because, in the high-order models, this and most other *n*-grams would never have been observed [13]. Using a simple counting the model would assume zero probability for these unseen events, thereby returning infinite entropy (a measure for the amount of perplexity at a stage - commonly used to evaluate statistical systems), should they occur in the test sequence. The solution is to reserve a small amount of probability mass to events that haven't occurred yet. This is done by reserving an escape probability for each level of the trie. Whenever an event encounters zero probability, the trie uses or returns the escape probability instead.

There are many ways to assign the value of escape probability, as a function of several attributes of the particular node of the trie, such as the total count of instances of the symbols seen so far, the number of unique symbols seen so far, and the number of unique symbols with exactly a certain count of instances. Bell and Witten have evaluated several of these methods [59] as applied to a certain work of written English. For most of the initial experimentation and in the illustrations (Figure 17, Figure 18) in this document, a simple method where the escape probability for each level *n* is assigned by Eq. (3) was used (method referred to as 'A' by Bell and Witten), where N is the total number of tokens/symbols seen by the model so far. Further, Bell and Witten have shown that an approximation of the Poisson distribution method (termed 'X' by them) proves to perform the best among the set of common methods they considered, by being able to result in escape probability values that are closer to the real probability value for unseen events, when applied to n-grams. The same method was further adopted in the current framework as the current work also employs n-grams. In this method, the escape probability for each level n is assigned by Eq. (4), where T_1 is the number of tokens that have occurred exactly once.

$$e(n) = \frac{1}{N(n) + 1}, \tag{3}$$

$$e(n) = \frac{T_1(n)}{N(n)}, \tag{4}$$

5.3 *N-gram Smoothing*

Smoothing addresses the trade-off between the specificity of higher-order models (if a match can be found) and the reliability of the n-gram counts for lower-order models. Since higher order models are much sparser, many n-grams will be assigned zero probability, and counts for n-grams that have been observed will tend to vary greatly based on the particular training database. This variance can be reduced by incorporating information form lower order models.

There are are two basic types of smoothing algorithms: back-off models and interpolation models. Given a test sequence, a back-off model searches for the entire sequence; if no match is found in the trie, the process continues recursively after dropping the first element of the sequence – in effect, searching for a matching node in the lower order models - higher up in the trie structure (envisioned with the root or the prior-based model at the top). The process stops once a positive match is found and the count for that n-gram node is greater than a set threshold. In the example of the trie shown in Figure 17, if a search sequence of

‘AAB’ is input, the prediction system first searches for the entire sequence ‘AAB’ from the root of the trie. Since the sequence is not found in the trie, the subsequence ‘AB’ is tried for next. This sequence is found, and ‘A’ or ‘C’ is seen as subsequent symbol at the third level. One of these symbols is output as the prediction according to their relative probability distribution. Interpolated smoothing by contrast always incorporates lower order information even if the *n*-gram count in question is non-zero – i.e., even if a longer sequence is found in the trie, the prediction routine will continue to to search for the shorter subsequences, and the probability distributions from various levels are subsequently combined to produce the final prediction.

In our earlier study on modeling of tabla sequences [11], we compared two smoothing methods: Kneser-Ney (KN) and an averaging method we termed *1/N*. These were also compared to a simple backoff procedure. KN was adopted because earlier work had shown it to be a superior smoothing method in the context of natural language processing [9]. We showed that Both $1/N$ and Kneser-Ney smoothing significantly outperformed the simple back-off method in the case of tabla sequence modeling, and that $1/N$ is the clear winner for strokes, durations, and joint prediction. In the initial version of the current work, $1/N$ was chosen to be used as the smoothing method, while leaving a systematic study of the smoothing methods to future work.

5.4 Types

Types are the fundamental elements of information stored as discrete variables in the nodes of the trie.

Three kinds of types are commonly recognized in literature [14]: the basic types, the derived types and the cross/product types. Basic types are those that are directly extracted from a musical piece and are independent of one another, such as notes and their durations. Cross types are formed when two or more basic types are combined and tracked simultaneously. A cross type formed using notes and durations would consist of all symbols for notes in combination with all symbols used for quantized durations. Each element of this crossed type is represented as a tuple, {Note, Duration}, instead of a single value. The number of all possible elements in a cross type is equal to the product of the number of elements in each basic type. A derived type depends on information extracted from a basic type. A simple example of this is melodic intervals, which are extracted from pitches. Derived types can further lead to the formation of cross types. Selection of appropriate representations is domain-dependent and often uses prior knowledge of the music.

In their work on multiple-viewpoint systems for music prediction in the context of Bach's chorale melodies [14], Conklin and Witten used 6 basic types: Pitch, Key Signature, Time Signature, Fermata, Start Time and Duration. These in turn gave rise to around 22 derived types used in their work. In the earlier work with tabla modeling done with colleagues at GTCMT [11] using the same framework as employed in the current work, two basic types were used: tabla strokes and durations. These were in turn used in two crossed types: Stroke $\otimes$ Duration, and Stroke $\otimes$ Postition-In-Bar (PIB), where PIB referred to the onset position of a stroke as a fraction of the bar. The tabla strokes also gave rise to three derived types. It was further shown that the cross types lead to small improvements in the model outcome in the context of solo tabla compositions.

Keeping the above choices for types as references, in the initial implementation of the current system, two basic types are used: (1) Note and (2) PositionInBar/PIB as a pair of Beat and Tatum value. PositionInBar in turn acts to define the duration of the notes indirectly in case of continuous voice input. These basic types are in turn combined into the cross type: Notes $\otimes$ PIB. This cross type is the fundamental symbol stored in the nodes of most of the tries built in the system. Over the course of several experiments with different types including the straight representations of Inter-Onset-Intervals (IOIs) and durations, this type proved to be suitable for representing both the melodic as well as the rhythmic information of the audio input in relation to the metric grid.

Two derived types are also introduced into the model: constructed by mapping the

MIDI note number into a reduced representation: (1) an integer specifying if the note falls in the same, lower or higher octave compared to that of the previous note (TriOctave), and (2) Pitch Class. These two are in turn combined into a cross type: TriOctave $\otimes$ PitchClass. Thus, the MIDI note set was reduced to a set of 3 octaves $\times$ 12 pitch classes = 36 symbols. This type would provide a good compression of the melodic space so that the resulting trie would be less sparse and more maturely built, and was thus found to be useful in cases where the melody would span several octaves of pitch more than the human voice - such as when the audio input is coming from a musical instrument like a keyboard synthesizer. However, in case of human voice input where the source spans less number of octaves, the compression offered by this derived type was not found to be of much advantage in front of the crossed type of Notes $\otimes$ PIB where exact note information could be preserved. Further experiments with other basic, derived and cross types is left for future improvements.

5.5 Storing parameters of expressive variations

While the structure described above is used to hold the details about the discrete notes as well as the durations between them or their positions in the time grid, the parameters for the expressive variations determined in Section 4.4 are not stored in the same manner. Instead, the parameters corresponding to each node in the above trie is stored as a packet of metadata along with the node. This serves to position the parameters as an optional entity in the trie, available for use upon the discretion of the user and the generation logic, without disturbing the control flow in the tree structure.

Each parameter in the metadata packet, in turn, is modeled as a zeroth-order Markov chain, i.e., as a collection of alternate values with an associated prior probability distribution. Thus, when the node is encountered as a part of the search through the trie when the system is generating audio output, one of these values is chosen according to their relative probability distribution, and used to generate the note with expressive variations. Figure 18 shows an illustration of one such metadata packet.

