國立臺灣大學電機資訊學院電機工程學系

碩士論文 (初稿)

Department of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis (DRAFT)

利用結構性支撐向量機的具音樂表現能力之半自動電腦演奏系統

A Semi-automatic Computer Expressive Music Performance System Using Structural Support Vector Machine

呂 行

Shing Hermes Lyu

指導教授：鄭士康博士

Advisor: Shyh-Kang Jeng, Ph.D.

中華民國 103 年 6 月

June, 2014

國立臺灣大學
電機工程學系

碩士論文
(初稿)

利用結構性支撐向量機的具音樂表現能力之半
自動電腦演奏系統

呂
行
撰

103
6

國立臺灣大學（碩）博士學位論文
口試委員會審定書

論文中文題目
論文英文題目

本論文係呂行君（R01921032）在國立臺灣大學電機工程學研究所完成之碩士學位論文，於民國 103 年○○月○○日承下列考試委員審查通過及口試及格，特此證明

口試委員：

______________________________（簽名）

（指導教授）

____________________　　____________________

____________________　　____________________

____________________　　____________________

____________________　　____________________

系主任、所長 ______________________________（簽名）

（是否須簽章依各院系所規定）

致謝

中文摘要

電腦合成的音樂一向被認為是僵硬、機械化而且沒有音樂表現能力。因此能夠產生具有表現能力的電腦自動演奏系統將會對音樂產業、個人化娛樂以及表驗藝術領域有重大的影響。在這篇論文中，我們藉由隱藏式馬可夫模型結構的結構性支撐向量機 (SVM-HMM) 來設計一個可以產生產生具有表現能力音樂的電腦自動演奏系統。我們邀請六位研究生錄製了克萊門蒂的小奏鳴曲集 Op.36。我們手動將這些錄音分割成樂句，並且利用程式從中抽取出音樂特徵。這些音樂特徵藉由 SVM-HMM 訓練成數學模型後，可以利用這個數學模型來演奏訓練過程中沒有見過的樂譜（需要手動標注分句）。此系統目前只能支援單音旋律。問卷調查的結果顯示，對於業餘或專業的音樂家來說，本系統產生的音樂尚不能達到真人的演奏水準，但是沒有音樂背景的受試者已經無法分辨本系統產生的音樂已經與真人演奏。

關鍵字：電腦自動演奏、結構性支撐向量機、支撐向量機

Abstract

Computer generated music is known to be robotic and inexpressive. A computer system that can generate expressive performance can potentially have significant impact on music production, personalized entertainment or even art. In this paper, we have designed and implemented a system that can generate expressive performance using structural support vector machine with hidden Markov model output (SVM-HMM). We recorded six sets of Muzio Clementi's Sonatina Op.36 performed by six graduate students. The recordings and scores are manually split into phrases. Their musical features are automatically extracted. Using the SVM-HMM algorithm, a mathematical model of performance knowledge is learned from these features. The trained model can generate expressive performances for previously unseen scores (with user-assigned phrasing). The system currently supports monophonic music only. Subjective test shows that for amateur and professional musician, the generated performance still need improvements to be comparable to human recording, but the generated performance received nearly the same rating as human recordings from participants without music background.

Key words: Computer Expressive Performance, Performance Rendering, Structural SVMs, Support Vector Machines.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

From the mechanical music performing automata from middle ages, to the latest Japanese virtual signer Hatune Miku, there had been many attempts to create automated system that performs music. However, many of these systems can only generate predefined expression. State-of-the-art text-to-speech system can already generate fluid and natural speech, but computer performance still can't perform very expressively. Therefore, many researcher have devoted their effort to develop systems that can automatically or semi-automatically perform music expressively. There is even a biannual contest for such systems called Music Performance Rendering Contest (RenCon) [1]. The RenCon roadmap suggest that by 2050, they wish that a computer performer can win the Chopin International Piano Contest.

There are many potential applications for a computer expressive performance system, many commercial music typesetting software like Finale and Sibelius already have expressive playback features built-in. For entertainment, such system can provide personalized music listening experience. For the music production industry, it can save a lot of cost on hiring musicians or licensing. Such system also opens up new opportunity in art, such as human-machine co-performance or interactive multimedia installation. In academia, researchers can use this technology to study the performance style of musicians, or restore historical music archive.

1.2 Goal and Contribution

The ultimate goal of this paper is to be able to play any music in any expressive style specified. But due to the technical and time constrains, we narrow down our goal to building a computer expressive performance system that performs monophonic music phrases by off-line supervised learning. The phrasing need to be annotated by human, so it's a semi-automatic system.

The major contribution of this paper is that we apply structural support vector machine on expressive performance problem. There exist no previous work that uses the discriminative learning power of SVM-HMM on computer expressive performance question. We also developed methods and tools to generate a expressive performance corpus.

TODO: normalization and quantization solution

1.3 Chapter Organization

In Chapter 2, we will give an overview of previous works and their varying goals, these works will be grouped by the way they learn performance knowledge, and we will discuss some additional specialities such as special instrument model or special user interaction pattern. In Chapter 3, we will first give a brief introduction to the mathematical background of SVM-HMM, and then give a top-down explanation to the proposed method. In Chapter 4, we will explain how the corpus used for training is designed and implemented. Finally, in Chapter 5, we will discuss several experiments that demostrates some design trade-offs and the outcome. We have also an appendix that presents some software tools used in this research, which may be helpful for other researchers in music and machine learning fields. REVIEW1

Chapter 2

Previous Works

2.1 Various Goals and Evaluation

The general goal of a computer expressive performance system is to generate expressive music, as opposed to the robotic and dull expression of rendered MIDI. But the definition of “expressive” is very vague and ambiguous, so each research will need to define a more precise and measurable goal. The following are the most popular goals a computer expressive performance system wants to achieve:

1. Perform music notations in a non-robotic way, regardless of the style.
2. Reproduce a human performance or a musician's style.
3. Accompany a human performer.
4. Validate a musicological theory of expressive performance.
5. Directly render computer composed music works.

Some systems try to perform music notations in a non-robotic way in a general sense, without a certain style in mind. These systems has been employed in music typesetting softwares, like Sibelius [2], to play the notation expressively. Most systems will implicitly achieve this goal.

Systems that are designed to reproduce certain human performance or style are usually designed or trained using a particular performer's recording as reference. One commercial example is the Zenph re-performance C [3], which reproduced the performance style of Rachimaninov, it can perform pieces that Rachimaninov never played in his lifetime in his style. Accompaniment systems try to render expressive music that act as an accompaniment for a human performer. The challenge is that the system must be able to track the progress of a human performance and render the accompaniment in real-time. One commercial example is Cadenz [**?**], using the technology created by Christopher Raphe [**?**]. It claims that it can help music student practice concertos with ease. Another goal is to validate musicological theories. Musicologist may propose theories on expressive music performance, some of them may want to build a generative model to validate their assumptions. These systems may focus more on the specific phenomenon that the theory tries to explain instead of generating music that is pleasant to human. Finally, some systems combines computer composition with expressive performance. These systems have a great advantage because the intention of the composition can be shared with the performance module. Other systems that performs past compositions can only guess the composer's intention by analyzing the score notation. These systems usually has their own data structure to represent music, which can contain more information than traditional music notation, but the resulting performance system is not backward compatible with past compositions.

Because of the high diversity in the goals they want to achieve, the capability of these systems also differs a lot. The capability of a expressive performance system can be broadly categorized into the following three key indicator [4]:

1. Expressive Expression Capability

2. Polyphonic Capability

3. Performance Creativity

Expressive expression capability can range from very high level structural expression (e.g. tempo contrast between sections) to note level expression (e.g. onset, loudness, duration) or even sub-note expression (e.g. loudness envelop, timbre). Most systems can generate note-level expression, but higher or lower level expressions are much rare.

Polyphonic capability indicates if the system can perform polyphonic input. Polyphonic systems are more challenging than monophonic ones because they requires synchronization between voices.

Performance creativity measures the ability of the system to create novel expression. The desired level of creativity varies from goal to goal. A system aiming to recreate human performance may want it to produce fixed expression based on the learning material, while a system that is combined with a composition system may want to create highly novel performance.

REVIEW1 Each system will design different experiment and metrics to verify their goals. Thus, the self-reported results are can't be compared. The only public contest that evaluates expressive performance systems is called RenCon (Performance Rendering Contest) [**?**]. RenCon is held biannually. Two scores (MIDI) will be given to participants one hour before the competition starts. The participants must generate the expressive MIDIs in the given time, and then the MIDIs will be played live on a Yamaha Disklavier piano. The audience and a jury cosists of professional musicians will give ratings for each performance. The performances are played in random order, so the audience and jury won't know the participant behind each performance, except semi-automatic ones.

The RenCon is divided into fully automatic and semi-automatic categories. But the degree of human intervention in the semi-automatic category varies widely between systems. So it's not very fair to compare them.

TODO: Discuss works that focus on timber only, e.g. Prof. Su's violin work

2.2 Researches Classified by Methods Used

Despite the difference between goals of different expressive performance systems, all expressive performance systems must have some strategy to learn and apply performance knowledge. There are generally two approach: rule-based or machine learning-based.

Using rules to generate expressive music is probably the earliest approach. Director Musices [5] is one of the early examples. Pop-E [6] is also a rule-based system which can generate polyphonic music, using its synchronization algorithm to synchronize voices.

Computational Music Emotion Rule System [7] tried to develop rules that express human emotions. Other systems like Hierarchical Parabola System [5] [8] [9] [10], Composer Pulse Syste [11, 12], Bach Fugue Syste [13], Trumpet Synthesis System [14, 15] and Rubato [16, 17] are also some systems that use rules to generate expressive performance. Most of the systems focus on expressive attributes like note onset, note duration and loudness, but Hermode Tuning System [18] put special emphasis on intonation. Rule-based systems are generally more computationally efficient because the mathematical model is much simple than those learned by machine learning algorithms. And rules are generally more understandable to human than complex model parameters. But some of the nuance, such as some subconscious deviation or grouping of notes, may be hard to describe by rules, so there is a emperical limit on how complex the rule-based system can be. Lack of creativity is also a problem for rule-based approach.

Another approach is to acquire performance knowledge by machine learning. Many machine learning methods have already been applied to this problem. For example, Music Interpretation System [19--21] and CaRo [22--24] both use linear regression to learn performance knowledge. But it is very unlikely that the expressive performance problem is a linear system, so Music Interpretation System try to solve it by using AND operations on linear regression results to handle non-linearity. But linear regression is still an oversimplification for such problem.

More complicated algorithms have also been applied: ANN Piano [25] and Emotional flute [26] uses artificial neural network. ESP Piano [27] and Music Plus One [28--30] uses Statistical Graphical Models such as Hidden Markov Model (HMM) and Bayesian Belief Network, but they did no use structural support vector machine to train the HMM.KCCA Piano System [31] uses kernel regression. Drumming System [32] tried different mapping models that generates drum patterns.

Evolutionary computation such as genetic programming is used in Genetic Programming Jazz Sax [33]. Other examples include the Sequential Covering Algorithm Genetic Algorith [34], Generative Performance Genetic Algorithm [35] and Multi-Agent System with Imitation [36,37]. Evolutionary computation takes long training time, and the results are unpredictable. But unpredictable also means there are more room for performance creativity, so these system can create unconventional but interesting performances.

