F0 Modeling For Singing Voice Synthesizers with LSTM Recurrent Neural Networks

Serkan Özer

MASTER THESIS UPF/2015

Master in Sound and Music Computing

Master Thesis Supervisors:

Merlijn Blauw and Marti Umbert

Department of Information and Communication Technologies

Universitat Pompeu Fabra,Barcelona

UNIVERSITAT POMPEU FABRA

Abstract

In singing voice synthesis process, score and lyrics for a target song are converted to singing voice expression parameters such as F0, spectra and dynamics. However, this study aims to model and automatically generate F0 parameter by assuring expressiveness and human-likeness in final synthesized singing voice. Musical contexts are important factor on evolution of F0 through a singing performance. Thus, we propose a machine-learning framework that learns F0 of the singing from a set of real human singing recordings with respect to musical contexts, at the same time, capturing expressiveness and naturalness of the human singer. Then, we can automatically generate F0 parameter from our trained model given musical contexts of the score. Recurrent Neural Networks with Long Short Term Memory networks are employed for first time to this specific problem due to their flexibility and strong power in modeling complex sequences. Two recurrent neural networks are trained to learn baseline and vibrato parts of F0 separately. Then, F0 sequences are generated from the trained networks and applied to a singing voice synthesizer. Finally, synthesized songs are evaluated with AB preference tests.

Acknowledgments

First of all, I would like to thank my supervisors Marti Umbert and Merlijn Blaauw for their guidance, support and patience throughout this year. I would also like to thank Xavier Serra for giving me the opportunity of being part of the Sound and Music Computing Master. I would also thank all the teachers for providing their precious knowledge on this exciting field. Moreover, I would like to thank all my classmates for the time that we spent together sharing, discussing and partying. Finally, I express my gratitude to all who involved in the evaluation survey of this thesis

Content

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Synthesizing human singing has been a topic of interest among researchers for many years. However, while simulation of acoustic instruments have been matured and become popular among musicians, singing voice synthesizers, just in the last decade, reached to a convincing point where synthesized singing voice can be used as yet another musical instrument. In this thesis, we will focus on one main component of these synthesizers, F0 or sung melodic contour. We propose a system that automatically generates F0 contour from a machine-learning framework when a musical score is given. It will be a statistical black box that creates a F0 contour containing necessary characteristics such as naturalness and expressiveness.

In the following sections, we will elaborate on the research problem and our goals. In chapter 2, we will provide state of the art research related to generation of F0 in the context of singing synthesizers. In chapter 3, we provide our methodology, and in chapter 4, evaluation strategy and the results will be provided. Finally, in chapter 5 we will provide the conclusions and results.

1.1 Research Problem

In the singing voice synthesis process, typically musical score and lyrics for a target song are converted to the singing voice. More specifically, the parameters of the singing voice that are F0, spectrum and dynamic are generated using the phonetic and pitch information described in the score. F0, or melodic contour is very important for the naturalness, and expressiveness of the singing voice. Therefore, one of the most important requirements of singing voice synthesis is to obtain an F0 contour that will conduct these characteristics of the singing voice. To achieve this, the factors affecting F0 contour and its characteristics should be carefully investigated and then a framework should be built upon this investigation to generate proper F0 contours.

1.2 Motivation

In the current singing voice synthesizers, users have to make some effort to tune the synthesized singing voice given by the system since the output does not satisfy them in terms of expressiveness and naturalness. For example, in Vocaloid [1], users are allowed to manipulate singing expression parameters such as pitch bend, and dynamics via a user interface, which involves a huge time cost. Therefore, one of the important points in the context of singing voice synthesis is to model these parameters and automatically generate them to provide a better, more expressive and human like initial contour to user. This can be achieved by modeling F0 contours with proper musical contexts.

Pitch contours also represent the style of the singer. Thus modeling a specific style in the singing voice requires the modeling of F0 contours. Analyzing and modeling the F0 contours of the songs from a singer or a singing style may enable singing voice synthesizers to provide the possibility of representing different styles and adjusting F0 output depending on this choice.

1.3 Thesis Statement

In this section we present our thesis statement:

The F0 contour of the singing human voice, similar to other pitched instruments, is highly dependent on the musical contexts. It is possible to capture characteristics and temporal evolution F0 contour of the singing voice by extracting information about the musical contexts. Using this information, statistical models can be trained with properly designed musical contexts to automatically predict F0 contour from the given score. F0 contours with desired properties on the singing voice can be learned with training by example strategy on set of singer recordings, which carry those properties themselves.

1.4 Goals

The specific goals of this thesis work are:

- Building a machine-learning model to automatically generate pitch contours from scores assuring naturalness and expressiveness of the singing voice.
- Proposing musical contexts for modeling F0 contour for the singing voice.
- In particular, to explore the possibility of applying recurrent neural networks to this problem and setting up the first recurrent neural network for the F0 generation problem .

CHAPTER 2

State of The Art

In this section, we will review state of the art in F0 modeling for singing voice synthesis and various applications of recurrent neural networks on sequence problems. First, we will explain approaches focusing just on F0 modeling and sections related to generation and control of F0 parameter from complete singing voice synthesizer frameworks. Then, we will provide RNN applications on sequential problems related to our topic such as text to speech synthesis and music generation

2.1. Performance Driven Approaches

Performance-driven approaches controls singing voice parameters from an external recording of the target song, which is usually performed by the user. Janer, Bonada and Blauw [3] proposed a system in which pitch, dynamics, vibrato and phonetic parameters for a singing voice synthesizer is extracted from user's singing. In this method, F0 is derived from the input voice using a frequency domain method. Then vibrato that is defined by its depth and rate is extracted by processing frequency curve. Because singing voice synthesizer needs a continuous pitch curve, F0 contour is made continuous with interpolation in transition regions or unvoiced phonemes since the variation caused by unvoiced phonemes is not related to singing expression. Finally, all the parameters including F0 are fed to the singing voice synthesizer's internal system.

Nakano and Gato proposed VocaListener[4], developed as a plugin for Vocaloid[1]. In the approach proposed by Janer et al. [3] parameters that system generates are not robust to different synthesizers since different synthesizers and different singer databases will synthesize different outputs with the parameters estimated from user's voice. VocaListener aimed to remove these deflections and to mimic user's voice as exactly as possible. This is done by estimating the pitch iteratively until it becomes very close the F0 contour of user's singing. This iterative approach both guarantees robustness with different synthesizers and gives better imitation of user's singing. Vibrato of the singing is also extracted as depth and rate and provided later to the user so that he can adjust it later. In the case of detunings in pitch user also can shift the pitch.

Speech recordings were also used as an external performance for deriving singing voice parameters. In such a speech to singing synthesis system [11], spectral envelope of the speech is converted to that of singing while F0 is generated after an F0 control model is applied on musical notes. First, a melody contour is created with concatenation of step functions, each having a multiplication factor equal to pitch of a note. Then four different types of fluctuations, which can be encountered in singing performances and related to naturalness of the singing, are added to this melody contour. These fluctuations are overshoot (a deflection exceeding the target note after a note change) vibrato (a quasi-periodic frequency modulation (4-7 Hz)), preparation (a deflection in the direction opposite to a note change observed just before the note change) and fine fluctuation (an irregular frequency fluctuation higher than 10 Hz).

Performance driven approaches apply singing characteristics extracted from user's singing voice to the synthesizer and provide a simple control system over singing expression parameters. Also it is very suitable for representing characteristics of an individual singer. However they also require people who know how to sing. Eliminating this need is one of the most important points of having singing synthesizers. Yet, for people who want to teach their way of singing to the synthesizer this approach provide a very convenient control.

2.2. Rule Based Approaches

In a basic architecture of a rule based singing voice synthesizer [10], score and lyrics information are processed with a set of performance, phonetic and articulatory (related to transition between notes) rules to manipulate singing voice parameters such as pitch, vibrato and sound level. The systems that create singing voice synthesis based on these rules are categorized as rule based systems. The foremost system employing this approach is KTH singing synthesis system [13]. This system is based on KTH rules for music performance, which is a result of a research made through analysis-synthesis method over 30 years [12]. This is a cyclic process in which, a music expert proposes the rules that will manipulate synthesis parameters according to the musical context or lyrics, synthesizer applies this rules in synthesis procedure, listeners evaluate the synthesis subjectively and rules are reviewed and proposed again. In KTH singing system [14], 23 performance rules (16 for musical context, 4 for phonetic, 3 for special singing techniques) that affect F0, formant frequencies, duration, sound level, vibrato depth and rate were used. Six of them are related to F0, named as melodic intonation, the higher the sharper, marcato, Bull's roaring Onset, timing of Pitch Change and coloratura:

Melodic intonation rule takes account of the fact that singers may not follow equal tempered tuning and intentionally make small pitch deviations on some notes [13]. It is thought that the amount of deviation is related to melodic scale [15]. Therefore, this rule assigns amount of intonation to each note in a chromatic scale. The rule "the higher the sharper" recommends to increase the amounts of deviation when the note is played in a higher octave, since some musicians may favor playing higher tones sharper and lower tones flatter.

Marcato and Bull's Roaring Onsets are two cases of ornamentation occurring in specific musical contexts. The rule for marcato recommends setting F0 two semitones below the target note at 60 ms before the target note onset since in the ornamentation F0 fall below the target frequency and go up again. Bull's Roaring Onset is the term coined by the authors for the quick ascending pitch changes at certain onsets. To simulate this, at the first note after the rest, F0 is set to 11 semitones down from target and F0 reached to the target 50ms later.

Timing of the pitch change is a phonetic rule. In the analysis by synthesis method it is found that singing sound strange if the F0 does not reach target pitch at the vowel onset in note transitions. Thus, this rule states to begin to change F0 at preceding consonant and setting F0 to target note frequency at vowel onset. Bonada et al.[16] also employed this rule in their system to improve naturalness.

Coloratura is a term used for rapid sequences of short notes each sung with same single vowel. Thus this rule is applied when there is such a note sequence. It formulates the F0 contour based on the note durations and the direction of intervals between successive notes in the coloratura sections.