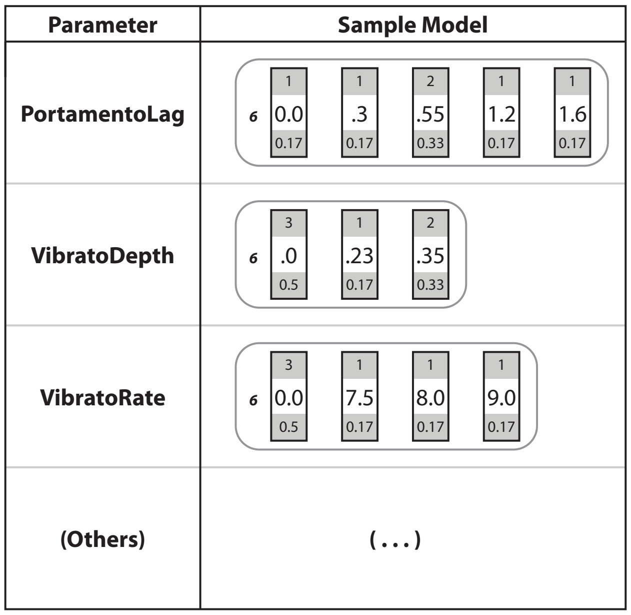


Figure 18: Illustration of a metadata packet showing zeroth-order Markov models of PortamentoLag, VibratoDepth and VibratoRate; superscripts represent ‘count’ values; subscripts represent ‘probability’ values; a rounded box represents a sibling set; italicized number at the left of a rounded box represents the total ‘count’ among the siblings - used in turn to calculate the ‘probability’ values

However, unlike in the case of discrete notes or discrete PIB values, the parameter values for expressive variations are continuous - which would mean that if all of their values are accepted, they would very soon consume a disproportionate amount of memory, leading to performance and resource bottlenecks. Moreover, the technique of using a zeroth-order Markov chain modeling the relative probability distribution among the parameter values would seldom work in this case as it would be very rare to get the exact same value of the parameter in its continuous domain. To counter this issue, each of the parameters was quantized with a reasonable resolution. For example, for Portamento Lag, 1/10th of a tatum (a tatum being the smallest division of a beat set as the resolution for note events for the current model) is set as the standard resolution. For Vibrato Depth, 1 cent of pitch is set as the standard resolution. For Vibrato rate, 0.25 Hz is set as the standard resolution in current implementation, while a logarithmic scale might be adapted in the future for a better representation.

Various other ways of storing the parameters of expressive variations were considered before the above method of using metadata packets with zeroth-order Markov chains was adopted in the platform. The trivial method of using a common value or a common range of values for each of the parameters is hypothesized to result in less expressive and musical output than the original human performance, and improving upon this trivial method is in fact one of the goals of the framework. Another method considered was to include the parameters as parts of the fundamental symbol vector used for each node in the trie. This would however make the symbol space disproportionally large and the trie branches would get sparse – rendering the trie unable to model closer relationships between the discrete elements that would otherwise get modeled in an obvious sense. On the other hand, the method of modeling the parameters for expressions completely separate from the discrete elements was considered, where a separate trie or multiple separate tries would be built for the parameters. This method can enable modeling of temporal relationships between values of parameters. However, it fails to enable modeling of dependencies between the parameters and the discrete elements, while such relationships were reported to be existing by several related works of research as mentioned in Chapter 2. Thus, the method of storing the parameters in metadata packets in the nodes of a trie with discrete elements was adopted with the expectation that it would strike a middle ground in the above considerations, being able to model relationships between parameters of continuous elements and discrete elements without making the trie get disproportionately sparse.

It is a further observational fact that the context of discrete elements in a certain node of the trie doesn't always exhibit the same values of parameters of continuous elements in the input stream; for example, the portamento may be well-exhibited for a certain note in an instance of a phrase, while there may only be a short hint of it in another repetition of the phrase, and the portamento may be completely absent in yet another repetition of the phrase, thus giving rise to three different values of Portamento Lag. The zeroth-order Markov chain was adopted to enable more efficient modeling of such scenarios by allowing multiple values of a parameter to be stored along with a corresponding probability of occurrence in the context of a trie node, instead of storing just one value that would have been either the most recent one or the one that is relatively more frequent.

CHAPTER VI

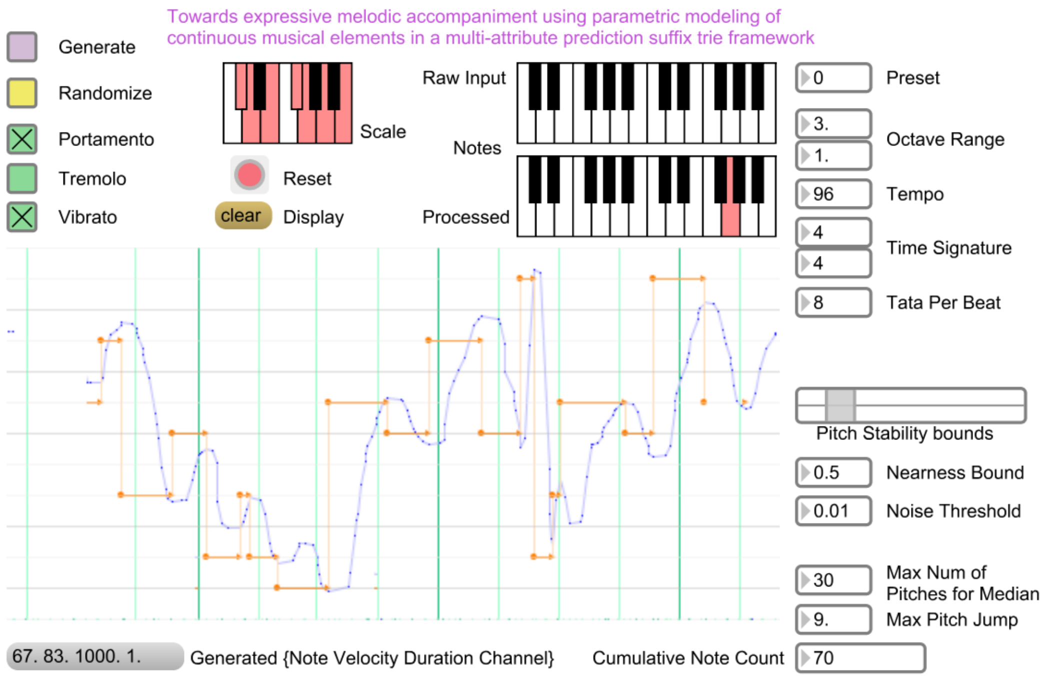
SOUND SYNTHESIS, INTERACTION AND INTERFACE

In the generation phase, the system traverses through the multi-attribute trie-based computational model built as described in the previous chapter, to predict the next note to be played and its corresponding attributes and placement. This is done as a part of the process to continue the sequence of notes registered till the current point of time – either as derived from the input or as generated by the system to be the output, depending on the interaction set-up. Once the predictions are derived from the model and the output is determined, in the form of a MIDI note number along with the pitch bend values, these are fed into a software sound synthesizer to render the voice in real-time. For the purposes of development and evaluation, a simple software waveform generator was used that generated anti-aliased sound output using simple waveforms such as sine, triangle, square and sawtooth on Max/MSP. The same MIDI note numbers along with the pitch bend values can potentially be fed into a hardware or software synthesizer for a more realistic instrument voice. For a performance scenario, the software instruments in a performance software such as Ableton Live would be a good example. Alternatively, these output values can potentially be used to drive a synthesis engine that programmatically generates waveforms using a mathematical model, such as one using the Karplus-Strong physical model [33][26], in which case the synthesis engine can be more tightly coupled to the current analysis and multi-attribute-trie-based modeling system, and can potentially render the expressive variations in a more realistic way.

In the preliminary implementation of the system, the system can continuously receive the incoming audio, generate the pitch track from it, analyze the pitch track and train the computational model with parameters for every new note detected, and when indicated by the user, can continue the sequence using the model developed so far and generate stylistically similar musical output using the voice chosen in the synthesizer. The system can potentially be used in other forms of interaction such as call-and-response and accompaniment with the human performer, and adopting the system for a performance involving such interactions is left for future work.