TODO: Discuss works that focus on timber only, e.g. Prof. Su's violin work Another approach is to use case-based reasoning. SaxE [38--40] use fuzzy rules based on emotions to generate Jazz saxophone performance. Kagurame [41,42] focus on style (Baroque, Romantic, Classic etc.) instead of emotion. Ha-Hi-Hun [43] has a more ambitions goal in mind: to accept natural language instructions like "Perform piece X in the style of Y." Another series of researches done by Widmer at el., called PLCG [44--46] uses data-mining to find rules for expressive performance. It's successor Phrase-decomposition/PLCG [47] added hierarchical phrase structures support to the original PLCG system. And the latest research in the series called DISTALL [48,49] added hierarchical rules to the original one.

Most of the of the performance systems discussed above takes digitalized traditional musical notation (MusicXML etc.) or neutral audio as input. They have to figures out the expressive intention of the composer by musical analysis or assigned by the user. But another type of computer expressive performance has a great advantage over the previous ones, by combining computer composition and expressive performance, the performance module can share the performance intention directly with the composition module. Ossia [50] and pMIMACS [51] are two examples of this category. This approach provides great possibility for creativity, but they can only play their own composition, which is rather limited.

2.3 Additional Specialties

Most expressive performance systems implicitly or explicitly generates piano performance, because it's relatively easy to collect training samples for piano and piano sound is relatively easy to synthesize. Yet, some systems generates music in other instruments, such as saxophon [38--40], trumpe [14,15], flute [26] and drums [52]. These systems requires extra effort in creating instrument models in training, generation and synthesizing.

If not specified, most systems handles traditional western tonal music. However, most saxophone-based work [38--40] generates Jazz music, because saxophone is an iconic instrument in Jazz performance. And the Drumming Syste [52] generates Brazilian drumming music.

Performing polyphonic music is much more challenging than monophonic music, because it requires synchronization between voices, while allowing each voice to have their own expression at the same time. Pop- [6] use a synchronization mechanism to achieve polyphonic performance. Bach Fugue System [13] is created using the polyphonic rules in music theory about fugue, so it's inherently able to play polyphonic fugue. KCCA Piano System [31]can generate homophonic music -- an upper melody with an accompaniment -- which is common in piano music. Music Plus One [28--30] is a little bit different because it's a accompaniment system, it adapts non-expressive orchestral accompaniment track to user's performance. Other systems usually generates monophonic tracks only.

REVIEW1

Chapter 3

Proposed Method

3.1 Overview

The high-level architecture of the purposed system is shown in Fig. 3.1. The system has two phases, the upper half of the figure is the learning phase, the lower half is the performing phase. In the training phase, score and expressive performance recording pairs, split into phrases by human, are used as training examples for structural support vector machine with hidden Markov model output (SVM-HMM) algorithm to learn performance knowledge model. In the performing phase, a score will be given to the system for expressive performance. The SVM-HMM generation module will use the performance knowledge learned in the previous phase to produce expressive performance. The SVM-HMM output then go through a MIDI generator and MIDI synthesizer to produce audible performance.

All the scores and recordings are monophonic and contains only one musical phrase. The phrasing is done by human, thus the system is a semi-automatic system. The learning algorithm, namely SVM-HMM, can only perform off-line learning, so the learning phase can only work in a non-realtime scenario. The generating phase can work much faster, expressive music can be generated almost instantaneously.

There are many ways the user can control the performance style of the final output: first, the user can choose the training corpus. Theoratically, a set of samples from a single performer can generate a model which capture his/her style. Second, the phrasing of a

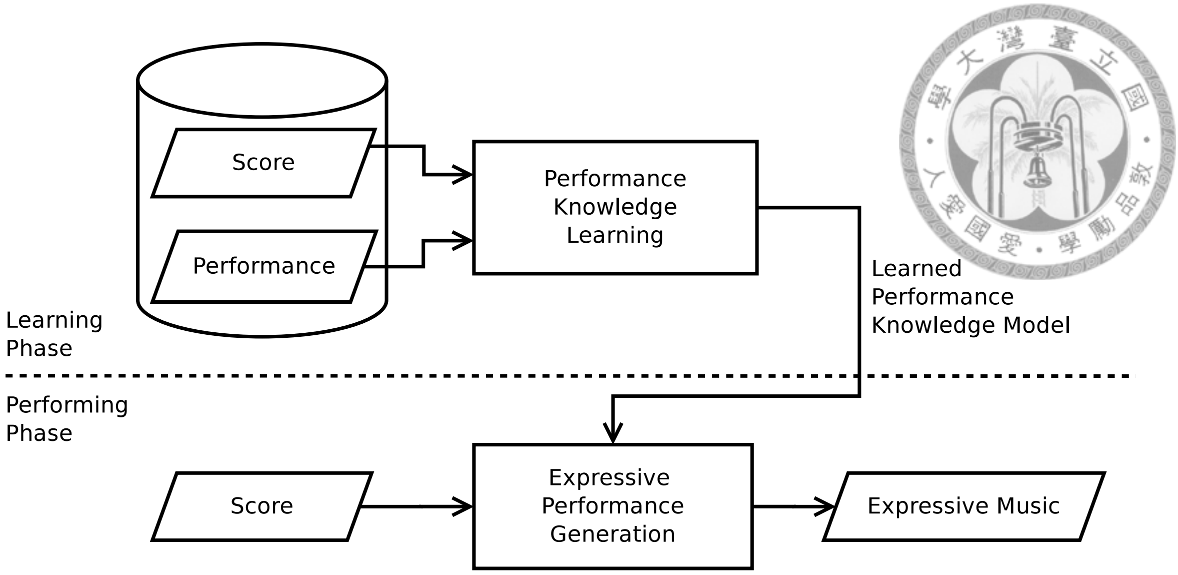
Score
Performance
Performance Knowledge Learning
Learned Performance Knowledge Model
Learning Phase
Performing Phase
Score
Expressive Performance Generation
Expressive Music

Figure 3.1: High-level system architecture

song is given by the user. Since phrasing controls the overall structural interpretation of a music piece, the user is given indirect control over the performance style.

In the following sections, we will walk through the detail steps in the learning and performing phases, and some implementation detail. The features used will be presented in the end of this chapter.

REVIEW1

3.2 A Brief Introduction to SVM-HMM

In this thesis, we use structural support vector machine to learn performance knowledge from expressive performance samples. Unlike traditional SVM algorithm, which can only produce univariate prediction, structural SVM can produce structural predictions like tree, sequence and hidden Markov model. Structural SVM with hidden Markov model output (SVM-HMM) has been successfully applied to part-of-speech problem. The part-of-speech tagging problem shares the same concept with expressive performance problem. In part-of-speech tagging, one tries to identify the role by which the word plays in the sentence, while in expressive performance, one tries to determine how a note should be played, according to it's role in the musical phrase. For example, a cadence at the end of a phrase is usually played louder and stronger than a embellishment note in the middle of a phrase. Thus, we believe SVM-HMM will be a good candidate for expressive performance. The following introduction and formulas relies heavily on [53--55].

Traditional SVM prediction problem can be described as finding a function

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

with lowest prediction error. $\mathcal{X}$ is the input features space, and $\mathcal{Y}$ is the prediction space. In traditional SVM, elements in $\mathcal{Y}$ are labels (classification) or real values (regression). But structural SVM extends the framework to generate structural output, such as tree, sequence, or hidden Markov model. To extend SVM to support structured output, the problem is modified to find a discriminant function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$, in which the input/ output pairs are mapped to a real number score. To predict an output y for an input x, one try to maximize f over all $y \in \mathcal{Y}$.

$h_{\mathbf{w}}(x) = \arg\max_{y \in \mathcal{Y}} f_{\mathbf{w}}(x, y)$

Let $f_{\mathbf{w}}$ be a linear function of the form:

$$f_{\mathbf{w}} = \mathbf{w}^T \Psi(x, y)$$

, where $\mathbf{w}$ is the parameter vector, and $\Psi(x, y)$ is the kernel function relating input x to output y. Ψ can be defined to accommodate various kind of structures.

For each structure we want to predict, a loss function that measures the accuracy of of a prediction is required. A loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow R$ need to satisfy the following property:

$$\Delta(y, y') \geq for y \neq y'$$

$$\Delta(y, y) = 0$$

The loss function is assumed to be bounded. Let's assume the input-output pair (x, y) is drawn from a join distribution P(x,y), the prediction problem is to minimize the total loss:

$$R_p^{\Delta} = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(y, f(x))dP(x, y)$$

Since we can't directly find the distribution P, we need to replace this total loss with a empirical loss, which can be calculated from the observed training set of (x_i, y_i) pairs.

$$R_s^{\Delta}(f) = \frac{1}{n}\sum_{i=1}^{n} \Delta(y_i, f(x_i))$$

With the definition of the loss function ready, we will demonstrate how to extend SVM to structural output, starting with a linear separable case, and then extend it to soft-margin formulation.

A linear separable case can be expressed by a set of linear constrains

$$\forall i \in \{1, \cdots, n\}, \forall \hat{y}_i \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \leq 0$$

However, in the SVM context, we want the solution to have the largest margin possible. So the above linear constrains will become this optimization problem:

$$\max_{\gamma, \mathbf{w}:\|\mathbf{w}\|=1} \gamma$$

$$s.t\ \forall i \in \{1, \cdots, n\}, \forall \hat{y}_i \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \leq \gamma$$

, which is equivalent to the convex quadratic programming problem:

$$\min_{\mathbf{w}, \xi_i \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$s.t.\ \forall i \in \{1, \cdots, n\}, \hat{y}_i \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \geq 1$$

To address possible non-separable problems, slack variables can be introduced to penalize prediction errors, and result in a soft-margin formalization:

$$\min_{\mathbf{w}, \xi_i \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n} \xi_i$$

$$s.t.\ \forall i \in \{1, \cdots, n\}, \hat{y}_i \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \geq 1 - \xi_i$$

C is the parameter for trade-off between low training error and large margin. The optimal C varies between different problems, so experiment should be conducted to find the optimal C for our problem.

Intuitively, a constrain violation with larger loss should be penalize more than the one with smaller loss. So [54] proposed two possible way to take the loss function into account. The first way is to re-scale the slack variable by the inverse of the loss, so a high loss leads to smaller re-scaled slack variable:

$$
\begin{aligned}
&\min_{\mathbf{w},\xi_i \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\
&s.t.\ \forall i \in \{1,\cdots,n\}, \hat{y_i} \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i,y_i) - \Psi(x_i,\hat{y_i})] \geq 1 - \frac{\xi_i}{\Delta(y_i,\hat{y_i})}
\end{aligned}
$$

The second way is to re-scale the margin, which yields

$$
\begin{aligned}
&\min_{\mathbf{w},\xi_i \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\
&s.t.\ \forall i \in \{1,\cdots,n\}, \hat{y_i} \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i,y_i) - \Psi(x_i,\hat{y_i})] \geq \Delta(y_i,\hat{y_i}) - \xi_i
\end{aligned}
$$

But the above quadratic programming problem has a extreme large number ($O(n|\mathcal{Y}|)$) of constrains , which will take considerable time to solve. [54] proposed a greedy algorithm to reduce the number of constrains. Initially, the solver starts with an empty working set with no constrains. Than the solver iteratively scans the training set to find the most violated constrains under the current solution. If a constrain is violated by more than the desired precision, the constrain is added to the working set. Then the solver re-calculate the solution under the new working set. The algorithm will terminate once no more constrain can be added under the desired precision.