Rule-based systems provides a simple, deterministic framework to synthesize singing. However, when a system is fully rule based and rules are applied collectively the output may sound unsatisfactory. Thus, these systems require switches and controllers to activate/deactivate rules and set amount of the effects. Difficulty in the control level increases when the number of control parameters increases. Rule based systems can also represent some aspects of specific styles when rules are decided considering that style. For example KTH rules are generally designed for classical singing. However, with the rule approach it is inescapable to miss many aspects in singing since complexity in singing performance cannot be covered with rules.

2.3. Unit Concatenation Approaches

Synthesizers following unit concatenation approaches takes musical score and lyrics, selects best matching units to the given notes and lyrics from a previously created database consisting of large number of units, where units are sung phones or diphones. Then, selected units are post-processed and concatenated.

Vocaloid[1], commercially the most successful singing voice synthesizer uses this approach to create complete singing synthesis from the best matching units. In this system, pitch is tuned to that of target note when a unit with desired phonetic context for target pitch does not exist in the database. The pitch is manipulated by scaling the each harmonic with the pitch conversion ratio (target pitch/current pitch) in the frequency domain.

Umbert et al. [2] employed unit selection for the generation of expressive pitch contour and providing it to the synthesizers. Here, units are defined as melodic context of three consecutive notes or silences to capture attack and release contours with preceding and succeeding notes. All the notes are sung only as vowels to eliminate to variations of F0 unrelated to singing that are caused by unvoiced phonemes. Sequence of consecutive units for target context is derived with Viterbi algorithm that minimizes a cost function penalizing selection of dissimilar source units or unit sequences to the targets. The selected set of units further pitch and time shifted to match target note duration and pitch. Finally, the units are concatenated to obtain final pitch contour.

Unit concatenation approach lets synthesizer represent a singing characteristics of the singer whose singing excerpts used in the databases. It also provides a high singing synthesis quality and naturalness when the database is large and comprehensive since the outputs are directly coming from the real recordings. The biggest drawback is creating the database and covering different musical contexts (notes sung with all pitches, phonemes or successive notes with different intervals) since transformations applied to units when they do not match to target sequence. Another disadvantage is the need for designing cost functions and setting weights between them.

2.4. Statistical Approaches

Statistical approaches follow the process depicted in Fig. 2.4. It consists of training and synthesis parts. In the training part, singing voice parameters are extracted from the singer database. Contextual factors such as note pitch and duration, and phonemes are fed to statistical model to learn singing voice parameters from data. When the training process is completed, singing voice parameters can be generated from the trained model given a target score and lyrics. In this section we will review different statistical models and contextual factors used in these approaches.

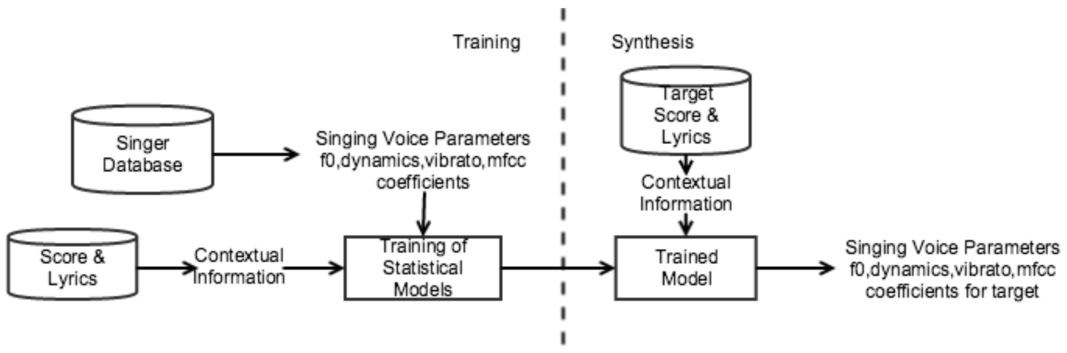
Training
Synthesis
Singer Database
Singing Voice Parameters f0,dynamics,vibrato,mfcc coefficients
Target Score & Lyrics
Contextual Information
Score & Lyrics
Contextual Information
Training of Statistical Models
Trained Model
Singing Voice Parameters f0,dynamics,vibrato,mfcc coefficients for target

Figure 2.1.Basic Architecture of Statistical Approaches

2.4.1. HMM based approaches

Hidden Markov Models are most extensively used statistical model for singing voice synthesis. Followers of this approach are based on a HMM based speech synthesis system [8]. Thus speech synthesis approach will be explained in detail first.

F0 modeling with HMMs in speech synthesis

An N state HMM can be denoted as $\lambda = (A, B, \pi)$ where A is state transition probabilities $\{a_{ij}\}_{i,j=1}^{N}$, B is output probability distribution$\{b_i(o)\}_{i=1}^{N}$ and π is initial state transition probabilities $\{\pi\}_{i=1}^{N}$. In HMM based speech synthesis [8], each HMM represents a phoneme. In addition each HMM has the left-to-right model structure in which only the transitions from lower index state to higher index state is possible through the time. This structure allows better modeling of speech since it is a signal whose properties are successively changing. Output vectors o consists of spectral (Mel Frequency Cepstral Coefficients) and pitch (F0) parameters. Output probability distributions $b_i(o)$ are continuous and modeled by Gaussian Mixtures. Gaussian mixtures allow smooth outputs and modeling complex data. Output distribution parameters (mean and variance vectors) are estimated from training data using expectation-maximization (Baum-Welsch) algorithm [17]. For the generation of outputs, a parameter generation algorithm working in a maximum likelihood sense is proposed for speech synthesis [17]. This algorithm finds the speech parameter vector sequence O^* with the maximum likelihood given HMM λ and sequence length T:

$$O^* = \underset{O}{\operatorname{argmax}} P(O|\lambda, T) \tag{2.1}$$

which is further divided into two sub problems:

$$q^* = \underset{q}{\operatorname{argmax}} P(q|\lambda, T) \tag{2.2}$$

$$O^* = \underset{O}{\operatorname{argmax}} P(O|q^*, \lambda, T) \tag{2.3}$$

The solutions of the equations and derivation of subproblems [18] are not provided here. However, solving the second equation, that is, finding the optimum hidden state sequence q^* given λ and T, requires an essential modification on the structure of HMM, termed as explicit state duration modeling [19]. In an HMM including self-transitions through the states, probability of staying d consecutive steps at state i is given by a geometric distribution $(p_i(d) = {a_{ii}}^{d-1}(1 - a_{ii}))$. Instead of using such a distribution and self transition probabilities, explicit probability distributions such as Gaussians are used. From these distributions, the best hidden state sequence q^* can be directly found since the HMM used has left-to-right structure.

One characteristic of speech synthesis systems is the use of contexts. The output vector for a phoneme is not independent of context it occurs. Therefore context information such as proceeding, current and succeeding phoneme, count of syllables in the phrase, position of the phoneme in the phrase is incorporated to the system. However introducing contexts requires a big training data containing all combinations of contexts. Creating such a database is impossible since the values each factor can take are many. To overcome this problem, context dependent factors are clustered using a decision-tree based context-clustering algorithm [20,21] in HMM approaches, where similar contexts are mapped to same output density distribution via a decision tree. This reduces the need for big training data and allows modeling of unseen contexts since each context is guaranteed to be mapped to one output distribution via decision trees.

For modeling of F0 MSD-HMM (Multi Space Probability Distribution

HMM)[18] is used. F0 contour values are 1-dimensional frequency values through voiced region and undefined for unvoiced regions. Therefore F0 samples are considered to be drawn from two different spaces, one 1-dimensional and other 0-dimensional space. This cannot be modeled by single space probability distribution HMMs unless some assumptions are made for the unvoiced regions. MSD-HMMs can model multi space distributions and solve this problem.

To sum up, hmm based speech synthesis system consists of training and synthesis stages. In the training stage, state duration, spectral and pitch parameter distributions are determined through Baum-Welsch algorithm. Then, decision-tree based context-clustering algorithm is applied to all of these distributions to cluster similar contexts. In the synthesis stage, a given text sequence is converted to context-dependent label sequence. Based on these context dependent labels, from spectrum, pitch and state duration probabilities, a sequence of phoneme HMMs is constructed with associated output probability distribution in the decision tree. Finally, parameter generation algorithm is applied to estimate F0 and spectrum output from the sequence of phoneme HMMs.

F0 modeling with HMMs in singing voice synthesis

HMM-based singing synthesis systems inherit the properties of the HMM-based speech synthesis approach such as the parameter generation algorithm, context dependent HMMs, explicit state duration modeling and MSD-HMMs for F0 modeling. Given a score and lyrics, speech synthesis framework can be used for singing voice synthesis in the manner of "singing the speech" when the defined contexts include factors related to singing voice.

Saino, K et al. [6] used preceding, current, and succeeding phonemes and musical tones, durations and positions of preceding, current and succeeding notes to model MFCCs (Mel Frequency Cepstral Coefficients) and F0. Sinsy[7,9] is a free online singing voice synthesizer using a similar approach. It defines "rich contexts" proposing a lot of contextual factors: (i) key, beat, length and dynamics of the previous, current and next note, (ii) ties and slurs, (iii) distance between the current note and the next, previous accent and staccato, (iv) the position of the current note in the current crescendo and decrescendo, (v) number of phonemes in previous, current and next phrase, (vi) the number of phonemes and phrases in the song. In addition, it extracts vibrato parameters depth and rate and models them with MSD-HMMs (0-dimensional space for nonvibrato and 1-dimensional space for vibrato regions) using the same contextual factors.

HMM based singing voice synthesis approach also inherits the problem of creating training data with different contexts from the speech synthesis approach. The problem is more dominant here since number of different context used is higher. However, for one context specific to singing training data can be extended artificially. This contextual factor is pitch. It is possible to shift F0 contour and note sequence of given song and create a training data with higher pitch coverage [7]. Another solution for the same problem is using relative pitch parameters against corresponding notes in observation vectors [5]. This allows estimation of pitches of the notes in the test songs whose notes are not trained with the database before.