To enable the user to make such interactions with the system as well as to modify the various parameters involved in learning and generation phases of the system, the user is given an interface such as the one in Figure 19. Using the interface, the user can specify if the system should generate music or if it should just learn from his input, and can enable or disable various expressive elements such as vibrato and portamento. The user can also specify the notes used in the melodic scale through an on-screen musical keyboard, as well as the meter of the performance through the time signature, tempo and time resolution. Further, the user can set the various bounds and thresholds used in the pitch tracking process, and see the tracking and generation process in action through real-time updates of the input or output pitch track in multiple colors, similar to the illustrations used in Figure 8 and Figure 11. The horizontal lines in the track display in Figure 19 represent the distinct MIDI note levels - the ones colored darker gray corresponding to the black keys and the ones colored lighter gray corresponding to the white keys on a standard musical keyboard (unlike in the illustrations in Figure 8 and Figure 11, where the darker gray color represented the notes belonging to the scale selected).

The Max/MSP objects demonstrating the functionality of the modeling framework may be downloaded from the thesis web page [35].


Towards expressive melodic accompaniment using parametric modeling of continuous musical elements in a multi-attribute prediction suffix trie framework
Generate
Randomize
Portamento
Tremolo
Vibrato
Scale
Reset
clear
Display
Raw Input
Notes
Processed
Pitch Stability bounds
Generated {Note Velocity Duration Channel}
Cumulative Note Count

0	Preset
3. / 1.	Octave Range
96	Tempo
4 / 4	Time Signature
8	Tata Per Beat
0.5	Nearness Bound
0.01	Noise Threshold
30	Max Num of Pitches for Median
9.	Max Pitch Jump
67. 83. 1000. 1.	Generated {Note Velocity Duration Channel}
70	Cumulative Note Count

Figure 19: A snapshot of the preliminary interface with controls to adjust various parameters

CHAPTER VII

SYSTEM EVALUATION

An anonymous survey was conducted over the internet [36] to determine the aggregate subjective opinion of a pool of uncontrolled adult participants involving both musically trained and untrained subjects, about several audio tracks generated by the system. The survey aimed at determining if the subjects perceived the output of the system to be musical and expressive in general, and if they perceived the version of the output with modeled expressive elements to be more musical and expressive compared to that without the expressive elements, as well as to that with randomly generated expressive elements, thereby establishing the success of the system and the method of modeling employed.

7.1 Design of the survey

Four different monophonic audio recordings of vocal performances by masters in North Indian Classical Music were chosen to be the source materials – involving two pieces each from two vocalists. The pieces were all in different *raag*s: *gaud malhAr*, *tODi/shubhapantuvarAli*, *asAvEri/natabhairavi* and *dEsh*, selected such that they span a wide array of pitches, have fairly different note selections and varied tempi - although, since the emphasis in the work was on expressive modulations, the pieces selected exhibited a working tempo limited to a range between 90 and 120 BPM. From each piece, four different audio tracks were generated for the survey: the first one as a re-rendering of the pitch track from the original recording, the second one generated by the system using the model but without expressive elements, the third one generated by the system using the model but with randomly assigned parameters to the expressive elements with a standard sigmoid curve for each portamento, and the fourth one generated by the system using the model with true

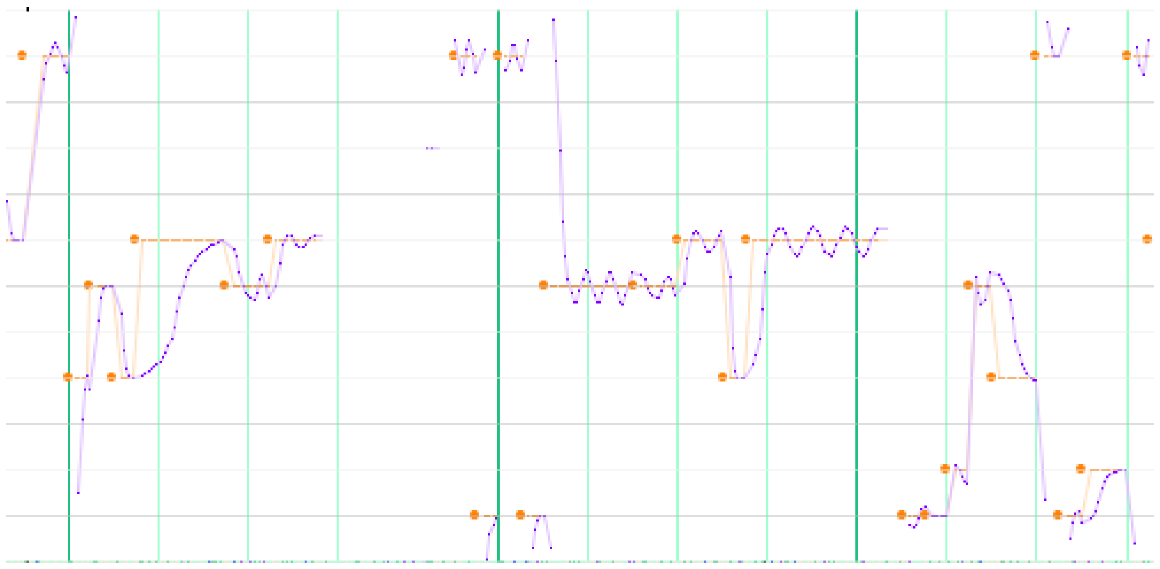


Figure 20: A screenshot of the waveform display in the development interface of the system showing an example generative pitch track generated using pseudo-random values for the parameters of vibrato and portamento, with a standard sigmoid curve for all the portamento regions; bluish line ≡ generated pitch track; greenish vertical lines ≡ metric grid; reddish lines ≡ pure note of the generated pitch track without vibrato and portamento; reddish circular markers ≡ start of a new note

expressive elements as modeled parametrically, including parametrically modeled portamento. Figure 20 shows an example generation pitch track for the third case, with the characteristic sigmoid curves. Thus, the subjects were presented with four sets of tracks containing four tracks each. In this chapter and elsewhere in the document, the 4 original recordings of the performances in different *raag*s are referred to as ’sources’ or ’pieces’, while the 4 versions derived out of each of them are referred to as ’tracks’.

To make sure that the subjects are not biased due to differing timbres, all the tracks – including the ones representing the original pitch track without using the model – were rendered using the same waveform in the synthesizer. The generated tracks however did not necessarily have the same line of melody as that in the original pitch track or even between one another, as the model in the system was allowed to run free for each of the generated tracks, after one common round of training. Also, to counter any biasing effect of the temporal order in which the tracks were presented, the order of the sets of tracks as well as that of the tracks in each set was scrambled independently for each participant in the survey.

The subjects were asked to rate the tracks for their musicality and expressiveness on a 7-point scale. Demographic information about the subjects such as their age, degree of musical experience and that of exposure to Indian Classical Music was also collected - to aid checking for any correlations that might exist between the responses and the demographics of the subjects, and to aid in future reviews of the survey results.

7.2 Transcription of the survey response plots

The figures derived from the survey responses and included in this chapter show box plots of the responses, along with the statistical multiple-comparison of the responses following a one-sided analysis of variance (ANOVA) with a 95% confidence interval. These are standard techniques followed in statistics [22] to analyze aggregate and pairwise trends among grouped data such as the responses received in the survey conducted in this work. The responses are presented here in their unweighted form – considering responses of all participants equally, as well as in a weighted form. The weighting was done with two schemes: one according to the rating the participants gave for their own musical abilities in the demographic section of the survey, and the other one according to the ratings they gave to their exposure to and skills in Indian Classical Music. Since the number of participants in the survey who explicitly specified their level of expertise in ICM was relatively low compared to the total number of participants, the results of weighting by that scheme are not presented here, and is planned to be considered for future analysis.