In a later work by Joachims et al. [53], they created a new formulation and algorithm to further speed up the algorithm. Instead of using one slack variables each training sample, which results in a total of n slack variables, they use a single slack variable for the n training samples. The following formula is the 1-slack version of slack-rescaling structural

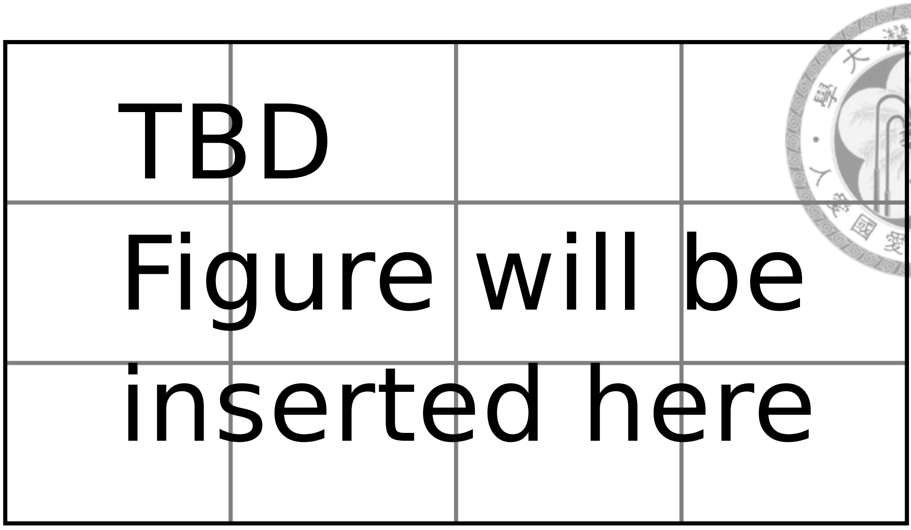

Figure 3.2: Hidden Markov Model

SVM:

$$
\begin{aligned}
&\min_{\mathbf{w},\xi_i \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\
&s.t.\ \forall i \in \{1, \cdots, n\}, \hat{y}_i \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \geq \frac{1}{n}\sum_{i=1}^{n} 1 - \frac{\xi}{\Delta(y_i, \hat{y}_i)}
\end{aligned}
$$

And margin-rescaling structural SVM:

$$
\begin{aligned}
&\min_{\mathbf{w},\xi_i \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\
&s.t.\ \forall i \in \{1, \cdots, n\}, \hat{y}_i \in \mathcal{Y} : \mathbf{w}^T[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \geq \frac{1}{n}\sum_{i=1}^{n} \Delta(y_i, \hat{y}_i) - \xi
\end{aligned}
$$

Detailed proof on how the new formulation is equivalently general as the old one is given in the paper.

With the framework described above, the only problem left is how to define the general loss function for Hidden Markov Model (HMM)? In [55], the authors proposed two types of features for a equal-length observation/label sequence pair (x, y). The first is the interaction of a observation with a label, the other is the interaction between neighboring labels.

Formally, for some observed features $\Phi_r(x^s)$ of a note x located in sth position of the phrase, and assume $[[y^t = \tau]]$ denotes the tth note is played at a velocity of τ, the interaction of the two predicate can be written as

$$\phi^{st}_{r\sigma}(\mathbf{x}, \mathbf{y}) = \left[\left[y^t = \tau\right]\right] \Psi_r(x^s),\ 1 \leq \gamma \leq d,\ \tau \in \Sigma$$

And for interaction between labels, the feature can be written as

$$\hat{\phi}^{st}_{r\sigma}(\mathbf{x}, \mathbf{y}) = \left[\left[y^s = \sigma \wedge y^t = \tau\right]\right],\ \sigma, \tau \in \Sigma$$

By selecting a dependency order for the HMM model, we can restrict s's and t's. For example, for a first-order HMM, $s = t$ for the first feature, and $s = t - 1$ for the second feature. The two features on the same time t is then stacked into a vector $\Psi(x, y; t)$. The feature map for the whole sequence is simply the sum of all the feature vectors

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \Phi(\mathbf{x}, \mathbf{y}; t)$$

Finally, the distance between two feature maps depends on the number of common label segments and the inner product between the input features sequence with common labels.

$$\langle \Phi(\mathbf{x}, \mathbf{y}), \Phi(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \rangle = \sum_{s,t} \left[\left[y^{s-1} = \hat{y}^{t-1} \wedge y^s = \hat{y}^t\right]\right] + \sum_{s,t} \left[\left[y^s = \hat{y}^t\right]\right] k(x^s, \hat{x}^t)$$

To speed up the computation of F for HMM, a Viterbi-like decoding algorithm is used.

3.3 Learning Performance Knowledge

The main goal in the learning phase is to extract performance knowledge from training samples. Fig. 3.3 shows the internal structure of the learning phase.

Training samples are matched score and expressive performance pairs (their format and preparation process is discussed in Chapter 4). The raw data from the samples are too

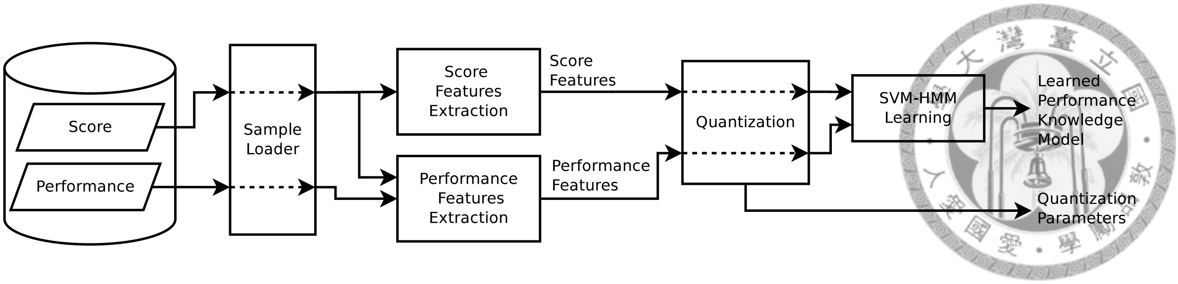

Figure 3.3: Learning phase flow chart

complex to process, so we need to extract important features from them. Two types of features will be extracted from the samples: first, the musicological cues from the scores are called score features; second, the measurable expression from the expressive performances are called the performance features. In order to learn the performance knowledge from the samples, we want to learn a prediction model using SVM-HMM. This process can be analogize to a human performer reading the explicit and implicit cues from the score, and perform the music with certain expressive expression. The definition of the features used will be presented in Section 3.5

3.3.1 Training Sample Loader

A training sample is loaded with the sample loader module, since a training sample is consisted of a score (musicXML format) and an expressive recording (MIDI format), the sample loader finds the two files given the sample name, and load them into an intermediate representation (`music21.Stream` object provided by the `music21` librar [56] from MIT). The music21 library will convert the musicXML and MIDI format into a python Object hierarchy that is easy to access and manipulate by python code.

One caveat here is that the recording in MIDI may contain very subtle expression. But the music21 library will quantize the MIDI in the time axis by default, which will destroy the subtle onset and duration expression. And the music21 library don't handle the “ticks per quarter note” information in the MIDI heade [**?**], so we must explicitly specify this value, and explicitly disable quantization during MIDI loading.

3.3.2 Features Extraction

In order to keep the system architecture simple, a feature extractor are designed to be independent to other feature extractors, so features can added or removed without affecting the rest of the system. Furthermore, this enables parallel feature extraction. But sometimes a feature inevitably depends on other features, for example, the “relative duration with the previous note” is calculated based on the “duration” feature. Since we want to avoid complex dependency management, the “relative duration” feature extractor has to calculated the duration itself, instead of depending on the “duration” extractor. Therefore, the “duration” feature extracted will be computed twice. To avoid redundant computation of the feature extractors, we implemented a caching mechanism. Once the “duration” feature had been computed, no matter it is calculated in the “duration” extractor or in the “relative dutaion” extractor, it's value will be cached during this execution session, so no matter how many feature extractors uses the “duration” feature, they can get the value directly from cache. This method can speed up the execution without handling dependencies.

The extracted features are aggregated and stored into a JavaScript Object Notation (JSON) file for the SVM-HMM module to load. By saving the features in a human-readable intermediate file, we can easily look into the file to debug potential problems.

3.3.3 SVM-HMM Learning

After all features are extracted, the next step is to learn performance knowledge from the features. In the early stage of this research, we have successfully applied linear regressio [57]. However, assuming this problem to be a linear system is clearly an oversimplification. So we switch to structural support vector machine with hidden Markov model output (SVM-HMM) [53--55] as our supervised learning algorithm.

The SVM-HMM learning module loads the feature file from the previous stage, and rearrange the features to fit the required input format of the SVM-HMM learner program. However, most features from the previous stage are real values, but SVM-HMM only takes discrete performance features[1], so quantization is required. For each performance feature, a quantizer calculates the overall mean and standard deviation from all training samples. There are many possible way to quantize the features, each will result in different output, here we will present a quantizer design for demonstration purpose: a uniform quantizer is employed to quantize a performance feature into 128 intervals/bins. The range between mean minus or plus four standard deviations into 128 uniform intervals. Values over than mean plus four standard deviations are quantized into the 128th bin, and values below mean minus four standard deviations are quantized into the 1st bin. The number of intervals decides how fine-grain the quantization will be, if the number is too low, the quantization error will be too large, expressions across a large range will be quantized into the same bin, and results in dull expression. However, if the number is too large, There will be too few samples for each interval, and the training process will take a lot of CPU and memory resources without significant gain in prediction accuracy. The range of four standard deviation is chosen by trail and error, a narrower range will make most of the extreme values be quantized into the largest of smallest bin, so the performance will have a lot of saturated values. But a very large range will make the interval between each quantization bin too large, rising the quantization error.

The theoretical background of SVM-HMM is already mentioned in Section 3.2. We leverage Thorsten Joachims's implementation called SVM^{hmm} [58]. SVM^{hmm} is an implementation of structural SVMs for sequence tagging [55] using the training algorithm described in [54] and [53]. The SVM^{hmm} package contains a SVM-HMM training program called `svm_hmm_learn` and a prediction program called `svm_hmm_classify`, which will be used in the performing phase. For structural simplicity, we train one independent model for each performance feature, each model uses all the score features to try to predict a single quantized performance feature. The `svm_hmm_learn` takes a training file describing those features. Each line represents features for a note, organized in the following format:

```
        PERF qid:EXNUM FEAT1:FEAT1_VAL FEAT2:FEAT2_VAL ... #comment
```

[1] SVM-HMM is initially designed for tasks like part-of-speech tagging, in which real value or binary featrues are used to predict discrete part-of-speech tags.

`PERF` is a quantized performance feature. The `EXNUM` after `qid:` identifies the phrases, all notes in a phrase will have the same `qid:EXNUM` identifier. Following the identifier are quantized score features, denote as `feature name : feature value`, separated by spaces. And anything after a # symbol is comment.