Another HMM based approach [5] estimates only pitch to be used by singing voice synthesizers. It proposes using note HMMs instead of using phoneme HMMs since only the pitch is generated from the model and the pitch of singing voice is assumed to be less dependent on phonemes. The variations in F0 contour caused by consonant are removed from the contour using interpolation. In addition to using note HMMs it introduces behavior-type HMMs where each HMM represents a melodic behavior type occurring inside the notes. These types are beginning (B), sustained (S) and end (E). Each note is assumed to have a behavior pattern consisting of one or more of three behaviors. Then, a behavior pattern for a single note can be B, S, E, BS, BE, SE, or BSE. Tree-based context clustering technique is applied to these patterns, obtaining discrete probability distributions for choosing among 7 discrete behavior-patterns for given contexts. While converting the musical score to HMM sequence, behavior-pattern for each extracted context dependent label is determined through probability distributions and a variable number (one to three) of HMMs are appended where each HMM corresponds to a behavior type. This approach considers the notes as having 3 different regions and model singing expression parameter behaviors in these regions accordingly.

HMM-based systems allow learning of singing voice parameters directly from the training data. Thus, they require less manual work compared to other approaches. For example, rule-designing process in the rule-based systems is implicitly done via learning or transformation and cost functions have to be adjusted in unit concatenation method. HMM approach is also flexible since model parameters can be adapted to different speaker, emotion or style characteristics. On the other hand, statistical approach may cause oversmoothing of singing voice expression parameters since it is working in the sense of statistical averaging and likelihood maximization. This, in turn, may impair naturalness and human-likeness in the generated expression parameters,

2.5 Recurrent Neural Networks Applications

Recurrent neural networks are very powerful sequence models. They are obtained by allowing ordinary multilayer perceptron to have cyclical connections. These cyclical connections provide a very strong theoretical property: an RNN is able to see its entire history of previous input to each output. However, for long time RNNs had not get much attention from the research community due to the difficulties in training in practice. The cause of this difficulty is vanishing gradients problem. This problem prevents RNNs from achieving its strong theoretic power of seeing entire history of inputs. It is found out that vanishing gradient problem makes it hard for an RNN to use its history of inputs more than 10 time steps before [22]. Fortunately, an efficient solution, which uses a special kind of neuron network, is proposed to solve vanishing gradient problem. These are LSTMs [22] (Long-Short Term Memory) and thanks to them RNNs started to get widespread usage in the last decade, which we will exploit in this paper also. In this section, we will provide literature using RNNs for the same goal as our paper: sequence generation.

Fan et al. [23] used Deep Bidirectional RNNs (DBLSTMs) with LSTMs to achieve text-to-speech (TTS) synthesis. DBLSTMs are deep, that is they have multiple hidden layers and deep representation of the input features as deep neural networks (DNNs). Deep-layer architectures allow complex function estimations by creating composite features of features in each deep hidden layer. Bidirectional structure allows accessing future context as well as preceding context [22]. The system converts utterances to a feature sequence through text analysis and then these sequences are mapped with output vectors to provide training example pairs. To train the RNN, back-propagation through time algorithm with a mean squared error function is used. TTS with DBLSTMs outperformed both HMM and DNN models in perceptual AB preference tests.

RNNs with LSTM are also applied to music related tasks. Eck et al. [24] used next-step prediction approach for music composition. In such approach, the network learns to predict the next note depending on all the previous notes. Data is encoded as binary on/off vector, representing which notes are played and which are not. These encodings are pushed into input layers of the network and the network predicts an output in same style of encoding; which notes should be active given all previous predictions.12-bar blues style is used for constructing training test and a binary probability distribution is used for predicting on and off state for each note. With this configuration RNN-LSTM was able to compose music that carry characteristics of 12-bar blues style

Graves used RNNs with LSTMs for handwriting prediction, which is related to our task because a sequence of continuous values is predicted. [26] Handwriting is encoded as two dimensional real-valued x-y pen-tip locations. The network generates such x-y values at each time step and invents handwritings. To model real valued outputs, mixture density network is used as probability distribution of the network. At the output layer of the network, a Gaussian mixture is predicted at each time step, that is, output neurons output set of means, standard deviations, correlations and mixture weights of the bivariate mixture components. The network is trained by maximizing the log probability densities of the target vectors with a training data consisting of text and its handwritten version encoded in x-y values. The experiments showed that the network learned strokes, letters and even common words such as “of” and “the” by modeling the problem only as next-step x-y point prediction.

CHAPTER 3

Methodology

3.1.Software

RNNLIB [33] is used for implementing recurrent neural network experiments. It is a stable recurrent neural network library for sequence learning tasks implemented in C++ and suitable for running low-and-high scale experiments. However, we needed regression for our task and it was not supported in the last version although it had been active on previous versions. Therefore, we reimplemented regression learning on the last version based on the previous versions [34].

Vocaloid[1], a singing voice synthesizer based on unit concatenation, is used to generate final singing voice outputs. Our model directly provides the F0 contour and Vocaloid applies this information to samples from the voice database while generating singing voice synthesis.

3.2.Dataset

Two singing voice database are used for learning F0 contours: melodic exercises database and jazz songs database. They are recorded with vocal and background music related to the style it is intended to convey. The background music helped the singer sing in tune and evoke the target style. The singing expression contours and score files are derived from the recordings semi-automatically.

Vocal sections are sung only with vowels. Any note sequence is sung as sequence of /ua-i-a-i-…/ where /i/ and /a/ alternated after first note. This approach allows controlling F0 for any target lyrics since unvoiced sections or micro-prosody due to the phonetics (not related to expression) are intrinsically removed and the model becomes lyric independent.

Systematic melodic exercise database (SysDB):

Systematic melodic exercise database are designed so that a pitch space of one octave and different musical intervals will be covered. Since these are just melodic exercises, there is relatively less expression and emotion involved in the recordings. Rather, it is a recording database in which specific intervals and pitches are sung correctly and in tune. After training with database, we would expect to get natural sounding but not much expressive synthesis outputs

Jazz songs database (SongDB):

It consists of standard jazz songs. The recorded songs are selected from a repertoire known by the singer. Therefore, here, the singer could put more expression and style specific properties to the singing. Thus, from a model trained with database we would like to represent these characteristics. The number of recordings and total duration in the two databases are given in Table 3.1.

Table 3.1. Statistics for singing voice databases

Database	Number of Recordings	Duration
SysDB	70	11 min 59 sec
SongDB	15	16 min 59 sec

3.3. Computational Modeling

In this section, we will provide the methodology with which we used to learn and predict F0 contours. We used recurrent neural networks as the statistical learning model. First, we will explain the RNN structure, and training procedure. Then we will provide how we model the singing expression contours.

3.3.1. RNN structure

Recurrent neural network structure used in this paper is depicted in Fig. 3.1.

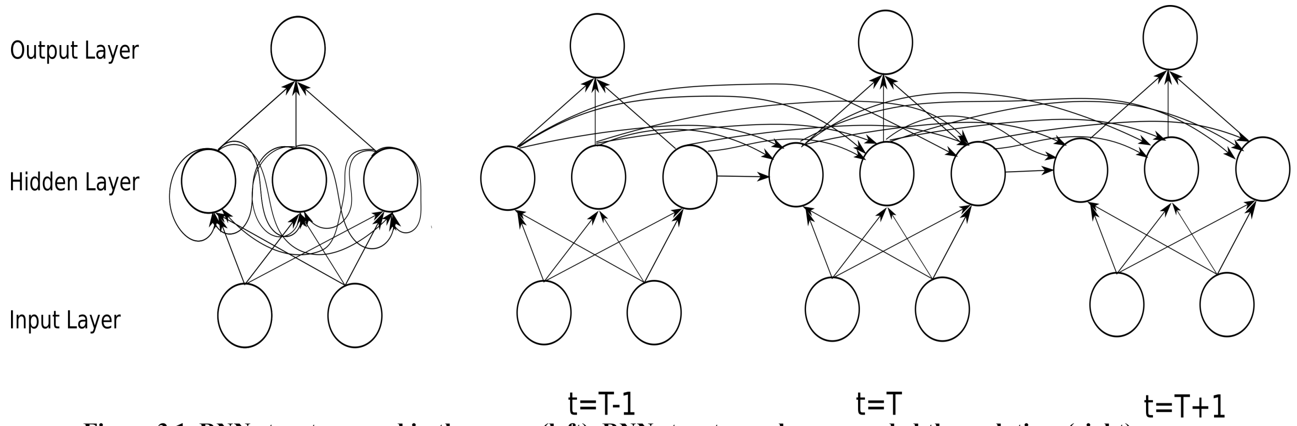

Figure 3.1. RNN structure used in the paper (left). RNN structure when expanded through time (right)

The difference from multilayer perceptron is that hidden layer is fully connected to itself (every hidden unit connected to all other recurrently). With this small change, RNNs can see and map their entire history of inputs to each output since recurrent connections between each time step allow inputs to stay in network's internal state. [22]. More formally, recurrent Neural Network (RNN) computes hidden state vector sequence $h_1, h_2 \ldots h_T$ and outputs vector sequence $y_1, y_2 \ldots y_T$, for a given input vector sequence $x_1, x_2 \ldots x_T$, iterating the following equations from t= 1 to t=T:

$$O = h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.1}$$

$$y_t = W_{hy}h_t + b_y \tag{3.2}$$

where W represent weight matrix between two layer, e.g. W_{hh} is weights of recurrent connections between the hidden layer, $\mathcal{H}$ represent a nonlinear activation function such as tanh and b represent biases of the layers.

However, RNN's ability of seeing all the history only remains in theory as explained in section 2.5. The previous inputs cannot stay for long delays in the network because of the vanishing gradient problem. To solve this problem, we used LSTM blocks instead of standard cells with logistic or tanh activation function in the hidden layer.