There are three types of figures presented here:

- The figures with 4 box plots such as Figure 28 separately show the trends for each of the 4 source audio recordings that were used to generate the audio tracks used for the survey, with each box plot showing the response statistics for each of the 4 track versions

- The figures with 2 box plots and 2 multi-comparison plots combined together such as Figure 21 show the aggregate trends for each version of the tracks - obtained by combining the responses received for all source recordings together
- The figures with 2 multi-comparison plots such as Figure 24 show the relative trends among all of the 16 audio tracks used in the survey

The graphical elements numbered 1 to 4 in the sub-plots – for example, those in the box-plot at the bottom-left of Figure 28 – correspond to the 4 versions of audio tracks generated from each source recording as previously mentioned:

1. A direct rendering of the pitch track generated from the original source recording without using the model
2. Output of the system with modeled discrete elements but without using any expressive elements
3. Output of the system with modeled discrete elements and using randomly generated parameters for the expressive elements – along with a sigmoid curve for portamentos
4. Full-fledged output of the system with modeled discrete elements and modeled expressive elements

It should be noted that in all of the multi-comparison plots, the indices of track numbers (1 to 4 or 1 to 16) are inverted with respect to the order mentioned above – or in other words, are placed from top to bottom in the above order. Thus, the elements numbered 4 in the multi-comp plots of Figure 21 are the versions with direct rendering of the pitch track.

In case of the graphs showing multiple comparison of all 16 of the audio tracks at once – such as in Figure 24, sets of 4 consecutive tracks refer to those derived from a single source, with an individual order as above. Thus, for example, tracks numbered 1 to 4 in the plot are from the 4th source, those numbered 9 to 12 are from the 2nd source, while the track numbered 15 is the non-expressive version derived from the 1st source.

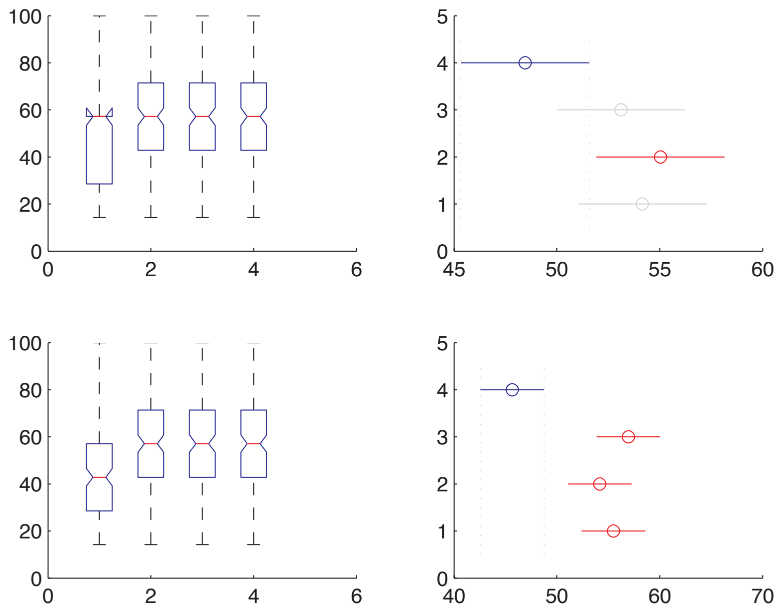


Figure 21: Box plots (left) and multiple comparison plots (right) of aggregate unweighted survey responses for expressiveness (top) and musicality (bottom) of the tracks

The musicality and expressiveness axes are represented in a scale between 0 and 100 – 0 being the minimum and 100 being the maximum value possible for each quality, also considering the weighting factor, if applied.

7.3 Results of the survey

Around 39 responses were obtained in total, from a population with a mean age of around 30 years, with around 1/6th of the participants being females. About half of the participants mentioned themselves to be musicians, and about half of the total number of participants were familiar with Indian Classical Music to various degrees.

In most of the cases, the participants rated the track with a rendering of the original pitch track to be lower in musicality and expressiveness compared to the other three synthesized tracks. This can be seen in Figure 21 showing the aggregate statistics of unweighted responses, as well as in Figure 22 showing the same aggregate statistics with the responses weighted by the musicianship of the participants. This result, considering the aggregate

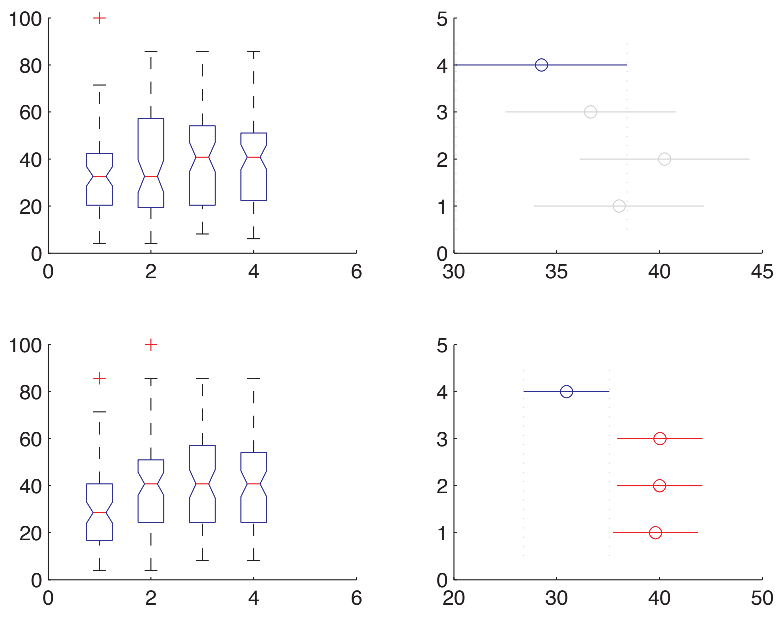


Figure 22: Box plots (left) and multiple comparison plots (right) of aggregate survey responses for expressiveness (top) and musicality (bottom) of the tracks weighted by the musicianship of the participants

responses, can be seen to be significant at the 95% confidence interval for all synthesized cases with respect to musicality, while being not equally significant in some synthesized cases for their expressiveness. Nevertheless, this proves that the tracks generated by the system – with or without expressive parameters – were perceived to be more musical and equally expressive – if not more expressive, than the tracks that had a reproduction of the original pitch track. This is possibly aided by the clearer and more stable track generated by the system – compared to the original pitch track which is prone to pitch tracking errors and uneven fluctuations of source voice. This result can also be considered a testament to the performance of the part of the system modeling the discrete elements of notes and their durations and positions – being able to produce musical results. Another factor that might have helped render the original pitch tracks to be less musical and expressive is the fact that many of the notes in the original source recordings were sung in just intonation scale, while

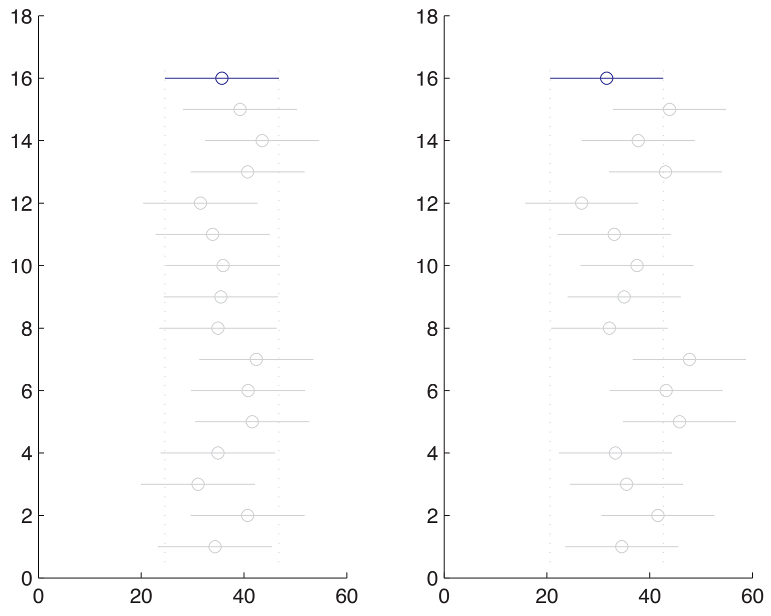


Figure 23: Multiple comparison plot of survey responses for expressiveness (left) and musicality (right) of the tracks weighted by the musicianship of the participants

the system produced output in equal temperament scale – owing to technical reasons as explained earlier in Section 4.3; this might have rendered sections of the original pitch track relatively out-of-tune in comparison to the generated tracks, especially to participants who are more exposed to the equal temperament tuning in popular music. This result however is limited to the pitch track generated from the source recording, and does not imply anything about the stand-off between the generated tracks and the original source recording itself, as the source recordings were not included in the survey. Further experiments are required to evaluate any such relations.