TODO: partial model

There are some key parameters needed to be specified for the training program. First the C parameter in SVM, which controls the trade-off between lowering training error and maximizing margin. Larger C will result in lower training error, but the margin may be smaller. Second, the ε parameter controls the required precision for termination. The smaller the ε, the precision should be higher, but may require more time and memory resource to compute. Finally, for the HMM part, we need to specify the order of dependencies of transition states and emission states. In our case, transition dependency is set to one, which stands for first-order Markov property, and emission dependency is set to zero. Since we train separate models for each performance feature, each model can have their own set of parameters. The parameter selection process is done by experiment, which will be presented in Chapter 5

Finally, the training program will output three model files (because we use three performance features). In the model file are the SVM-HMM model parameters, such as the support vectors and other metadata. Since it takes considerable time (roughly a dozen minutes or to a few hours) to train a model (depending on the amount of training samples and the power of the computer running the system), the system can only support off-line learning. But the learning process only need to be run once. The performance knowledge model can be reused over and over again in the performing phase.

3.4 Performing Expressively

The performing phase uses the performance knowledge model learned in the previous phase to generate expressive performances. The input is a score file, which should not be used during training to prevent overfitting. Score features will be extracted from it using the same code as in the learning phase. The SVM-HMM generation module will use the

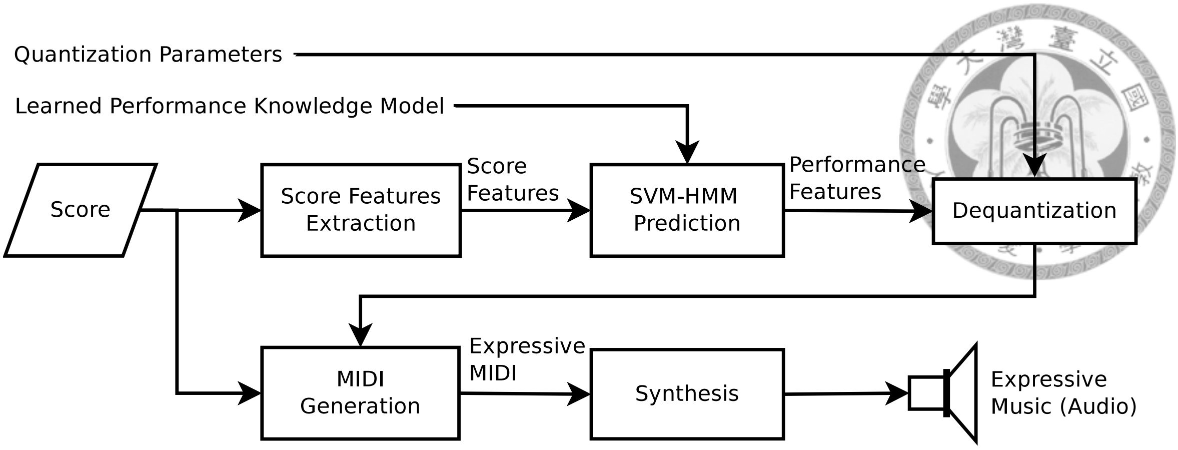

Figure 3.4: Performing phase flow chart

learned model and the score features to predict the performance features. These features will than be de-quantized back to real values using the method described previously. An MIDI generation module will apply those performance features onto the score to produce a expressive MIDI file. The MIDI file itself is already a expressive performance, in order to listen to the sound, an software synthesizer can be used to render the MIDI file into WAV or MP3 format.

3.4.1 SVM-HMM Generation

The feature extraction and aggregation process in the performing phase is similar to the learning phase, but the `PERF` fields in the SVM-HMM input file are all set to zero, meaning that we don't know its value and wish the algorithm to predict it. The `svm_hmm_classify` program will take these inputs with the learned model file and predict the quantized labels of the performance features. These performance features are de-quantized back to real values. When reconstructing the features from the quantized value during the performing phase, the middle point of each bin is used. For the 128th bin, the mean plus four standard deviation is used, and similarly for the 1st bin, the mean minus four standard deviation is used.

3.4.2 MIDI Generation

The performance features are than applied onto the input score. The onset timings will be shifted, the duration extended or shortened, and the loudness shifted according to the predicted performance features. The resulting expressive performance will be transfromed into MIDI files using `music21` library.

TODO:Dramatization/post processing

3.4.3 Audio Synthesis

In order to actually hear the expressive performance, the MIDI file can be rendered by a software MIDI synthesizer. Such as `timidity++` software synthesizer for Linux. The output will be an WAV (Waveform Audio Format) file, which can be compressed into MP3 (MPEG-2 Audio Layer III) by `lame` audio encoder. Alternatively, one can use hardware synthesizers, for example, RenCo [1] contest uses Yamaha Disklavier Digital Piano to render contestants' submission.

Because sub-note-level expression is not the primary goal of this research, we choose standard MIDI grand piano sound to render the music. The system can be extended to used more advanced physical model or musical instrument specific audio synthesizer. Sub-note level features, such as special techniques for violins, can be added to the features list and be learned by the SVM-HMM model.

TODO: phrase concatenation REVIEW1

3.5 Features

As mentioned in Section 3.3, there are two types of features, score features and performance features. We will present the features used in the system, and discuss the difficulties encountered.

3.5.1 Score Features

Score features are musicological cues presented in the score. The purpose of score features are to simulate the high level information a performer may perceive when he/she reads the score. The basic time unit for the features are notes. Each note will have one of each features presented below. Score features includes:

Relative position in a phrase: The relative position of a note in the phrase, its value ranges from 0% to 100%. This feature is intended to capture the special expression in the start or the end of a phrase, or to capture time-variant expression like arch-type loudness variation.

Relative pitch: The pitch of a note relative to the pitch range of the phrase, denoted by MIDI pitch number (resolution is down to semitone). For a phrase of n notes with pitch $P_1, P_2, \ldots, P_n$,

$$RP = \frac{P_i - min(P_1, P_2, \ldots, P_n)}{max(P_1, P_2, \ldots, P_n) - min(P_1, P_2, \ldots, P_n)}$$

Where P_i is the pitch of note at position t.

Interval from the previous note: The interval between the current note and its previous note (in semitone). This represents the direction of the melodic line.

$$IP = P_i - P_{i-1}$$

See Fig. 3.5 for example.

Interval to the next note: The interval between the current note and its previous note (in semitone).

$$IN = P_{i+1} - P_i$$

See Fig. 3.5 for example.

Note duration: The duration of a note (quarter notes).

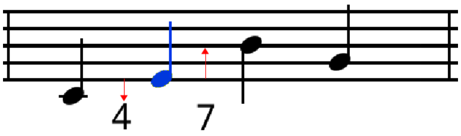

Figure 3.5: Interval from/to neighbor notes

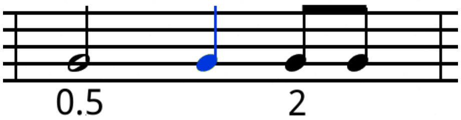

Figure 3.6: Relative duration with the previous/next note

Grace notes have zero duration in musicXML specification. The reason for this is that grace notes are considered very short ornaments that does not occupy real beat position. But zero duration is hard to handle in math formulation. So we assigned a duration of $\frac{1}{64} = 0.0625$ quarter note to all grace notes, which is equivalent to the length of a sixty-fourth note. Sixty-fourth note is chosen because it's far shorter than all the notes in our corpus.

Relative Duration with the previous note: The duration of a note divided by the duration of its previous note. For a phrase of n notes with duration $D_1, D_2, \ldots, D_n$,

$$RDP = \frac{D_i}{D_{i-1}}$$

See Fig. 3.6 for example. This feature is intended to locate local change in tempo, such as a series of rapid consecutive notes followed by a long note, which will cause a discontinuity in this feature.

Relative duration with the next note: The duration of a note divided by duration of its next note.

$$RDN = \frac{D_i}{D_{i+1}}$$

See Fig. 3.6 for example.

Metric position: The position (beat) of a note in a measure. For example, a $\frac{4}{4}$ time signa-

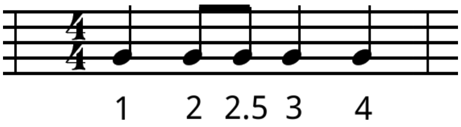

Figure 3.7: Metric position

ture will have a beat unit of a quarter note. So if the measure consists of four quarter notes, each of them will have metric position of 1, 2, 3 and 4. Please refer to Fig. 3.7 for example.

Metric position usually implies beat strength. In most tonal music, there exist a hierarchy of beat strength. For example, in a measure of a $\frac{4}{4}$ piece, the first note is usually the strongest, the third note is the second strongest, and the second and fourth notes are the least strong.

3.5.2 Performance Features

Performance features are the expressions we would like to learn from a performance. Performance features are extracted by comparing the expressive performance with the score. Performance features includes:

Onset time deviation: The onset time of a natural human recording will not be exactly on the beat, this phenomenon is roughly corresponding to the music term “rubato.” The onset time deviation is the difference of onset timing between the performance and the score. Namely,

$$ROB = O_i^{perf} - O_i^{score}$$

Where O_i^{perf} is the onset time of note i in the performance, O_i^{score} is the onset time of note i in the score.

However, the above formula assumes the performance is played at the exact same tempo as assigned in the score. In the corpus we use, test subject can't always keep up with the speed of the score because of limited piano skill, or they may speed up or slow down certain sections as their expression. Therefore, the performance should

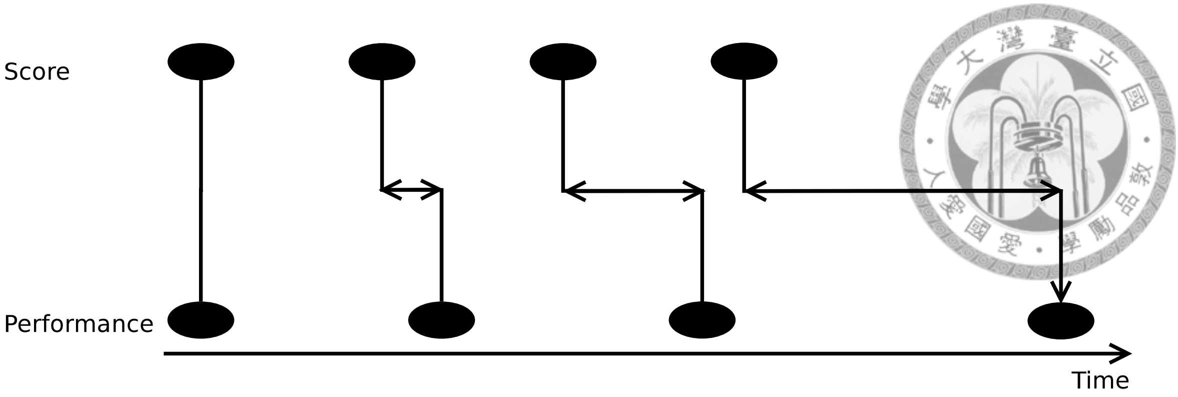

Figure 3.8: Systematic bias in onset deviation

be linearly scaled to avoid systematic bias, We will discuss this issue in Section 3.5.3.

loudness: The loudness of a note. Measured by MIDI velocity level 0 through 127.

$$RL = \frac{L_i}{max(L_1, L_2, \ldots, L_n)}$$

Relative duration: The performed duration of a note divided by the nominal duration on the score.

$$RD = \frac{D_i^{perf}}{D_i^{score}}$$

3.5.3 Normalizing Onset Deviation

In th previous section, we mentioned that the onset deviation feature will have problem when performance is not played at the exact temp indicated on the score. As illustrated in Fig. 3.8, if the performance is played faster than expected, the deviation will grow larger and larger over time. The systematic bias caused by the difference in total duration will mix up with the local note deviation, For a long phrase, the onset deviation of the last notes can larger than a dozen quarter notes. These kind of extreme values will cause erroneous predictions in the model: a note may be delayed for a very large deviation, causing it to be played after the next note, the swapped notes will destroy the melody.