The structure LSTM block is given in Fig. 3.2. It consists of an input and output neuron, three multiplicative units, input, output and forgets gates that act as write, read and reset operations [22]. These multiplicative gates allows keeping a previous input in the memory of the network over long periods of time and access when it is needed, eliminating the vanishing gradient problem. LSTM is implemented as follows:

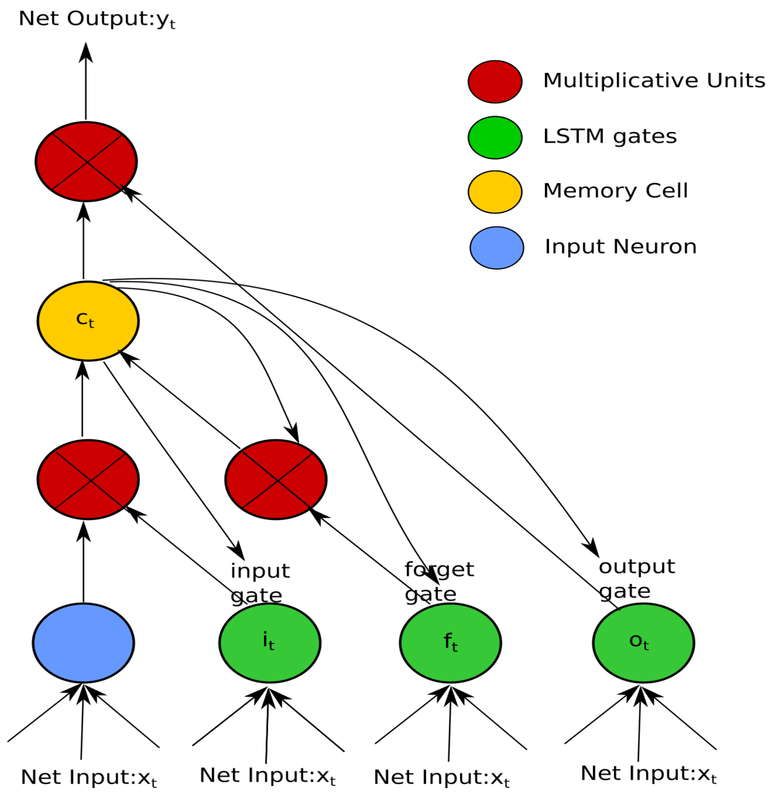

Figure 3.2.LSTM structure

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i \tag{3.3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f \tag{3.4}$$

$$c_t = f_t c_{t-1} + i_t \tanh\,(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3.5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{3.6}$$

$$h_t = o_t \tanh\,(c_t) \tag{3.7}$$

where σ is the activation function and *f,i,o,c* are forget,input and output gates and memory cell.

3.3.2. Error function and Training Procedure

RNNs are differentiable by construction[22]. Therefore, gradient descent algorithms that minimize an objective error function of training-target examples can train them. Objective error function differs according to type of the task learned such as regression or classification. In this paper, we applied regression with sum squares error since we simply predict a continuous value in each time frame given input features. Let z and y be the target, network output pairs. Then error function O for regression is given by

$$O = \sum (z - y)^2 \tag{3.8}$$

where y is given by linear combination of input activations of the output unit(eqn. 1,2).

Gradient descent methods minimize objective error function by finding the derivatives of objective function with respect to network weights and adjusting the weights in the negative slope of the derivative. In this paper, steepest descent algorithm is used. In this algorithm, after each training example, each weight w in the network is updated as follows:

$$w_{n+1} = w_n + \Delta w_n \tag{3.9}$$

$$\Delta w_n = m \Delta w_{n-1} - \alpha \frac{\partial O}{\partial w_n} \tag{3.10}$$

$$where\ 0 \leq \alpha, m \leq 1$$

The second term $(\alpha \frac{\partial O}{\partial w_n})$ in error gradient (Δw_n) is the adjustment in the direction of negative slope of the error derivative and α represent the amount of change that will be applied and named as "learning rate". The first term is the momentum term in 3.10. A ratio of the previous gradient is also added to the error update. This helps steepest descent algorithm to converge fast and to avoid getting stuck at local minimum points. The momentum and learning parameters are generally chosen experimentally depending on the dataset, learning task and other training related factors.

In multilayer perceptrons, back propagation algorithm is used to efficiently calculate error derivatives $\frac{\partial O}{\partial w_n}$. It first finds the derivatives of objective function with respect to network outputs and propagates error terms from output through input layer and calculates error gradients in each layer by repeated application of chain rule. In this work, we used back propagation through time algorithm [22], which is also based on back propagation but modified for recurrent neural works to handle the peculiarity in RNNs: objective function not only depends on the current activations (as in standard neural networks) but also the previous history

3.4.Learning Singing Expression contours

Our framework is given by the Fig. 3.3.

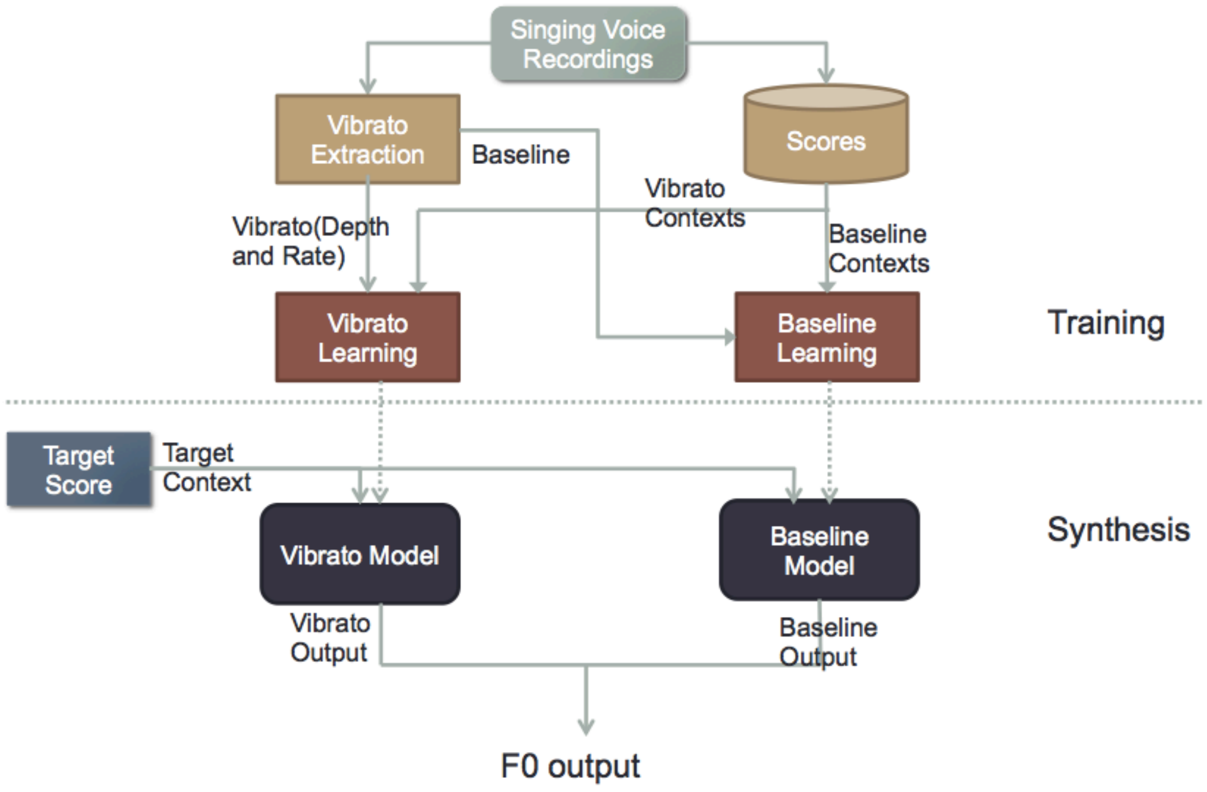

Figure 3.3.General Framework

Scores are semi-automatically derived from singing voice recordings by labeling the notes in the F0 contour. Total F0 is separated into baseline and vibrato sequences for each song. Then, vibrato and baseline are learned by training the vibrato and baseline streams with contextual information derived from the corresponding scores. Finally, after models are created with learning procedure, given a target score vibrato and baseline models can generate vibrato and baseline output of the target score and these are summed to get total F0 output.

One important point not given in the Fig.3.3. is that, we used phrases as the training examples, not the whole songs. We extracted phrases from songs by dividing each song by the rests in the musical score of the song. Therefore, we eliminated all the rest and only modeled the notes. The reason behind is that our model fit a regression; on the other hand F0 contour or vibrato parameters during rest regions are undefined. Thus, it is unreasonable to predict the values during these regions simply by a regression model. Either they should be removed from training process or a multi-space probability distribution that decides if a frame is rest or not should be used. However, there is one drawback of this operation: since we reduced training sequences from whole excerpts to note sequences they become independent of each other and one phrase in a song cannot see history that would be formed by previous phrases in the same song. On the other hand, it also allow network to learn better in phrase level.

The learning in RNNs is done at frame level. That is, given a phrase the output sequences are streams of F0 or vibrato values of all the audio frames as well as input sequences are streams of contextual information of all the audio frames and there is a one to mapping between input and output sequences. Our model learns to predict output value (F0 or vibrato) sequence given contextual information sequence of all the frames of a target phrase.

To create singing expression contours, we modeled vibrato and baseline F0 contour separately. Thus, F0 training data is converted to vibrato data and baseline data. The baseline is the remaining part when vibratos are excluded from the F0 contour. Vibrato is defined by vibrato depth and rate at each time frame. The relation between the baseline, vibrato parameters and the total F0 contour is given by the equations[2]:

$$\varphi(n) = \sum_{k=0}^{n-1} 2\pi r(k)\Delta_t + \varphi_{correc}(n) \tag{3.11}$$

$$\widetilde{F0}(n) = \overline{F0}(n) + d(n)\sin\big(\varphi(n) + \varphi_{sign}\big) \tag{3.12}$$

where $\widetilde{F0}(n)$ is reconstructed total F0, $\overline{F0}(n)$ is baseline pitch at frame n, $d(n)$ is vibrato depth at frame n, $\varphi(n)$ is sinusoid's phase and φ_{sign} value indicating initial phase, $r(k)$ is vibrato rate at frame k, Δ_t is frame time and $\varphi_{correc}(n)$ is reconstruction error

3.4.1.Learning baseline

F0 contour carries important characteristics of the singing and related to the naturalness [28]. Behaviors in F0 contour such as overshoot and preparation or fine fluctuations [29] exists in human singing and found be to related to musical contexts [30]. Thus we should be able to model and simulate this kind of behaviors within our statistical model.