The multi-comparison plots comparing the 16 tracks in both the weighted (Figure 23) and unweighted (Figure 24) cases show that in case of 3 out of the 4 source recordings (the 3rd piece in *asAvEri/natabhairavi raag* being the exception), the participants on an average rated the tracks using expressive parameters (either random or modeled) to be more expressive than the one generated with plain notes without vibrato and portamento. This shows that the generation of expressive parameters by the system is indeed causing a

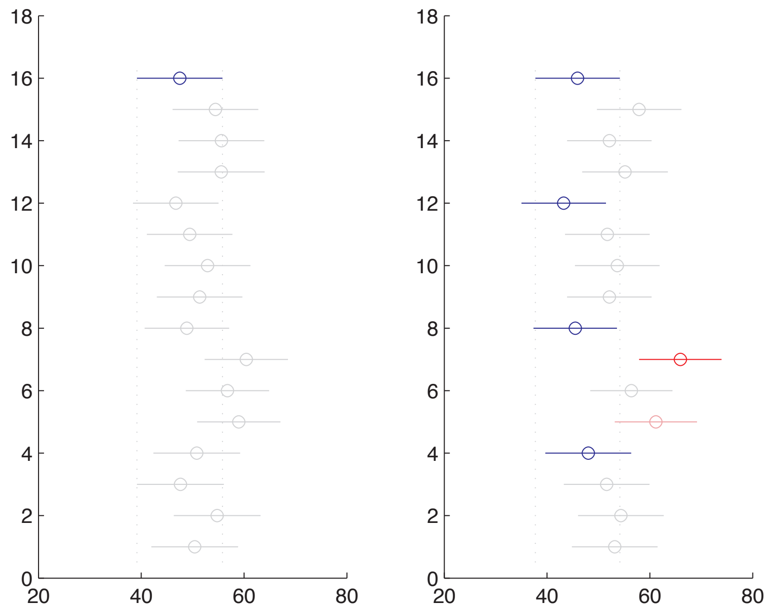


Figure 24: Multiple comparison plot of unweighted survey responses for expressiveness (left) and musicality (right) of the tracks

perceivable difference and improvement in the expressivity of the tracks. However, these differences are not statistically significant with the 95% confidence interval, and thus call for more experimentation to draw strong conclusions about the relative stand-off between the versions with and without expressive elements. The participants' individual views about expressivity and about excessive use of expressivity may be attributed to have played a role in this case, as well as the negative predisposition induced by a few impulsive glitches in the generated expressive tracks due to octave errors in pitch detection along with some occasional discontinuities found in the tracks. The result nevertheless serves as a positive basis and inspiration for future improvements.

Regarding the stand-off between the two versions of expressive tracks generated, the graphs with both weighted (Figure 23) and unweighted (Figure 24) responses show that in case of two of the sources, the version with modeled parameters is perceived to be better

Table 2: Numerical mean values of unweighted survey responses

	Piece 1	Piece 2	Piece 3	Piece 4
		Expressiveness		
Original	47.43	46.71	48.86	50.86
No Expressions	54.43	49.43	60.43	47.57
Random Expressions	55.57	52.86	56.71	54.71
Modeled Expressions	55.57	51.29	59.00	50.43
		Musicality		
Original	46.00	43.29	45.43	48.00
No Expressions	57.86	51.71	66.00	51.57
Random Expressions	52.14	53.71	56.43	54.43
Modeled Expressions	55.14	52.14	61.14	53.14

Table 3: Numerical mean values of survey responses weighted by the musicality of the participants

	Piece 1	Piece 2	Piece 3	Piece 4
		Expressiveness		
Original	17.43	19.14	21.14	20.29
No Expressions	15.29	16.43	17.43	17.29
Random Expressions	15.57	19.57	18.86	19.14
Modeled Expressions	17.43	15.57	20.29	17.14
		Musicality		
Original	15.43	21.29	18.43	21.57
No Expressions	13.00	16.14	18.29	17.00
Random Expressions	14.43	22.00	20.00	21.14
Modeled Expressions	16.71	17.71	20.86	17.29

musically compared to the version with random parameters and sigmoid curve. However, the result is reversed in case of the other two sources. Hence, and also owing to the fact that this result is again not shown to be statistically significant, it may be concluded that more experimentation is needed to establish the stand-off between the versions with modeled and random parameters. One aspect to consider here is the fact that in case of the track generated using random values for parameters, while the vibrato rate, depth and portamento lag were randomly assigned values from within a predefined range, the curve for portamento was kept constant as a sigmoid curve. Many portamento regions in the source tracks were observed to indeed have a sigmoid shape, thus resulting in the version with random parameters being closer to that with modeled parameters. This may be considered a possible reason as to why the survey results did not show a strong stand-off between the two versions. A better test would be one in which all of the parameters, including the portamento shape, are randomly generated. It may also be noted that the ranges set for random values assigned for the above parameters were chosen to be optimal ranges within which most of the actual values of parameters as observed in the original source recordings lied, determined by the author through manual hearing of the source tracks. For example, the range for random vibrato depth was set to be between 0.01 and 0.45 times a semitone, that for vibrato rate was set to be between 2 and 10 Hz. This also can be expected to have rendered the two versions closer together. These considerations will be made during future evaluations.

The box plots of the various distributions of survey responses are given in Figure 25, Figure 26, Figure 27 and Figure 28, along with numerical mean values in Table 2 and Table 3 for further reference and comparison.

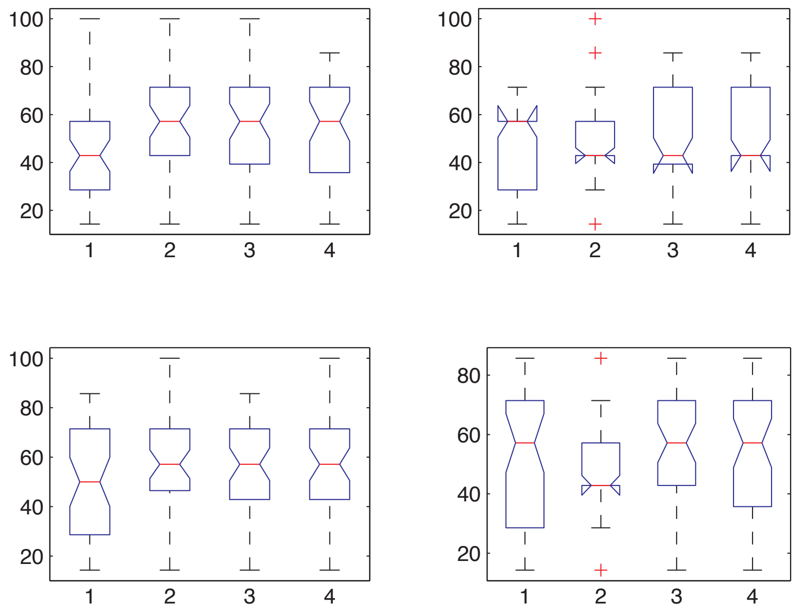


Figure 25: Box plot of unweighted survey responses for expressiveness of the tracks