In other words, the original definition of the onset deviation actually contains two type of deviation: a global/systematic deviation cause by the difference between performed and nominal tempo, and a local deviation cause by note-level expression. Since the intention of the onset deviation features is to catch the note-level expression, the performance must be linearly scaled to cancel the global deviation.

Initially, we tried two possible type of normalization methods :

1. Align the onset of the first notes, align the onset of the last notes

2. Align the onset of the first notes, align the end of the last notes

We proposed an automated approach to find the best scaling ratio to address this problem. First we have to define the distance between scaled phrases as our target to minimize. If we represent a phrase as a vector of all the onset timings, the l^2-norm of the tow vectors can be treated as the distance. Note that the two vectors must have the same size, because the recordings are required to match note-to-note with the score. Using this distance measure, we would like to find an optimal scaling ratio such that the scaled recording has the minimum distance from the score. Brent's Metho [59] is used to find the optimal ratio. To speed up the optimization and prevent unreasonable value, a search range of $[initial\ guess \times 0.5, initial\ guess \times 2]$ is imposed on the optimizer. The $initial\ guess$ is used as a rough estimate of the ratio, calculated by aligning the first and last onset of the phrase. Than we assume the actual ratio will not be smaller than half of $initial\ guess$ and not larger than twice of $initial\ guess$. The two numbers 0.5 and 2 are chosen by trail and error, but most of the empirical data suggest is valid most of the time. We will demonstrate the effectiveness of this solution in Section 5.1

Chapter 4

Corpus Preparation

An expressive performance corpus is a set of performance samples. Since this research is based on a supervised learning algorithm, a high-quality corpus is essential to our success. Each sample consists of a score and its corresponding human recording. Some metadata such as structure analysis, harmonic analysis etc. may also be included. In this chapter, we will review some the existing corpora, specifications and formats of our corpus, and how we actually construct it.

4.1 Existing Corpora

Unlike other research fields like speech processing or natural language processing, there exist virtually no public accessible corpus for computer expressive performance. CrestMusePED [60] (PEDB stands for “Performance Expression Database”) is a corpus created by Japan Science and Technology Agency's CREST program, however, until the time of this writing, we can't establish any contact with the database administrators. They use a graphical interface to annotate the expressive performance parameters from audio recordings.The repertoire covers many piano works from well-known classical composers like Bach, Mozart, and Chopin, and the recordings are from famous pianists. From their websit [60]they claim to contain the following data: PEDB-SCR - score text information, PEDB-DEV - performance deviation data and PEDB-IDX - audio performance credit. But the quality of the data is unknown.

Another example is the Magaloff Projec [61], which is a joint effort of some universities in Austria. They invite Russian pianist Nikita Magaloff to record all solo works for piano by Frederic Chopin on a Bösendorfer SE computer-controlled grand piano. This corpus became the material for many subsequent researches [62--68]. Flossmann et al., one of the leading researchers of the project, also won the 2008 RenCon contest with a expressive performance system call YQ [69] based on this corpus. However, the corpus is not opened up to the public.

Since both corpora are not available, we need to implement our own one. We will start by defining the specification.

4.2 Corpus Specification

The corpus we need must fulfill the following constrains:

1. All the samples are monophonic, containing only a single melody without chords.
2. No human error, such as insertion, deletion, or wrong pitch exist in the recording; the score and recording are matched note-to-note.
3. Phrasing is annotated by human.
4. The score, recording and phrasing data are in machine-readable format.

Some useful information are excluded because they are less relevant to our system. Examples are:

1. Advanced Structural Analysis, such as GTTM (Generative Theory of Tonal Music) [70]
2. Harmonic Analysis
3. Instrument specific techniques, such as violin pizzicato, tapping, or bow techniques.
4. Piano paddle usage

Table 4.1: Clementi's Sonatinas Op.36

Title	Movement	Time Signature
No.1 Sonatina in C major	I. Allegro	4/4
	II. Andante	3/4
	III. Vivace	3/8
No.2 Sonatina in G major	I. Allegretto	2/4
	II. Allegretto	3/4
	III. Allegro	3/8
No.3 Sonatina in C major	I. Spiritoso	4/4
	II. Un poco adagio	2/2
	III. Allegro	2/4
No.4 Sonatina in F major	I. Con spirito	3/4
	II. Andante con espressione	2/4
	III. Rondó: Allegro vivace	2/4
No.5 Sonatina in G major	I. Presto	2/2
	II. Allegretto moderato	3/8
	III. Rondó: Allegro molto	2/4
No.6 Sonatina in D major	I. Allegro con spirito	4/4
	II. Allegretto	6/8

5. Other instrument specific instructions, such as piano fingering, violin bow techniques etc.

We choose Clementi's Sonatina Op.36 for our corpus, because it is a must-learn repertoire for piano students, so it's easy to find a wide skill range of performers to record the corpus. These sonatinas are in classical style, so the learned model can be easily extended to other classical era works like Mozart and Haydn. There are six sonatinas included in Op.36, the first five have three movements each, and the last one has two movements. The titles, tempo markers and time signatures of all the pieces are listed in Table 4.1

To represent Clemetni's work in digital format, we choose MusicXML. MusicXML is a digital score notation using XML (eXtensible Markup Language) , it can express most traditional music notations and metadata. Most music notation software and software tool supports musicXML format. Although MIDI is also a popular candidate for score representation in computer music research, it is designed to hold instrument control signal rather than notation. Some music symbols may not be available in MIDI. Furthermore, MIDI represents music as a series of note-on and note-off events, which requires additional effort to transform into traditional notation.

But for performance, MIDI is the most suitable format. Using a pressure sensitive digital piano, pianist can record in a natural way. The recordings have high precision in time and pitch, and polyphonic tracks can easily be separated. Although WAV (Waveform Audio Format) audio recording has higher fidelity than MIDI, but they are harder to parse by computers. Without robust onset detection, pitch detection, and source separation, the information is extremely difficult to extract. Manually annotate each WAV recording takes a lot of manpower, and the accuracy may not be consistent.

There's a way to keep both the score and the recording in one single MIDI file. Instead of recording the actual note-on and note-off timing, we keep the nominal note-on and note-off the same as in score. Then, MIDI tempo-change events are inserted before each note to shift the actual timing of the recorded notes. But since MIDI is so limited to represent a score, and it requires complex calculations to recover the performance from fixed notes and tempo-change events, this method is not used in the research.

Finally, we store the phrasing, which is the only metadata we used, in a plaintext file, each line in the phrasing file is the starting point of each phrase. The starting point is defined as the onset timing (in quarter notes) from the beginning of the piece[1] The phrasing is assigned by the us using the following principles:,

1. Phrase may be separated by a salient pause.

2. Phrase may end with a cadence.

3. Phrase may be separated by dramatic change in tempo, key or loudness

4. Repeated structures in tempo or pitch may be a phrase.

An automatic phrasing algorithm may be achieved in the future, either by fuzzy rules or by machine learning. With the automatic phrasing capability, the system can become fully automatic. But phrasing controls the structural expression of a piece, we left the phrasing decision to the user. Because we intend to leave some degree of freedom for

[1] For a phrase that start at a point which is a circulating decimal, for example $\frac{1}{3}$, the starting point can be alternatively defined as any finite decimal between the end of the last phrase and the start of the current phrase. For example, if the last phrase stops at beat 1, the second phrase start at $2\overset{1/3}{=}2.333\cdots$ beat, the start point of the second phrase can be written as 2.3 or 2.0, etc.

users to expressive themselves. But note-level expression is too trivial to be assigned by hand, but deciding phrase-level expression is less demanding for an ordinary user.

4.3 Implementation

4.3.1 Score Preparation

The digital scores are downloaded from KernScore website [71]. The scores are transformed into MusicXML from the original Hundrum file format (.krn) using the music21 toolki [56]. Because this research focus on monophonic melody only, the accompaniments are remove and the chords are reduced to their highest-pitched note, which is usually the most salient melody. The reduced scores are doubled-checked against a printed version publish by Durand & Cie., Paris [72] to eliminate all errors.

4.3.2 MIDI Recording

We have implemented two methods for recording: First, using a Yamaha digital piano to record MIDI. Second, by tapping on a touch sensitive device to express tempo, duration and loudness. Due to accuracy consideration, only the recordings from Yamaha digital piano are selected.

We used a Yamaha P80 88-key graded hammer effect[2]digital piano for recording. Using a MIDI-to-USB converter, the keyboard was connected to Rosegarden Digital Audio Workstation (DAW) software on a Linux computer. The Rosegarden DAW also generated the metronome sound to help the performer maintain a steady speed. Metronome is mandatory because if the performer plays freely, the tempo information written in the MIDI file will be invalid, which makes subsequent parsing and linear scaling very difficult. So the performers were asked to follow the speed of the metronome, but they can apply any level of rubato they like.

The second method, which is not used in the final experiments, is to utilize touch-

[2] Graded Hammer Effect feature provides realistic key pressure response similar to a traditional acoustic piano.

enabled input device like smartphone touchscreen or laptop touchpad. We have implemented an prototype using a Synaptics Touchpad on a Lenovo ThinkPad X200i laptop. When the user taps the touchpad, one note from a predefined score will be played, the duration and loudness will be controlled by the time and pressure of the tap. So the user can "play" the touchpad like a musical instrument. This idea has already be used in musical games like Magic Piano [**?**]. This method requires minimal instrument skill and utilize widely available hardware. But the accuracy in pressure is not satisfying, because most touchpad use contact area to estimate pressure. But it is indeed a low cost alternative toa MIDI digital piano.

4.3.3 MIDI Cleaning and Phrase Splitting

After the MIDIs are recorded, we use scripts to check if each recording is matched note-to-note with its corresponding score. If not, the mistakes are manually corrected. If there are a small segments that is totally messed up, it will be reconstruct using repeated or similar segments from the same piece. The matched score and MIDI pair are then split into phrases according to the phrasing file. The split phrases are checked again for note-to-note match before they are used.

4.4 Results

Six graduate students (not majored in music) with varying piano skill are invited to record the samples. The number of mistakes they made are listed in Table 4.2.[3] These mistakes are identified using the unix `diff` [**?**] tool. Five of them (A to E) finished Clementi's entire Op.36, performer F only recorded part of the work. The total number of recordings and the corresponding phrases/notes counts are shown in Table 4.3. TODO:mistakes

The number of phrases (according to our phrasing) and notes are shown in Table 4.4. Fig. 4.1 shows the length distribution of each movement, most movements are around a few hundred notes, except the long No. 6 and some short second movements. Fig.

[3] The performers are allowed to re-record as many time as they want, so the actual number of mistakes may be higher.