An RNN-LSTM with a linear output neuron trained with sum squares error function is used to learn baseline F0. The following contextual factors are provided as input to the network:

1. The pitch of the note in the current frame in units of integer MIDI notes.
2. The interval between current note and the next note in semitones.
3. The interval between current note and the previous note in semitones.
4. The duration of the current note in seconds.
5. The position of the frame within the current note. For the nth frame in a note it is given as n / total number of frames in the note.
6. Binary feature telling if the next note is a rest
7. Binary feature telling if the previous note is a rest

Current pitch feature is important for system to learn to be intune with the notes since F0 contour follows the frequency of the note with small deviations. Frame position within the note helps learning amounts of these deviations through the note. Interval between next note and previous note or the information of whether they are rest is important for the beginning and end regions of the notes and modeling the transitions made from the previous or to the next note.

The network configurations and statistics of SysDB and SongDB for learning baseline is as in Table 3.2.

Table 3.2. Network configurations and statistics for melodic exercises and jazz songs database

Database	Number of Frames	Number of Sequences	Learning Rate	Momentum	Hidden Layer Size	Number Of Epochs
SysDB	123873	154	10^{-5}	0.9	120	300
SongDB	145237	196	10^{-5}	0.9	120	324

Learning rate and momentum corresponds the parameters in equations 9 and 10. Hidden layer size is the number of neurons in the hidden layer and is chosen experimentally. Epoch is one-step iteration of training with all the input dataset. The number of epochs is not selected beforehand. Instead, a threshold that states the number of subsequent epochs without improvement on the network error is provided. The network automatically stops when it reached to this threshold. This threshold was given as 30 for both database and the networks automatically stopped when they reached to 300th and 324th epoch for SysDB and SongDB respectively.

Pitch shifted data

Pitch as a contextual factor is very important for model to generate proper and intune F0 contours since the frequency of the note and the frequency that singer sings are very correlated. Therefore, this contextual factor should be covered in training set as much as possible. However, our database had a sparse distribution of pitch and also had a limited range. In our first experiments, this caused some undesired outputs for the target contexts where their pitch is not covered in database (Fig. 3.4).

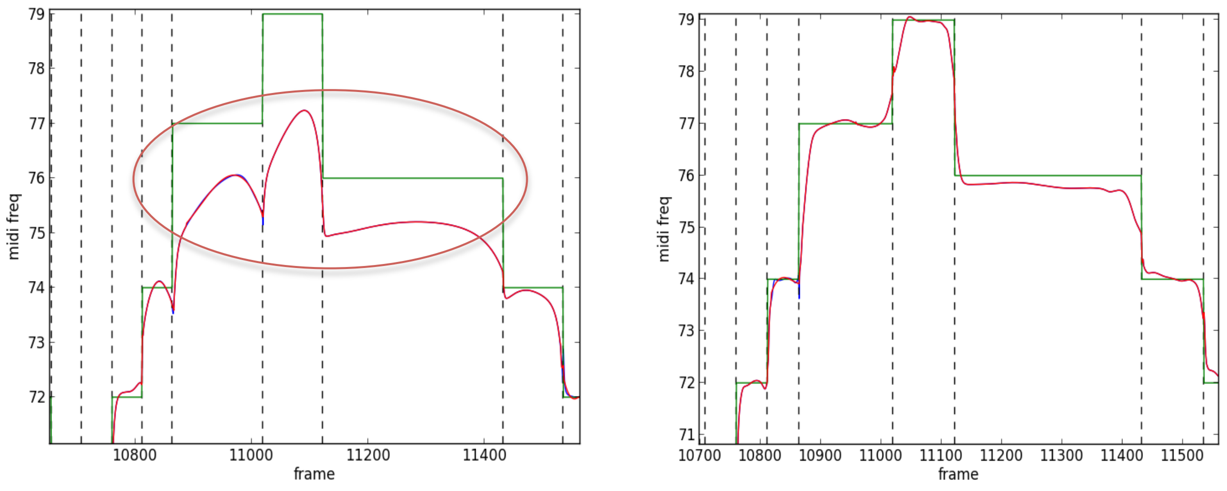

Figure 3.4. Detunings in pitch contour before and after pitch shifting.The model trained with SysDB fails to be intune with some target notes which is out of pitch range of the training set(left).The problem is solved after pitch shifting(right)

To solve this problem, we applied pseudo-pitch shifting[7] to our database. Since F0 contours are just represented in logF0 (midi) units, new training examples can be generated by shifting the F0 contours in semitones. This is an easy way to increase F0 training data and cover pitch context better. We applied a shifting of +/- 1 and +/-6 semitones to all training songs. After this operation, we got the pitch distributions in the Fig.3.5 and get a better coverage of the pitch.

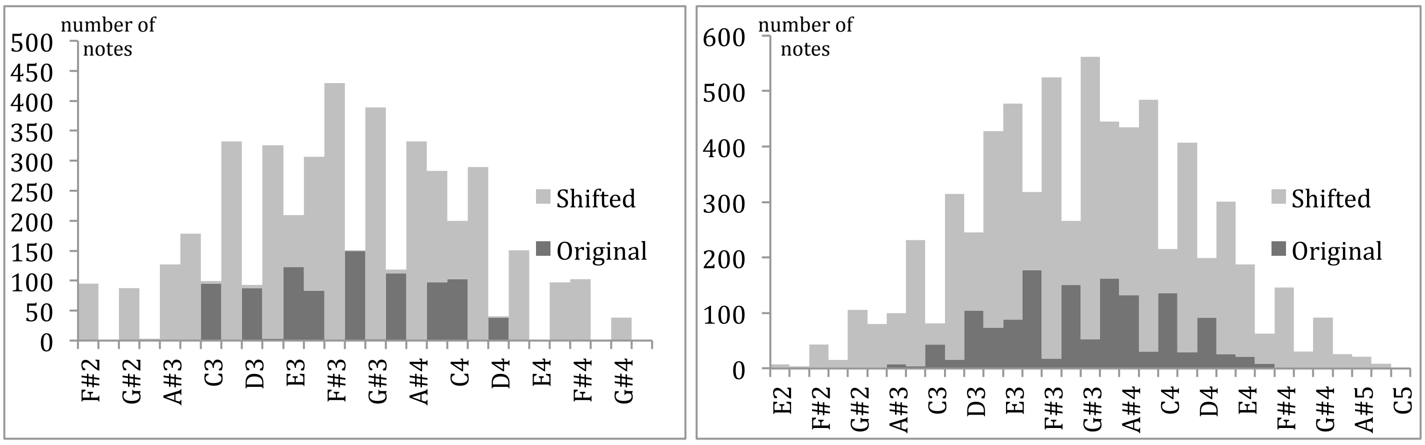

Figure 3.5.Distributions of notes before and after pitch shifting for SysDB(left) and SongDB(right).Y axis is the number of occurrences.

3.4.2.Learning Vibrato

Vibrato corresponds to a modulation of the frequency and characterized by the rate and depth of this modulation. It is an important technique that is used by singers to add expression to the performance. In addition, its characteristics change depending on the singer and style. Thus, it is needed to incorporate vibrato modeling in our model. Musical contexts and aspects of vibrato are found to be related. For example, Prame found out that depth of the vibrato is affected by the duration of the note, and the relative position inside a note [27]. Thus, it is reasonable to train and predict the vibrato characteristics by providing contextual information of the note.

Vibrato prediction is done with two separate networks. (Fig.3.6) One model consists of a RNN-LSTM with two linear output neurons predicts vibrato depth and rate simultaneously. This model predicts a continuous two-dimensional stream for a given sequence. However, vibrato depth and rate are defined only when the frame is vibrato frame. When the frame is unvibrato frame the network should predict 0.This is not possible with a simple regression model that minimize sum squares error since the network will always predict a continuous value in an interval. Therefore we created another model based on binary classification to predict whether the frame is a vibrato or nonvibrato frame. To get the final output, output of the two models for a given sequence are multiplied so that binary classification model will act as a mask and will filter the nonzero vibrato depth and rate predictions from the other model, which actually come from a nonvibrato frame (Fig 3.7.). The objective function of the binary classification network is given as follows [22]:

$$O = -\sum z ln y + (1 - z) \ln(1 - y) \tag{3.8}$$

where z and y are target and network output pairs

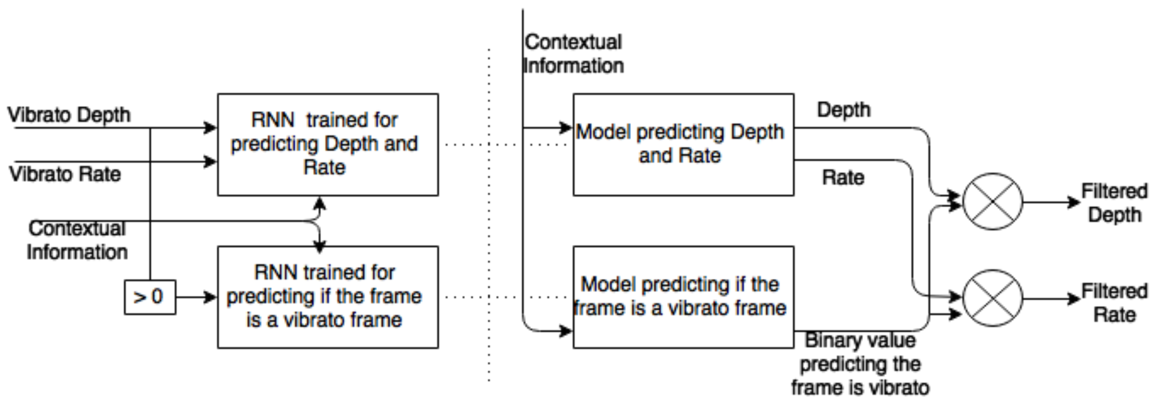
Vibrato Depth
Vibrato Rate
Contextual Information
> 0
RNN trained for predicting Depth and Rate
RNN trained for predicting if the frame is a vibrato frame
Contextual Information
Model predicting Depth and Rate
Model predicting if the frame is a vibrato frame
Depth
Rate
Binary value predicting the frame is vibrato
Filtered Depth
Filtered Rate

Figure 3.6.Vibrato Framework

Following features are used in both of the networks:

1. The duration of the current note in seconds.
2. The position of the frame within the current note. For the nth frame in a note it is given as n / total number of frames in the note.
3. Binary feature telling if the next note is a rest
4. Phrase position of the note

Many vibratos occur when the next note is a rest since these represents generally phrase endings and the singer generally accents this climax point with a vibrato. However, vibrato can also occur at other points occasionally. Therefore, phrase position is provided to predict such points. The other features are chosen especially for predicting the timing and intensity of the vibrato.