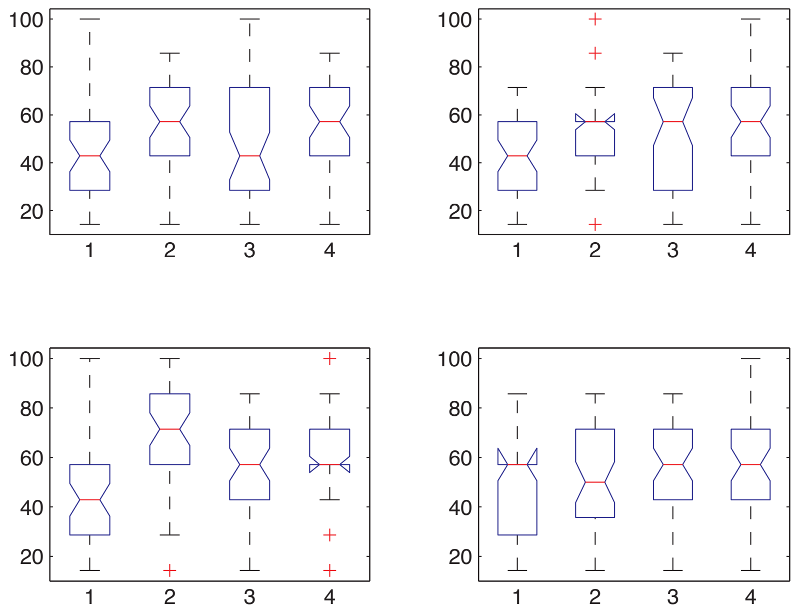


Figure 26: Box plot of unweighted survey responses for musicality of the tracks

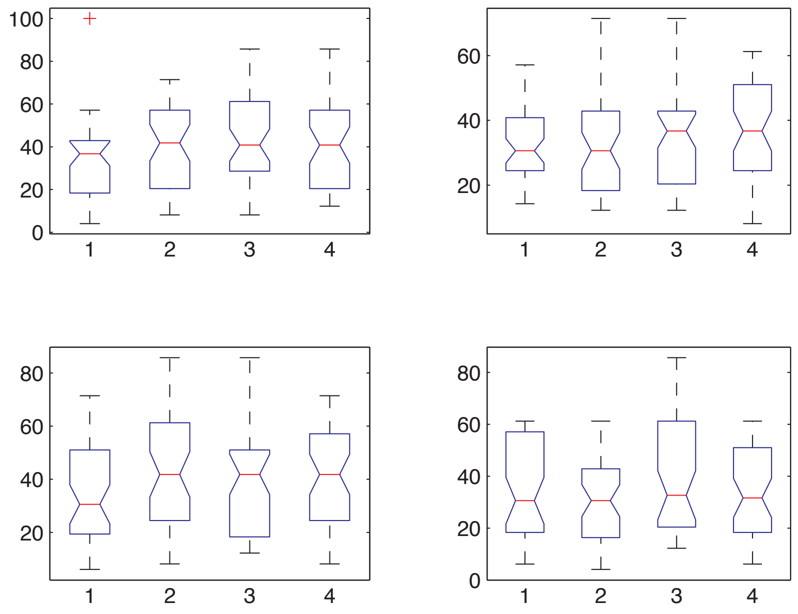


Figure 27: Box plot of survey responses for expressiveness of the tracks weighted by the musicianship of the participants

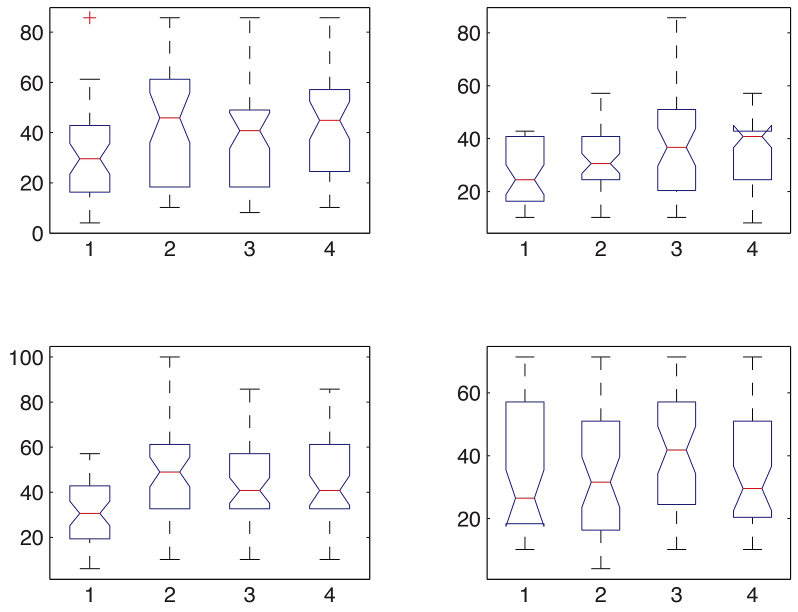


Figure 28: Box plot of survey responses for musicality of the tracks weighted by the musicianship of the participants

CHAPTER VIII

FUTURE WORK

While there are many ways in which the system can be improved to model musical input in a better way and to be better adopted for live performance, the following are the key improvements being considered.

8.1 An improved pitch tracking

While the pitch estimator in the analyzer~ object [29] currently used works well enough to derive a pitch track from certain input types, several algorithms have been proposed over the recent years that may be better in specific ways. Among them, SWIPE′ [7] seems to be a good algorithm for musical input – being inspired by sawtooth waveform, it has resulted in better tracking of pitch in our preliminary simulation on MATLAB, and it provides the convenience of being able to specify limits on the pitch to be estimated – which may be of great help in the current system as such a limit is already being set on the notes in the training stage. Elsewhere, wavelet-based pitch estimators have been used to detect pitches in speech [28][21] – showing by nature a good resolution in time, along with that in frequency – which may be an important factor in the current system. An approach combining the spectral performance of SWIPE′ with the time resolution of wavelets might work best for the system.

8.2 Implementation of various non-intrusive interaction modes

The user currently needs to explicitly specify if the system needs to generate notes or if it has to just listen and learn, and he also needs to adjust various parameters and musical performance settings manually. The system also doesn't have the interaction modes necessary for a live performance – such as call-and-response and coherent accompaniment [58].

Implementation of these modes with a reasonable amount of automation is thus desirable.

8.3 Adoption of Hidden Markov Models

The current implementation of the modeling only has a modified trie structure with varying orders – essentially a Markov model, but does not contain any hidden layers as in case of a Hidden Markov Model (HMM). HMMs have been widely used in computational modeling of music [25][10] and language [51] elsewhere, and especially are applicable where the observable output needs to be modeled as a result of an intermediate abstraction. In case of the current system, such a HMM structure may prove to be useful for segmentation of the pitch track, as well as for representation of the interaction among various elements of melodic and rhythmic data.

8.4 Adoption of spline curves

In the current implementation, sample values while generating the portamento section of a note are obtained by scaling, interpolating and offsetting a predetermined and saved wavetable of simple and appropriately shaped polynomial fraction curves. The current method of applying these curves may however introduce undesirable discontinuities or roughness in the output pitch waveform, as no smoothness consideration is being made at the joining points between these curves. Instead of this static approach, a dynamic and full-fledged spline-based approach may be taken for better smoothness and precision in the output waveform. Splines are parametric piece-wise polynomial curves extensively used in computer graphics – especially to draw smooth 2-dimensional curves [3]. A couple of preliminary experiments have been done with the system in this regard.