Table 4.2: Number of Mistakes in Corpus

Performer	1-1	1-2	1-3	2-1	2-2	2-3	3-1	3-2	3-3	4-1	4-2	4-3	5-1	5-2	5-3	6-1	6-2	Subtotal
A	0	5	2	4	3	0	4	2	2	4	5	9	9	2	3	4	1	59
B	2	1	1	2	2	1	6	0	3	2	3	6	12	3	3	10	7	64
C	1	1	0	1	0	1	2	0	0	3	2	3	10	1	35	6	1	67
D	0	1	1	2	3	1	4	1	1	10	6	3	10	2	7	13	2	67
E	2	3	4	4	0	3	4	0	0	21	6	22	23	3	9	18	13	135
F	1	3	2	11	6	8	7	2	6		15				20			81
Subtotal	6	14	10	24	14	14	27	5	12	40	37	43	64	11	77	51	24	473

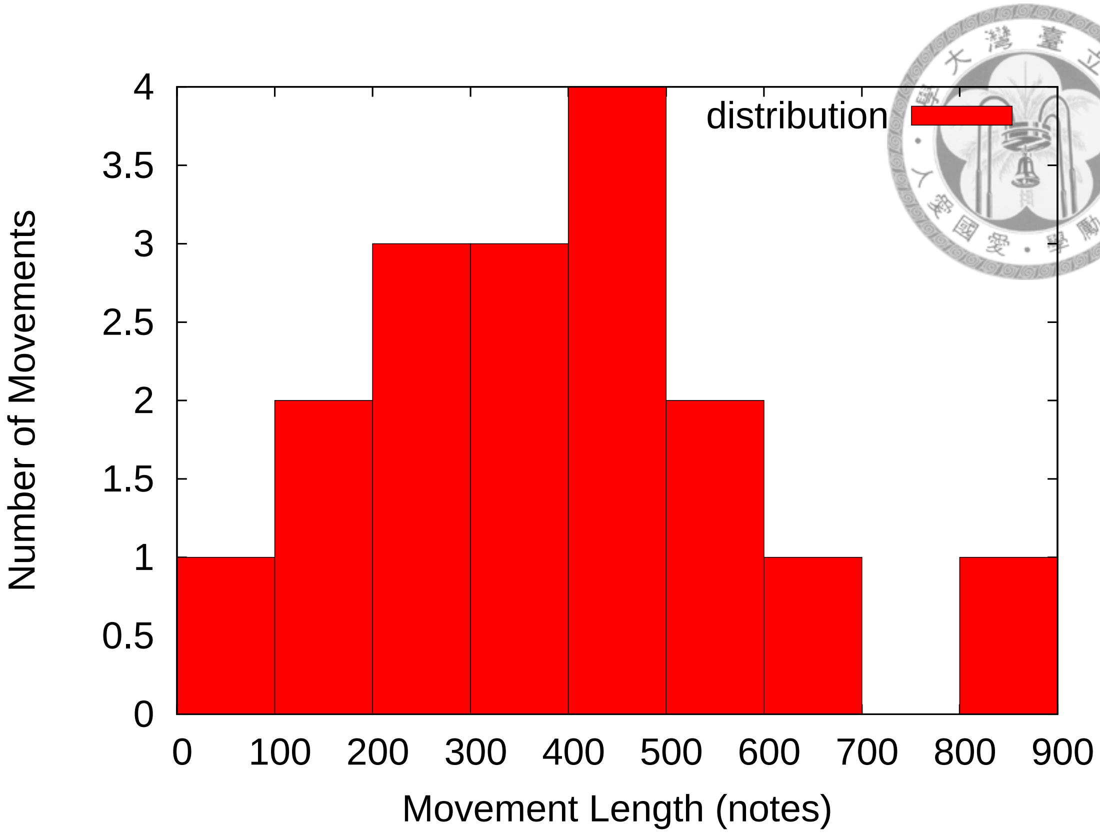

Figure 4.1: Movement length (notes) distribution

4.2 shows the length distribution in numbers of phrases, most movements are around 20 phrases. The length distribution of the phrases in all six pieces are shown in Fig. 4.3, most phrases are shorter than 30 notes. Some very long phrases exists, they are often virtuoso segments with very fast note sequences, so they are hard to be further split.

Table 4.3: Total Recorded Phrases and Notes Count

Title	Recordings Count	Total Phrases	Total Notes
No.1 Mov. I	6	72	1332
No.1 Mov. II	6	60	882
No.1 Mov. III	6	102	1566
No.2 Mov. I	6	108	1920
No.2 Mov. II	6	36	750
No.2 Mov. III	6	168	2484
No.3 Mov. I	6	156	3156
No.3 Mov. II	6	42	444
No.3 Mov. III	6	120	2628
No.4 Mov. I	5	80	2325
No.4 Mov. II	6	78	1332
No.4 Mov. III	5	85	1920
No.5 Mov. I	5	85	3360
No.5 Mov. II	5	70	1580
No.5 Mov. III	6	144	3384
No.6 Mov. I	5	145	4180
No.6 Mov. II	6	78	2754
Total	97	1629	35997

Table 4.4: Phrases and Notes Count for Clementi's Sonatina Op.36

Title	Phrases Count	Notes Count
No.1 Mov. I	12	222
No.1 Mov. II	10	147
No.1 Mov. III	16	261
No.2 Mov. I	18	320
No.2 Mov. II	6	125
No.2 Mov. III	28	414
No.3 Mov. I	25	526
No.3 Mov. II	6	74
No.3 Mov. III	19	438
No.4 Mov. I	25	465
No.4 Mov. II	12	222
No.4 Mov. III	16	384
No.5 Mov. I	17	672
No.5 Mov. II	13	316
No.5 Mov. III	24	564
No.6 Mov. I	28	836
No.6 Mov. II	11	459
Total	286	6445

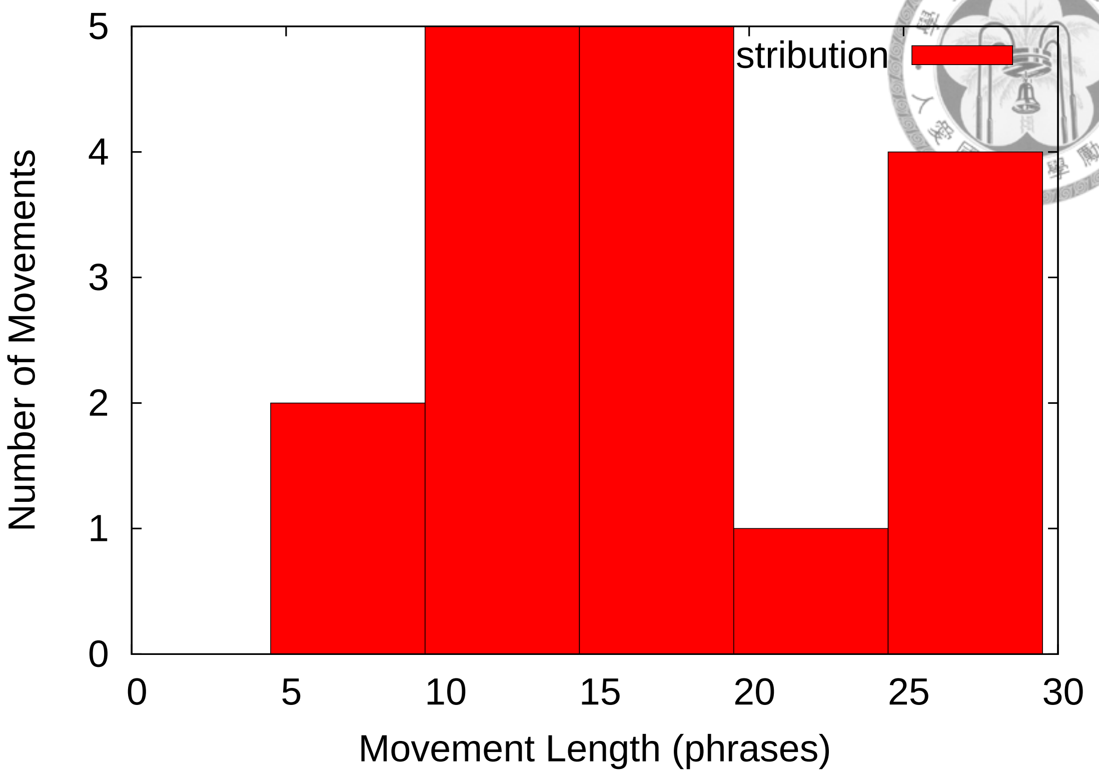

Figure 4.2: Movement length (phrases) distribution

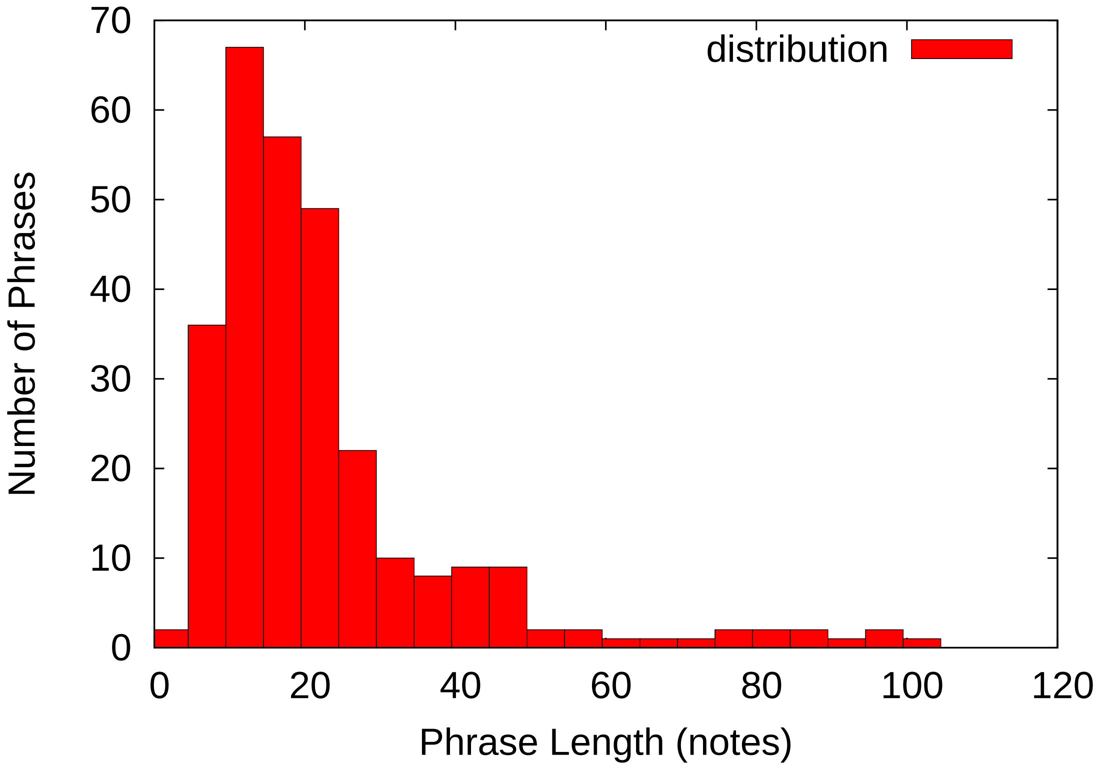

Figure 4.3: Phrase length (notes) distribution

Chapter 5

Experiments, Results and Discussions

In this chapter, we will show some experiment results to demonstrate the effectiveness of our method. Section 5.1 deals with the onset deviation problem highlighted in Section 3.5.3. Section 5.2 discusses how the various parameters in our system are chosen. Section 5.3 describes a subjective test to see if audience can or cannot identify the difference between generated and human performances.