The network configuration of SysDB and SongDB for learning vibrato is as follow: learning rate, momentum, and hidden layer size is chosen as 10^{-6}, 0.9, and 40, respectively for both database. The network trained with SysDB to predict depth and rate stopped after 670 epochs while the network predicting vibrato frames stopped at 630 epochs. The network trained with SongDB predicting depth and rate stopped after 843 epochs and the network predicting vibrato frames stopped at 810 epochs.

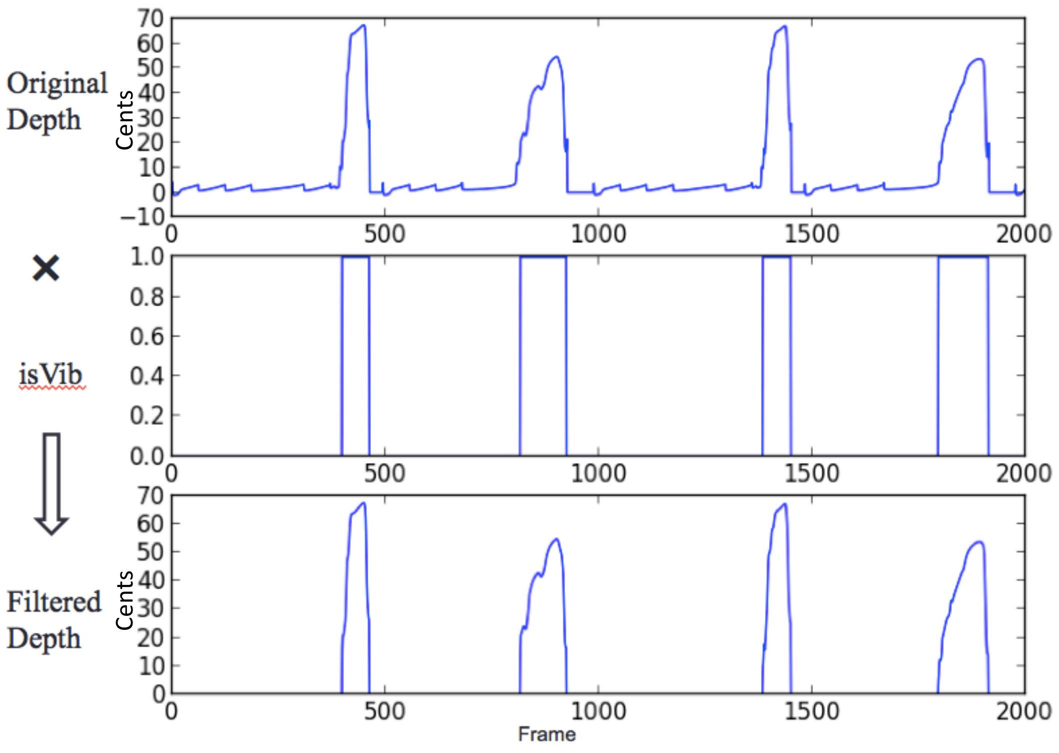

Figure 3.7. Vibrato models and their interaction. To get the final output, output of the two models (original depth prediction and isVib,binary output telling if the frame is vibrato frame) for a given sequence are multiplied

CHAPTER 4

Results

We applied RNNs for modeling F0 of the singing voice. In this section, we will explain how our model generates final outputs, discuss some consistent behavior we observed in contour outputs and provide our qualitative evaluation methodology and its results.

4.1.Output Generation

Given a musical score, we first divide it into the phrases since it is the basic unit of our database, for each phrase we generate F0 contours from the vibrato and baseline models, and concatenate them in order to create complete F0 stream. This F0 stream is provided to Vocaloid. Dynamics parameter is provided as a constant stream since we haven't modeled dynamics. Vocaloid handles the remaining processes in the synthesis and returns wav files for each song.

We generated outputs for 7 targets that are some jazz songs, which is listed online [35]. Target information are parsed from score files (MusicXML[32]) and since they are just scores the timing is not natural. Although it would be better to derive scores from human performances (for the qualitative test for example), this is not the main issue in our work. We wanted to evaluate F0 contours by holding the other factors identical.

One additional modification while we generate the synthesized audios was applying a smoothing to baseline contour at note boundaries(fig 4.1.). This is because we encountered some discontinuities at the note boundaries because of the sharp context changes at the note boundaries. We could not find a way to solve this problem from scratch (from our networks directly), instead we applied a smoothing at the frames close to the note transitions. Although there was no difference at the synthesized audios with and without smoothing since the discontinuity occurs at just few frames, it might cause problems for the possible target songs we did not cover in our tests. The method ideally should avoid any discontinuities at the boundaries.

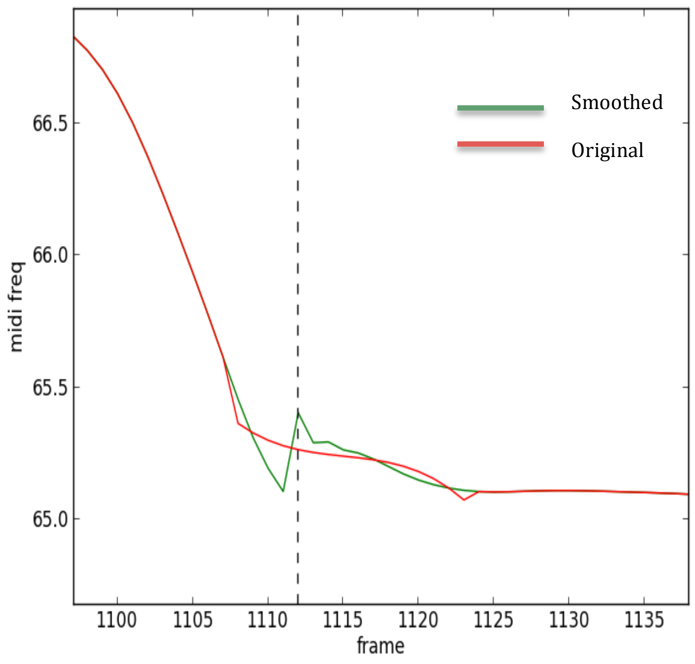

Figure 4.1. Discontinuities (labeled green) at note transitions.

4.2. Contour Samples

In this section we will describe the results of F0 and vibrato modeling by presenting some common behaviors learned by the models. One such behavior is that our model learns to exhibit overshoot (deflection of F0 contour exceeding the target note after note changes) and preparation (deflection of the opposite direction of note change observed just before note changes), which are F0 fluctuations that occur in human singing and related to naturalness [29] (Fig. 4.2.).

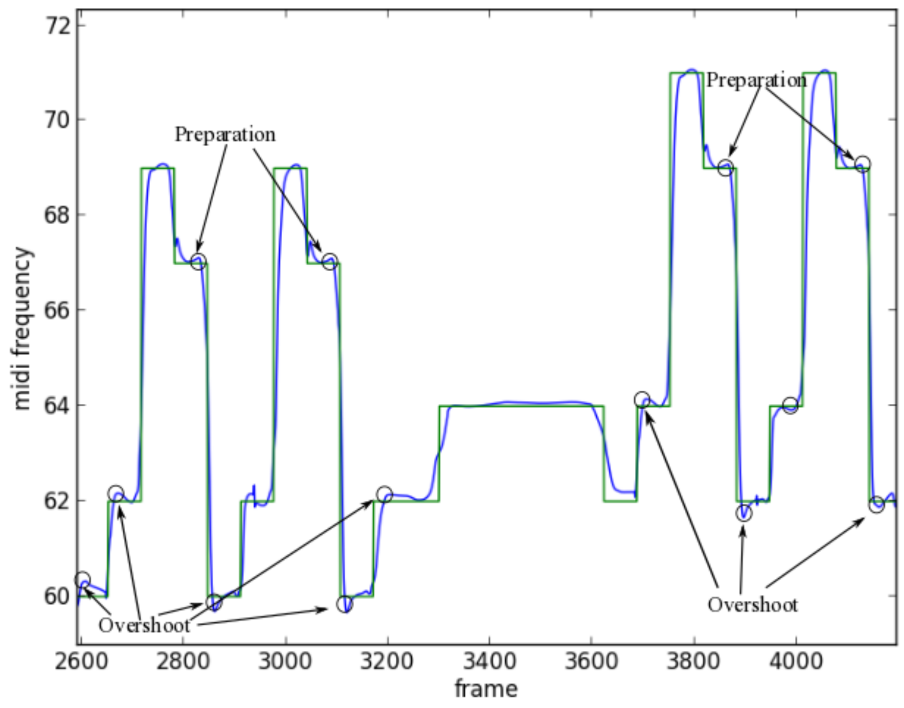

Figure 4.2. An excerpt from September In The Rain output showing overshoots and preparations

Being in tune with the notes is also essential for the F0 modeling systems. It is a must to avoid detunings and mistakes in singing. Our model also successfully learned being in tune (Fig. 4.3.).