8.5 Working with a larger database

The current system has been developed, tried and tested with a small subset of only 25 audio track recordings of vocal performances of North Indian Classical Music, involving only 2 artists and 14 *raag*s, and the formal evaluation has been done with a still smaller sub-set. Trial with a larger database of audio tracks, in different styles, scales, instruments and by different artists is desirable to make the system more robust and generic.

8.6 Building up a long-term model

The current implementation always functions within the context of a single piece of audio input, requiring to re-learn the parametric model every time a different input is given – essentially working as a Short Term Model (STM). A Long Term Model (LTM) may be desired in such a case, involving building of a library of models to be stored in the memory and retrieved to supplement the STM. This has been shown earlier to be especially helpful at the beginnings of a performance or audio input [11] – where the STM has not adequately developed yet.

CHAPTER IX

CONCLUSION

The thesis thus has presented a computational system that was developed to parametrically model the expressive continuous variations such as vibrato and portamento from real-time musical audio input, in a multi-attribute multi-order trie-based framework. A rule-based and highly user-customizable sub-system that seamlessly integrated the processes of segmentation of notes and extraction of their parameters using a real-time state-machine based approach was developed and presented. While the musical notes and their positions were modeled as a modification of the traditional suffix trie, the parameters of the expressive variations were stored as metadata associated with each node of the trie. Each parameter stored in the metadata was further modeled as a zeroth-order Markov chain – equivalently with a prior probability distribution. This scheme of storing the parameters of expressive musical elements inside the trie framework being the novel aspect, the model was further used to generate musical output stylistically similar to the audio input. Several vocal recordings of North Indian Classical Music were used for development as well as to evaluate the system. A survey was conducted with both musically trained and untrained participants over the internet to obtain their subjective opinions about the musicality and expressiveness of audio tracks generated by the system. The results of the survey show the output tracks generated from the system to be as musical and expressive, if not more, than the case where the pitch track generated from the original audio was directly rendered as output. They also show the output with expressive elements to be perceivably more expressive than the version of the output without expressive parameters. The results further suggest that more experimentation may be required to conclude the efficacy of the framework employed in relation to using randomly selected parameter values for the expressive elements. The document further presented some of the important improvements that may be considered for the system and the framework in the future.

REFERENCES

[1] Abhyankar, S. S. and Bajaj, C. J., "Automatic parametrization of rational curves and surfaces II: cubics and cubicoids," *Computer Aided Design*, vol. 19, no. 9, pp. 499–502, 1987. 4.4.4

[2] Ames, C., "The Markov process as a compositional model: A survey and tutorial," *Leonardo*, vol. 22, no. 2, pp. 175–187, 1989. 1

[3] Bartels, R. H., Beatty, J. C., and Barsky, B. A., *An introduction to splines for use in computer graphics and geometric modeling*. Los Altos, California, USA: M. Kaufmann Publishers, 1987. 4.4.3, 4.4.4, 8.4

[4] Bertotti, G. and Mayergoyz, I., *The Science of Hysteresis*. Reading, Massachusetts: Academic Press, Dec. 2005. 4.3

[5] Brossier, P., Bello, J. P., and Plumbley, M. D., "Real-time temporal segmentation of note objects in music signals," in *Proceedings of the International Computer Music Conference (ICMC)*, (Miami, Florida, USA), International Computer Music Association (ICMA), Nov. 2004. 4.3

[6] Burden, R. L. and Faires, D. J., *Numerical Analysis*. Brooks Cole, 7 ed., Dec. 2000. 4.4.4

[7] Camacho, A., *SWIPE: A Sawtooth Waveform Inspired Pitch Estimator for Speech and Music*. PhD dissertation, University of Florida, Department of Computer and Information Science and Engineering, 2007. 4.1, 8.1

[8] Chen, F. and Wang, W., "Computing real inflection points of cubic algebraic curves," *Comput. Aided Geom. Des.*, vol. 20, no. 2, pp. 101–117, 2003. 4.4.4

[9] Chen, S. and Goodman, J., "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting of the Association for Computational Linguistics (ACL)*, pp. 310–318, 1996. 5.3

[10] Chordia, P., *Automatic Transcription of Solo Tabla Music*. PhD thesis, Stanford University, Dec. 2005. 8.3

[11] Chordia, P., Albin, A., Sastry, A., and Mallikarjuna, T., "Multiple viewpoints modeling of tabla sequences," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, (Utrecht, Netherlands), International Society for Music Information Retrieval (ISMIR), Aug. 2010. 2.2, 5.3, 5.4, 8.6

[12] Chordia, P. and Rae, A., "Understanding emotion in Raag: An empirical survey of listener responses," in *Proceedings of the International Computer Music Conference (ICMC)*, 2007. 1

[13] Cleary, J. G. and Teahan, W. J., "Experiments on the zero frequency problem," in *Proceedings of the Data Compression Conference (DCC)*, p. 480, 1995. 5.2

[14] Conklin, D. and Witten, I. H., "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, pp. 51–73, 1995. 2.2, 5.4

[15] Davis, S. B. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980. 4.1

[16] Desain, P. and Honing, H., "Towards algorithmic descriptions for continuous modulations of musical parameters," in *Proceedings of the International Computer Music Conference (ICMC)*, (San Francisco, USA), pp. 393–395, International Computer Music Association (ICMA), 1995. 2

[17] Desain, P. and Honing, H., "Modeling continuous aspects of music performance: Vibrato and portamento." Keynote address at International Conference of Music Perception and Cognition (ICMPC), Aug. 1996. 2.1

[18] Desain, P., Honing, H., Aarts, R., and Timmers, R., "Rhythmic aspects of vibrato," in *Rhythm Perception and Production* (Desain, P. and Windsor, W. L., eds.), pp. 203–216, Lisse: Swets & Zeitlinger, 2000. 2.1

[19] Deva, B. C., *The music of India: a scientific study*. New Delhi: Munshiram Manoharlal, 1981. 1

[20] Dubnov, S., Assayag, G., and Cont, A., "Audio oracle: A new algorithm for fast learning of audio structures," in *Proceedings of International Computer Music Conference (ICMC)*, (Copenhagen), pp. 224–228, International Computer Music Association (ICMA), 2007. 2.2

[21] ErÁelebi, E., "Second generation wavelet transform-based pitch period estimation and voiced/unvoiced decision for speech signals," *Applied Acoustics*, vol. 64, no. 1, pp. 25 – 41, 2003. 8.1

[22] Fisher, R. A., Bennett, J. H., and Yates, F., *Statistical methods, experimental design and scientific inference*. Oxford University Press, Oxford, England, 1990. edited by J.H. Bennett, foreword by F. Yates. 7.2

[23] Foote, J., "Automatic audio segmentation using a measure of audio novelty," in *Proceedings of IEEE Inernational Conference on Multimedia and Expo (ICME)*, vol. 1, pp. 452–455, 2000. 4.3

[24] Freeman, J. and Colella, A., "Tools for real-time notation," *Contemporary Music Review*, vol. 29, no. 1, 2010. 4.3

[25] Gillet, O. and Richard, G., "Supervised and unsupervised sequence modeling for drum transcription," in *Proceedings of International Conference on Music Information Retrieval*, 2007. 8.3

[26] Jaffe, D. A. and Smith, J. O., "Extensions of the Karplus-Strong plucked-string algorithm," *Computer Music Journal (CMJ)*, vol. 7, pp. 56–69, Summer 1983. 6

[27] Janer, J., *Singing-driven interfaces for sound synthesizers*. PhD dissertation, Universitat Pompeu Fabra, Music Technology Group, Department of Information and Communication Technologies, Barcelona, Spain, 2008. 4.4.1

[28] Janer, L., Bonet, J. J., and Lleida-Solano, E., "Pitch detection and voiced/unvoiced decision algorithm based on wavelet transforms," in *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP)*, pp. 1209–1212, 1996. 8.1