5.1 Onset Deviation Normalization

TODO:onset deviation problem review As mentioned in Section 3.5.3, a bad normalization method will usually result in unreasonable high onset deviation. To overcome this challenge, we proposed a automatic way to select the normalization parameter. In this section, we will evaluate the effectiveness of the method.

We extract the onset deviation feature from performer E's recording[1], using the two types of fixed normalization method and the automatic normalization method mentioned in Section 3.5.3. The extracted onset deviations are shown in Fig. 5.1, Fig. 5.2 and Fig.5.3. Each dotted line from left to right represents a phrase in the corpus. Each dot represents the onset deviation value of a note. The notes are spread uniformly on the horizontal axis, which only shows the order of appearance, not the real time scale. First, we can see

[1] The effect of this method is less obvious for performer with better piano skill, because they have better control over tempo stability.

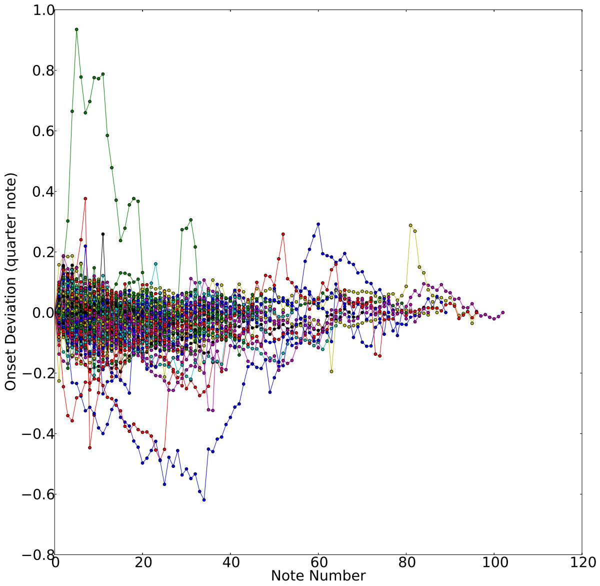

Figure 5.1: Onset deviations by aligning last note onsets

in Fig. 5.2 that aligning the note-off of the last notes results in very large deviation in some phrases. This is because performers tend to extend the last note in certain phrases to emphasize the end of a phrase or section. This kind of extension will cause the last notes onset in the performance far apart from the score. Fig. 5.3 and Fig. 5.1 looks much similar, but the onset deviation values in Fig. 5.1 is more dramatic than those in Fig. 5.3, which proofs that the automatic normalization method can indeed reduce the overall onset deviation. Another benefit of the automatic normalization over aligning last notes onsets is that the last note onset is not force aligned, which allows more space for free expression for the last note. This effect can be seen in Fig. 5.1, in which the right-most end of a line, i.e. the last note, always goes back to zero, while in Fig. 5.3, the end of a line can end in a wide range of values.

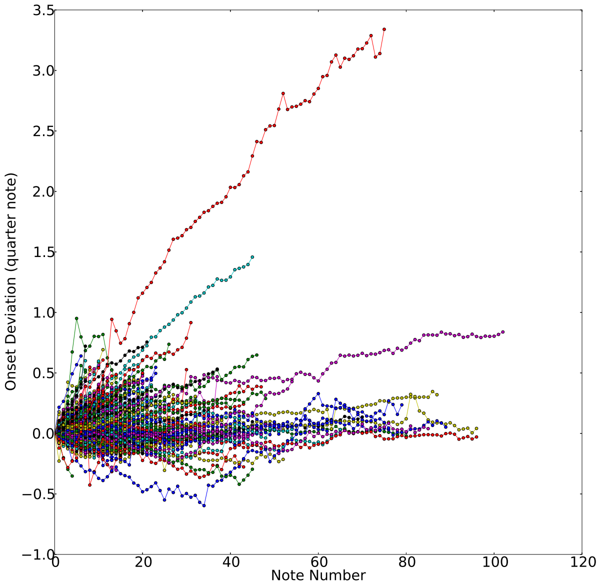

Figure 5.2: Onset deviations by aligning last notes note-off

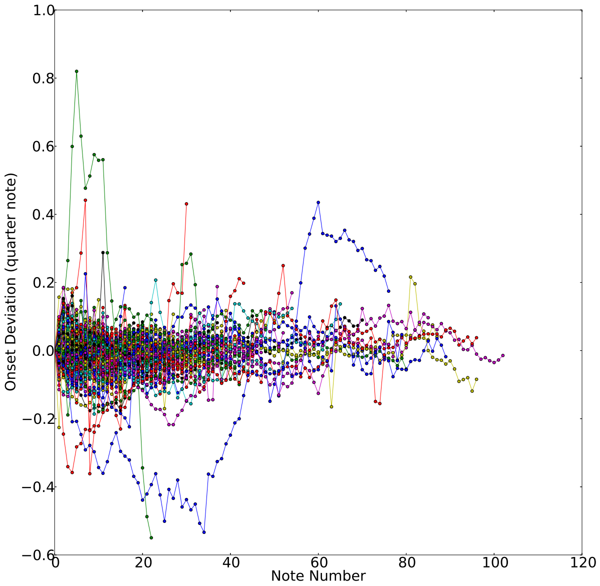

Figure 5.3: Onset deviations using automated normalization method

5.2 Parameter Selection

5.2.1 SVM-HMM-related Parameters

Since SVM-HMM is a combination of SVM and HMM, there are many parameters which need adjustment from both model. Two parameters are tested in this experiment to find the optimal value: the termination accuracy ε and the misclassification penalty factor C in SVM. SVM-HMM is an iterative algorithm, the ε parameter defines the required accuracy for the algorithm to terminate. A smaller ε will result in higher accuracy, but may need to run more iterations. The C parameter determines how hard non-separable samples should be penalised. A large C will sacrifice larger margin for lower misclassification error, but it will make the execution longer.

We use the whole set of Clementi's Sonatinas Op.36 from performer A, split them into two sets: the training set includes pieces No.2 to No.6, and the testing set includes piece No.1. We train a model with the training set, and use the learned model to generate the testing set. The generated expressive performance is compared to the original recordings to see verify the accuracy of the prediction.

Ideally, the generated performance will be very similar (in expression) to the recording. So, for every pair of the generated and recorded performances, we calculate the distance (defined below) of the performance features, and take the median value of all the distances for every C. Note that each performance feature has its own model, so we will be looking at a performance feature and its C parameter one at a time. First, the generated performance features sequence and the recorded one are normalized to a range from 0 to 1. The normalization is required because we want to tolerate linear scaling. Then the Euclidean distance of the two normalized sequence is calculated and divided by the length (in notes) of the phrase, since the phrase can have arbitrary length.

First we fixed C at 0.1 and tried different ε's: 100, 10, 1, 0.75, 0.5 and 0.1. Then, we fix ε at the optimal value determined in the previous step and test 's: $10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1, 5$. For each ε and C combination, we calculate the distance between the generated pieces and recorded examples for all phrases in the testing set for each performer. Then we take the

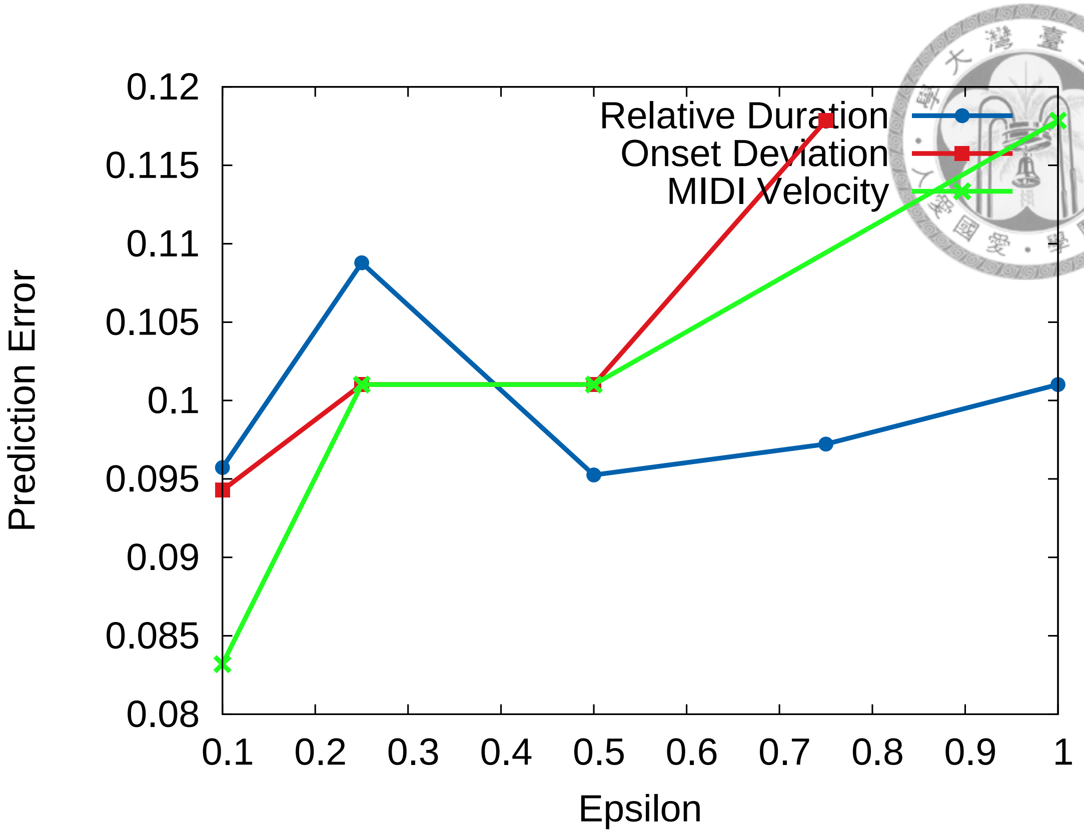

Figure 5.4: Median distance between generated performances and recordings for different ε's

median of all these distances for each ε or C. The optimal ε or C is the one which can minimize the median of the distance.

TODO: median The median distance of the generated performance from the recording for various ε's are shown in Fig. 5.4. The execution time for various ε's are shown in Fig. 5.5. For ε value 100 and 10, the termination criteria is too generous so the learning algorithm terminates almost immediately without any real calculation. Therefore, the model didn't learns anything, the outputs are fixed values for any input. So we abandon the data points for $\varepsilon = 100 or 10$. We can see that the distance drops slowly when ε becomes smaller. We choose $\varepsilon = 0.1$ for later experiments.

As for different C parameter, the accuracy and execution time are shown in Fig.5.6 and Fig. 5.7 respectively. We can not find a clear trend in Fig. 5.6, but just as ε, some C failed to produce meaningful model, so the data points are omitted. In Fig. 5.7 the execution time grows as C goes larger, so considereing the robustness (always producing meaningful model) and time tradeoff, we choose C = 0.1 as our optimal C.