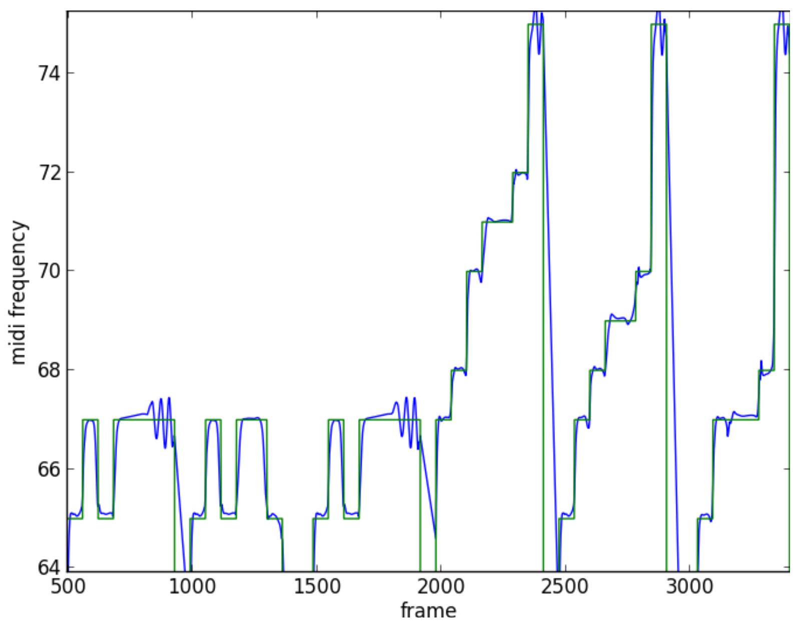

Figure 4.3. An excerpt from output for the But Not For Me, which is succesfully in tune.

The vibratos are also learned properly, increasing depth and rate through the beginning and smoothly decreasing at the end of the note. However, the vibratos are predicted only at the phrase endings (i.e. when the next note is a rest). This is one problem of our method. Although the phrase endings generally involves a vibrato in singing the model should also predict vibratos in other positions occasionally.

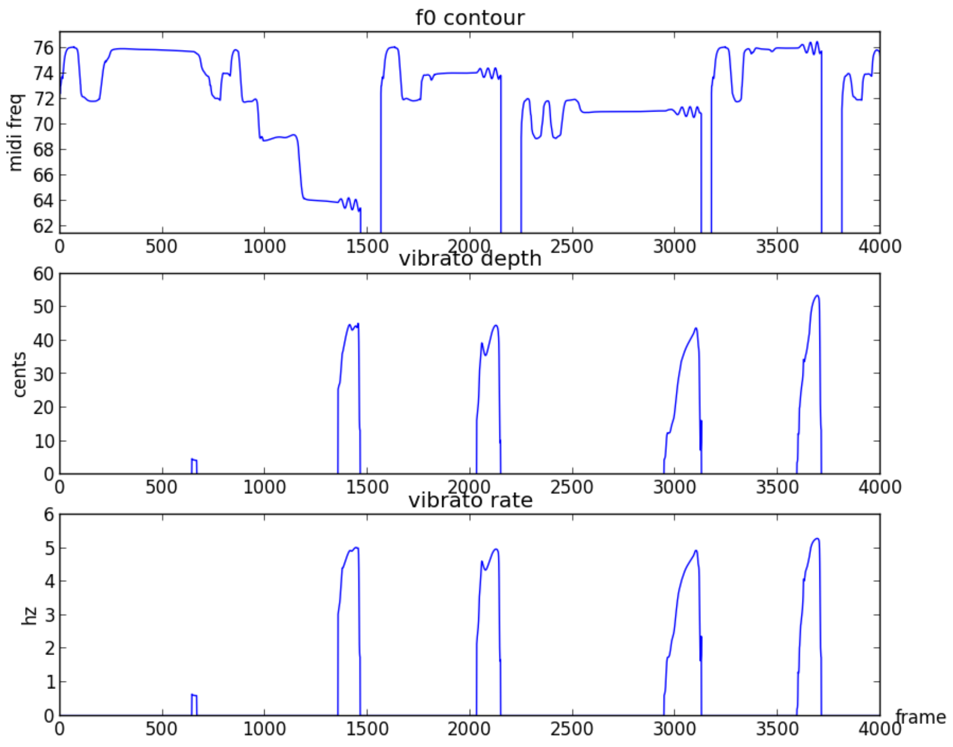

Figure 4.4.Vibrato predictions for Stars Fell on Alabama. Total F0, depth and rate from top the bottom

4.3. Subjective Evaluation

We compared our method with 3 different methods: HMM approach, Unit Selection Approach, and default strategy of Vocaloid. We used a perceptual A/B preference test as the evaluation method. An online survey is prepared for the test [31]. In the test, users do a comparison between pairs of synthesis methods by stating the better audio or giving a “no preference” option and they consider naturalness and expressiveness as the evaluation criteria. For five target songs in the target set, outputs are created with each of the 4 methods (RNN,HMM,Unit Selection, and Default Vocaloid), and for each song RNN is compared against hidden markov models, Unit Selection, and Default Vocaloid.