[29] Jehan, T., *Creating Music by Listening*. PhD dissertation, Massachusetts Institute of Technology, Program in Media Arts and Sciences, School of Architecture and Planning, Sept. 2005. Appendix A. 4.2, 8.1

[30] Johnson, A. M., Zhang, Y., and Jiang, J., "Nonlinear dynamic analysis and modeling of vibrato in the singing voice," in *4th International Conference on the Physiology and Acoustics of Singing*, 2009. 1

[31] Kalman, R. E., "A new approach to linear filtering and prediction problems," *Transactions of the American Society of Mechanical Engineers (ASME). Series D, Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 4.4.4

[32] Kalman, R. E. and Bucy, R. S., "New results in linear filtering and prediction theory," *Transactions of the American Society of Mechanical Engineers (ASME). Series D, Journal of Basic Engineering*, vol. 83, pp. 95–107, 1961. 4.4.4

[33] Karplus, K. and Strong, A., "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal (CMJ)*, vol. 7, pp. 43–55, Summer 1983. 6

[34] Krishnaswamy, A., "Application of pitch tracking to South Indian classical music," in *Proceedings of the Inernational Conference on Multimedia and Expo (ICME)*, vol. 3, (Washington, DC, USA), pp. 389–392, IEEE Computer Society, 2003. 1

[35] Mallikarjuna, T., "Demonstration media for thesis towards Master of Science in Music Technology." URL: `http://www.trishulmallikarjuna.com/msmt_thesis.php`, Aug. 2010. Georgia Tech Center for Music Technology (GTCMT), Georgia Institute of Technology, Atlanta, USA. 3, 4.1, 4.2, 4.4.5, 6

[36] Mallikarjuna, T., "A survey evaluating audio tracks." URL: `http://gtcmt.gatech.edu/mm/survey01`, Aug. 2010. Georgia Tech Center for Music Technology (GTCMT), Georgia Institute of Technology, Atlanta, USA. 7

[37] Manning, C. and Schute, H., *Foundations of Statistical Natural Language Processing*. MIT Press, 2002. 5.1

[38] Mermelstein, P., "Distance measures for speech recognition – psychological and instrumental," in *Joint Workshop on Pattern Recognition and Artificial Intelligence*, 1976. 4.1

[39] Müller, M., *Information Retrieval for Music and Motion*, section 3.4, pp. 65–67. Springer, 2007. 4.1

[40] Pachet, F., "The continuator: Musical interaction with style," in *Proceedings of International Computer Music Conference (ICMC)* (ICMA, I., ed.), (Gteborg, Sweden), pp. 211–218, International Computer Music Association (ICMA), Sept. 2002. 1, 2.2

[41] Paiva, R. P., Mendes, T., and Cardoso, A., "From pitches to notes: Creation and segmentation of pitch tracks for melody detection in polyphonic audio," *Journal of New Music Research*, vol. 37, pp. 185–205, Sept. 2008. 4.3

[42] Pearce, M. T., *The construction and evaluation of statistical models of melodic structure in music perception and cognition*. PhD thesis, City University, London, 2005. 2.2

[43] Perfecto Herrera, J. B., "Vibrato extraction and parameterization in the spectral modeling synthesis framework," in *Proceedings of Digital Audio Effects Workshop (DAFX) (DAFX98)*, 1998. 1, 2.1

[44] Puckette, M. S., Apel, T., and Zicarelli, D. D., "Real-time audio analysis tools for Pd and MSP," in *Proceedings of the International Computer Music Conference (ICMC)* (Oz, W. V. and Yannakakis, M., eds.), 1998. 4.2

[45] Rowe, R., *Interactive Music Systems: Machine Listening and Composing*. Cambridge, Massachusetts: The MIT Press, 16 Jan. 1992. 1, 2.2

[46] Rowe, R., *Machine Musicianship*. Cambridge, Massachusetts: The MIT Press, 1 Jan. 2004. 1, 2.2

[47] Schmitt, O. H., "A thermionic trigger," *Journal of Scientific Instruments*, vol. 15, no. 1, p. 24, 1938. 4.3

[48] Schoonderwaldt, E. and Friberg, A., "Toward a rule-based model for violin vibrato," in *Proceedings of the Workshop on Current Research Directions in Computer Music*, (Barcelona, Spain), pp. 61–64, Pompeu Fabra University, Audiovisual Institute, Nov. 2001. 1, 2.1, 1

[49] Swift, G. N., *South Indian 'Gamaka' and the Violin*, vol. 21 of *Asian Music*, pp. 71–89. University of Texas Press, Spring – Summer 1990. 1

[50] Tedeschi, B., "Stoking a music fan's fancy with apps that rock," *The New York Times*, p. B8, 20th May 2010. New York Edition. 4.3

[51] Thede, S. M. and Harper, M. P., "A second-order Hidden Markov Model for part-of-speech tagging," in *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 175–182, 1999. 8.3

[52] Timmers, R. and Desain, P., "Vibrato: questions and answers from musicians and science," in *Proceedings of the 6th International Conference of Music Perception and Cognition (ICMPC)*, (Keele, UK), Aug. 2000. 2.1

[53] Trivino-Rodriguez, J. L. and Morales-Bueno, R., "Using Multiattribute Prediction Suffix Graphs to predict and generate music," *Computer Music Journal (CMJ)*, vol. 25, pp. 62–79, Autumn 2001. 2.2

[54] Wang, G., "Designing Smule's Ocarina: The iPhone's Magic Flute," in *Proceedings of New Interfaces for Musical Expression (NIME)*, (Pittsburgh, PA, USA), pp. 303–307, June 2009. 4.3

[55] Weinberg, G., "Robotic musicianship jam session - Eastern." URL: `http://www.youtube.com/watch?v=5DYOqSTmGDA`, Nov. 2008. 2.2

[56] Weinberg, G., Blosser, B., Mallikarjuna, T., and Raman, A., "The creation of a multi-human, multi-robot interactive jam session," in *Proceedings of International Conference on New Interfaces for Musical Expression (NIME)*, (Pittsburgh, Pennsylvania, USA), June 2009. 2.2

[57] Weinberg, G., Godfrey, M., and Beck, A., "ZOOZbeat: mobile music recreation," in *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Extended Abstracts*, (Atlanta, Georgia, USA), pp. 4817–4822, Apr. 2010. 4.3

[58] Weinberg, G., Hoffman, G., Nikolaidis, R., and Mallikarjuna, T., "A survey of recent interactive compositions for Shimon – the perceptual and improvisational robotic marimba player," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Taipei, Taiwan), Oct. 2010. IROS Workshop on Robots and Musical Expressions (IRWME). 2.2, 8.2

[59] Witten, I. H. and Bell, T. C., "The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression," *IEEE Transactions on Information Theory*, vol. 37, no. 4, 1991. 5.1, 5.2

VITA

Trishul Mallikarjuna hails from the state of Karnataka in India. He graduated with the degree of Bachelor of Engineering in Electronics and Communication Engineering from Sri Jayachamarajendra College of Engineering (SJCE), Mysore, affiliated to Visvesvaraya Technological University (VTU), Belgaum. He worked as an audio firmware engineer at Broadcom Corporation, specializing in bringing up integrated decoders for commercial audio formats on real-time embedded digital signal processors, before pursuing his Master of Science degree in Music Technology at the Georgia Institute of Technology, Atlanta, USA. At Georgia Tech, his interests and work has involved programming the marimba-playing robot Shimon to improvise musically, analyzing 3D motion capture data to investigate the effect of visual cues on music, musical information retrieval from visual as well as audio sources, and methods of computational modeling of musical performance. He has played electronic drums and has electronically sequenced music for several music groups in India. Apart from music, he shares interest in digital visual arts and media, photography, poetry and writing, social dancing, computer technology, swimming, adventure sports and traveling. More details are in his personal website `http://www.trishulmallikarjuna.com`.