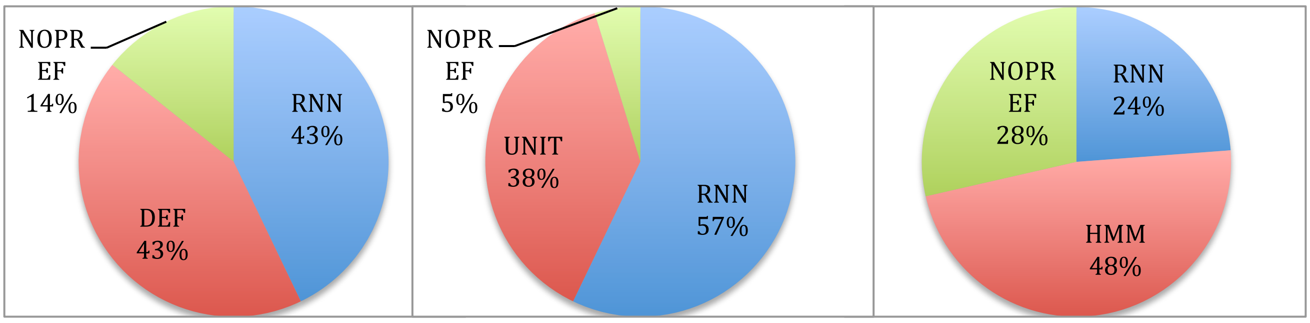

Figure 4.5.AB preference results for Summertime

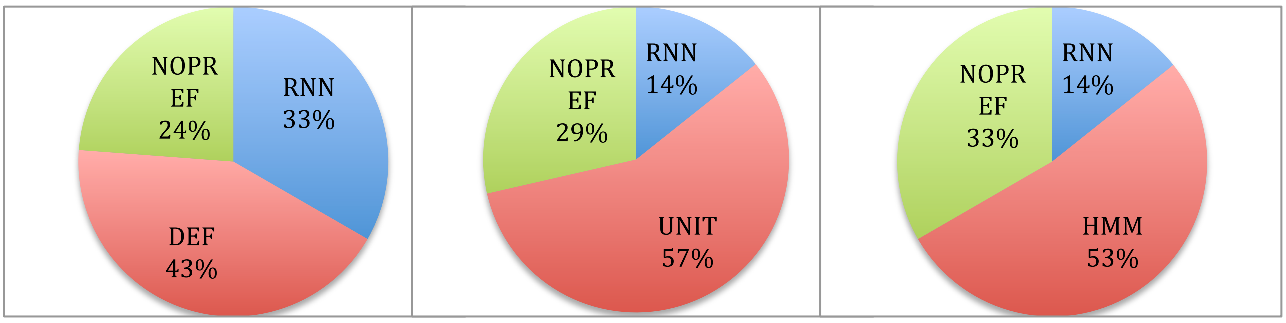

Figure 4.6. AB preference results for But Not For Me

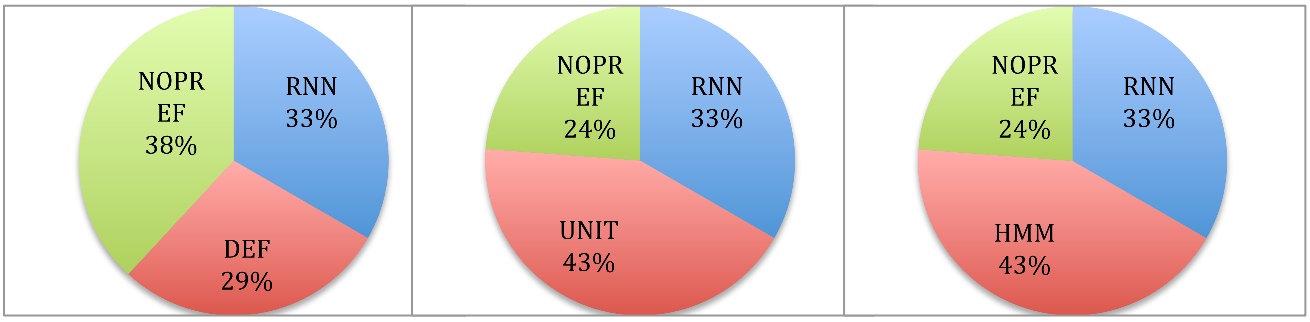

Figure 4.7. AB preference results for I Thought About You

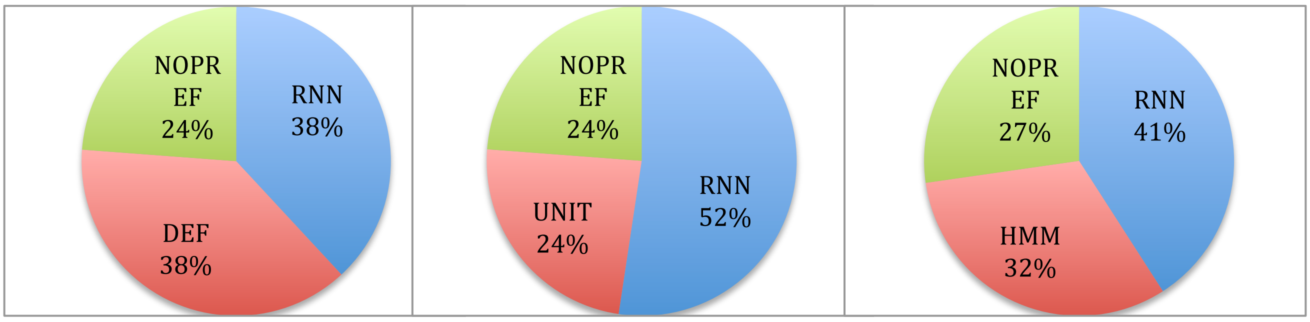

Figure 4.8. AB preference results for September In The Rain

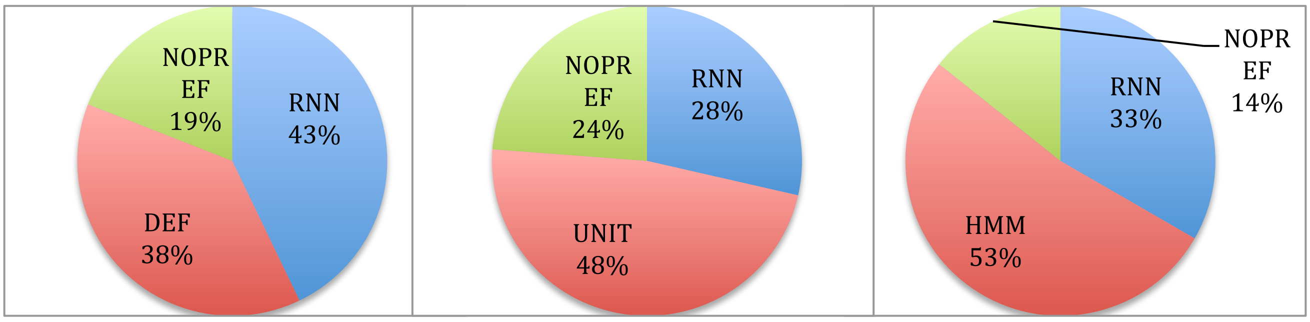

Figure 4.9. AB preference results for Stars Fell On Alabama

The results of test made with 22 people are shown on Fig. 4.5, 4.6, 4.7,4.8 and 4.9. Preference percentages are given for each song for RNN-DEF, RNN-UNIT, and RNN_HMM comparisons,where RNN is our method, DEF is default Vocaloid synthesis, UNIT is unit selection and HMM is hidden markov model approach

From these results we can state that rnn and def performance are very similar since in all the songs rnn-def comparison showing a tendency towards equal percentages. In, rnn-hmm comparison we can say that hmm outperformed rnn for 4 songs out of 5, therefore it seems to be better than RNN. Between unit selection and RNN it is hard to decide which is better since for some songs RNN and for some other UNIT is preferred. In RNN-DEF comparison percentages are close to each other in all the cases.

From the results, we would at least expect to outperform the default method. Also, we were expecting at least to get close results with HMM. However, when we listened pairs in which our method performs much worse than the other (e.g RNN-HMM comparison for But Not For Me) or much better than other (e.g. RNN-UNIT comparison for September In The Rain) it was very hard to explain the reason for such differences only by listening. The reason for is that synthesized audios are very similar to each other and hard to differentiate although there are slight differences between audios. Also we never see a common method that is always preferred in different songs of the same comparison pair. Thus, we think that our method performs very close to other methods and works reasonably well. Using a different dataset with higher size and variety or modeling also the dynamics might expose real pros and cons of the different methods.

CHAPTER 5

Conclusions and Future Work

In this study, we have build up a framework for generating F0 contours automatically from given scores. We have used recurrent neural network with LSTMs to model our data and create our predictor. We have a decent statistical model that is able to represent F0 fluctuations occurring in human singing such as overshoot, preparation, and vibrato, and avoid detunings.

Our main contribution is that this work is the first application of Recurrent Neural Networks to F0 modeling for singing voice synthesis. Although we could not outperform the competent methods in our subjective listening tests, we provided a good starting point by creating a decent model. With their power of modeling complex data by providing compact nonlinear functions, RNNs has a huge potential than the others for sequence problems such as f0 modeling. They are also very flexible and powerful against modeling complex and large data, which are the properties that will be exploited more in the future.

Limitations and Future Work

Some changes for enhancing our work would be as follows:

- Changing basic training units from phrases to songs. Currently we were dividing whole songs to the phrases and this prevents RNNs from learning dependencies between contexts in two different phrases of the same song. This can be solved by adding another Recurrent Neural Network with binary classification layer, which predicts if the note is a rest or not.

- We encountered with the small discontinuities at the note boundaries. Although these did not make differences on the synthesized audios, it may cause problems in the other possible targets. To solve this problem, a Gaussian probability distribution can be used at the output layer and smoother outputs could be obtained.

- The vibrato model only predicts vibratos at phrase endings. One reasonable explanation is the examples that training database provide contains this behavior. However, vibratos also occur inside the phrases even if it is not as much as at the phrase endings. Thus, the statistical model should occassionaly predict vibratos inside phrase. This can be achieved by using different network configurations (different hidden layer size, features etc.)

- Our model is missing the dynamics. It is also very valuable for representing expressiveness and naturalness. It can be modeled in the same way F0 modeled, with a regression RNN-LSTM.

References

1. Kenmochi, H., & Ohshita, H. (2007, August). VOCALOID-commercial singing synthesizer based on sample concatenation. *INTERSPEECH* (pp. 4009-4010).
2. Umbert, M., Bonada, J., & Blaauw, M. (2013, July). Generating singing voice expression contours based on unit selection. In *Proc. SMAC*.
3. Janer, J., Bonada, J., & Blaauw, M. (2006, September). Performance-driven control for sample-based singing voice synthesis. In *Proc. of DAFx*, 6, 41-44.
4. Nakano, T., & Goto, M. (2009). VocaListener: A singing-to-singing synthesis system based on iterative parameter estimation.In *Proc. of SMC* (pp. 343-348).
5. Saino, K., Tachibana, M., & Kenmochi, H. (2010). A singing style modeling system for singing voice synthesizers.*INTERSPEECH(pp.2894-2897).*
6. Saino, K., Zen, H., Nankaku, Y., Lee, A., & Tokuda, K. (2006, September). An HMM-based singing voice synthesis system. In *Proc. of ICSLP*, 9, 1141-1144.
7. Oura, K., Mase, A., Yamada, T., Muto, S., Nankaku, Y., & Tokuda, K. (2010). Recent development of the HMM-based singing voice synthesis system—Sinsy. *Seventh ISCA Workshop on Speech Synthesis*.
8. Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., & Kitamura, T. (1999). Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis.*Proc. of Eurospeech*,(pp. 2347-2350).
9. HMM-Based Singing Voice Synthesis System (Sinsy), http://www.sinsy.jp/
10. Rodet, X. (2002, November). Synthesis and processing of the singing voice. *Proc. 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA-2002)* (pp. 15-31).
11. Saitou, T., Goto, M., Unoki, M., & Akagi, M. (2007, October). Speech-to-singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices. *Applications of Signal Processing to Audio and Acoustics, 2007 IEEE Workshop on* (pp. 215-218).
12. Friberg, A. (1991). Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15(2), 56-71.
13. Sundberg, J. (2006). The KTH synthesis of singing. *Advances in cognitive Psychology*, 2,131-143.
14. Berndtsson, G. (1996). The KTH rule system for singing synthesis. *Computer Music Journal*, 20, 76-91.
15. Friberg, A., Bresin, R., & Sundberg, J. (2006). Overview of the KTH rule system for musical performance. *Advances in Cognitive Psychology*, *2*(2-3), 145-161.
16. Bonada, J., Celma, O., Loscos, À., Ortolà, J., & Serra, X.(2001). Singing Voice Synthesis Combining Excitation plus Resonance and Sinusoidal plus Residual Models. MTG,Universidad Pompeu Fabra, Spain.

17. Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., & Kitamura, T. (2000). Speech parameter generation algorithms for HMM- based speech synthesis. *Proc. ICASSP*, 3, 1315–1318.
18. Yamagishi, J. (2006). An introduction to hmm-based speech synthesis. Technical report, Tokyo Institute of Technology.
19. Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., & Kitamura, T. (1998). Duration Modeling in HMM-based Speech Synthesis System. *Proc. of ICSLP*, 2, 29–32.
20. Young, S. J., Odell, J. J., & Woodland, P. C. (1994, March). Tree-based state tying for high accuracy acoustic modelling. *Proc. ARPA Human Language Technology Workshop*, (pp.307–312).
21. Shinoda, K., & Watanabe, T. (2000). MDL-based context-dependent subword modeling for speech recognition. *The Journal of the Acoustical Society of Japan (E)*, *21*(2), 79-86.
22. Graves, A (2008). Supervised sequence labelling with recurrent neural networks. PhD thesis, Technical University Munich.
23. Fan, Y., Qian, Y., Xie, F., & Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks. *Proc. Interspeech* (pp. 1964-1968).
24. Eck, D., & Schmidhuber, J. (2002). A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*.
25. Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.
26. Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.
27. Prame, E. (1997,July). Vibrato extent and intonation in professional Western lyric singing. *The Journal of the Acoustical Society of America*, 102(1), 616-621.
28. Ohishi, Y., Kameoka, H., Mochihashi, D., & Kashino, K. (2012, September). A Stochastic Model of Singing Voice F0 Contours for Characterizing Expressive Dynamic Components. *INTERSPEECH*.
29. Saitou, T., Unoki, M., & Akagi, M. (2004). Development of the F0 Control Model for Singing-Voices Synthesis. *Speech Prosody 2004, International Conference*.
30. Lee, S. W., Ang, S. T., Dong, M., & Li, H. (2012, March). Generalized F0 modelling with absolute and relative pitch features for singing voice synthesis. *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* (pp. 429-432).
31. http://sekozer.com/test.php
32. http://www.musicxml.com/
33. http://sourceforge.net/p/rnnl/wiki/Home/
34. https://github.com/meierue/RNNLIB/
35. http://sekozer.com/rnn/rnn.html