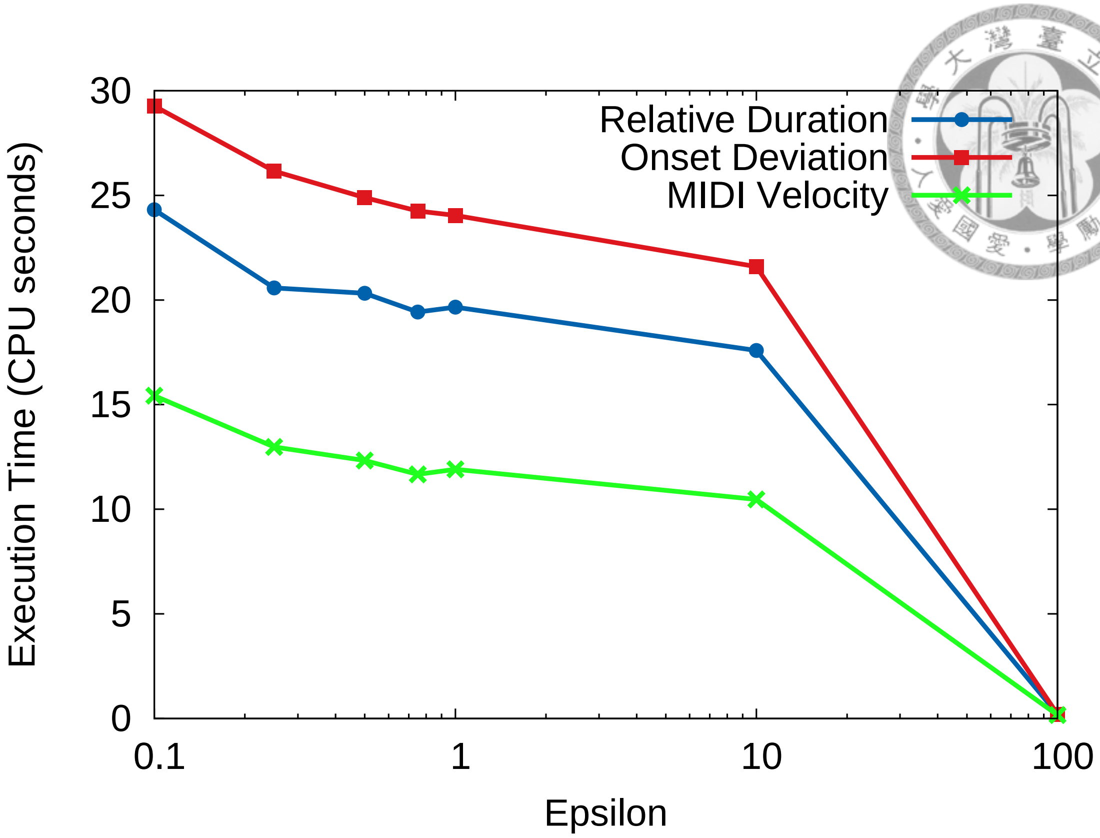

Figure 5.5: Execution time for different ε's

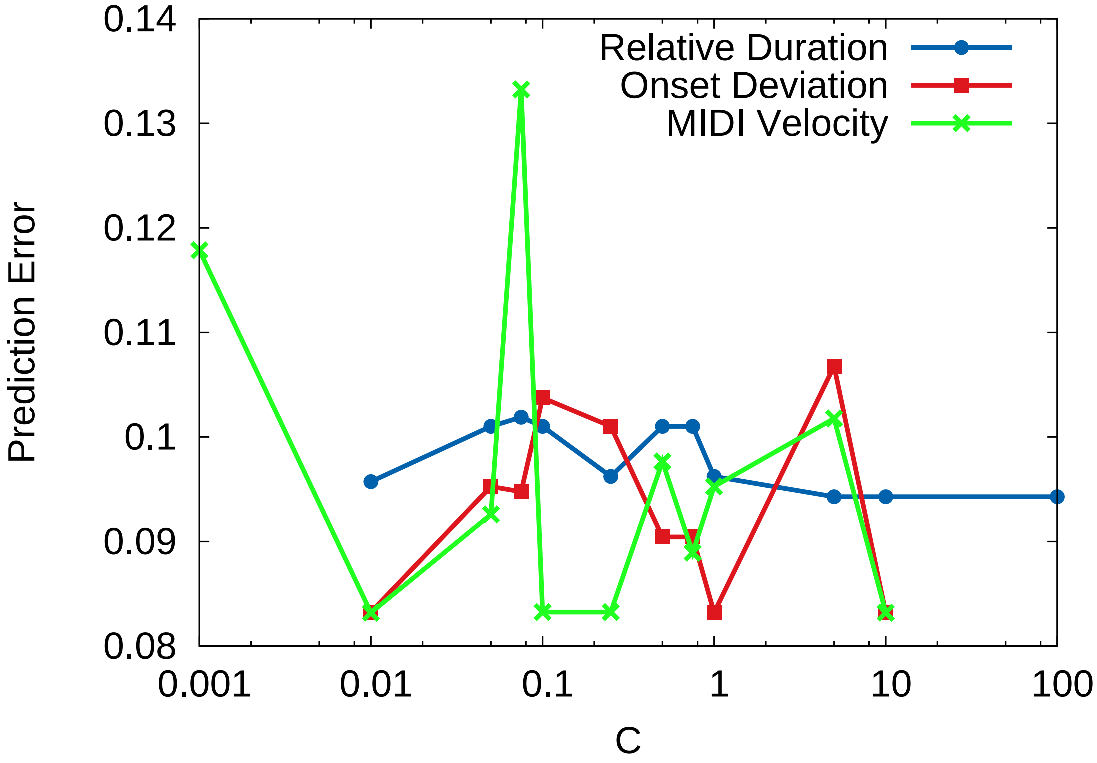

Figure 5.6: Median distance between generated performances and recordings for different C's

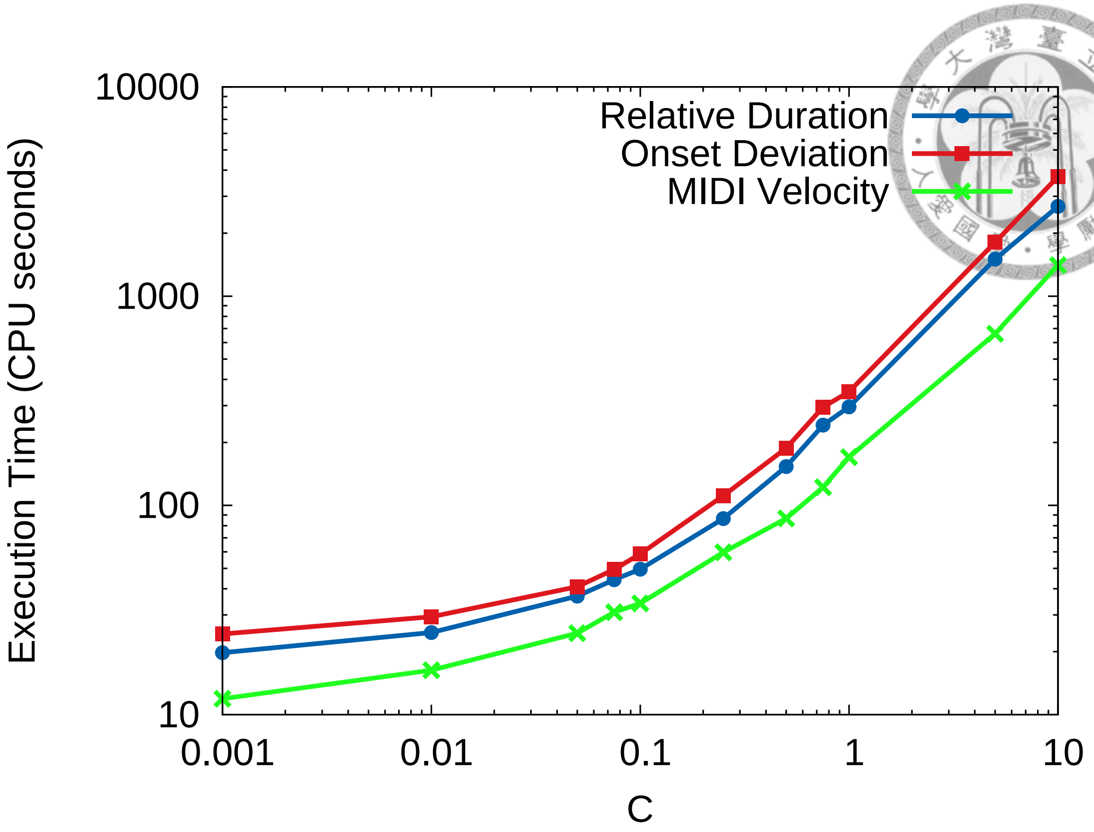

Figure 5.7: Execution time for different C's

5.2.2 Quantization Parameter

Besides ε and C, the number of quantization levels for SVM-HMM input is also has some impact on the execution time. If the performance features are quantized into more fine-grained levels, the quantization errors can be reduced, but the execution time and memory usage will grow dramatically. Also, larger number of intervals doesn't imply more accurate or robust model. Because SVM-HMM is originally used in part-of-speech tagging problem, if we use divide the performance features into more intervals, there will be fewer samples in each interval. But from a statistical learning point of view, it is desirable to have fewer bins with more samples in each, rather than a large number of bins with very sparse samples. For example, if a three note segment is played once in the following MIDI velocity: (60, 70, 80), and the same phrase is played again in (60.1, 69.9, 80.1). If we have a quantization interval width of, say, 0.05, then 60 and 60.1 may be quantized into different bins, and 70 and 69.9 may also be quantized to different bins, so the two phrases will be considered as two different case. However, if the quantization interval width is

1, both phrases may be the same after quantization, which is more desirable because the SVM-HMM algorithm can capture the similarity in the two samples.

Initially, we tried to quantized the values into 1025 uniform width bins, wishing to minimize quantization error. But it take very long (hours, even days) to learn a model, and the output only falls on a very sparse set of values. So we reduce the number to 128. Taking a rough estimate, onset deviation feature rarely exceeds ± 1, so the quantization interval is around $\frac{1-(-1)}{128} = 0.015625$. Most duration ratios falls between zero and three, so the interval is $\frac{3-0}{128} = 0.0234375$. MIDI velocity is roughly around 30 to 90, so the interval is about $\frac{90-30}{128} = 0.46875$. This level of granularity is good enough for our performance system, and can dramatically reduce the execution time without sacrificing the expressiveness of the models.

We repeat the ε selection experiment in the previous section for quantization level of 1025 and 128. The execution time (in CPU second) is shown in Fig. 5.8. The time required for 1025 is larger than 128 by orders of magnitudes. The expressiveness of the output is even improved (evaluated by subjective listening).

REVIEW1

5.3 Human-like Performance

The goal of our system is to create expressive, non-robotic music as oppose to deadpan MIDI. Therefore, we would like to perform a Turing-test-style survey to find out how people think about the our generated expressive music.

In this survey, 1518 computer generated expressive phrases and their corresponding human recording were selected as samples. Each test subject was given 10 randomly selected computer generated phrase and 10 random human recordings, these 20 phrases are presented in random order. He/She was asked to rate each phrase according to the following criteria, which were proposed by the RenCon contes [?]:

1. Technical control: if a performance sounds like it is technically skilled thus performed with accurate and secure notes, rhythms, tempo and articulation.

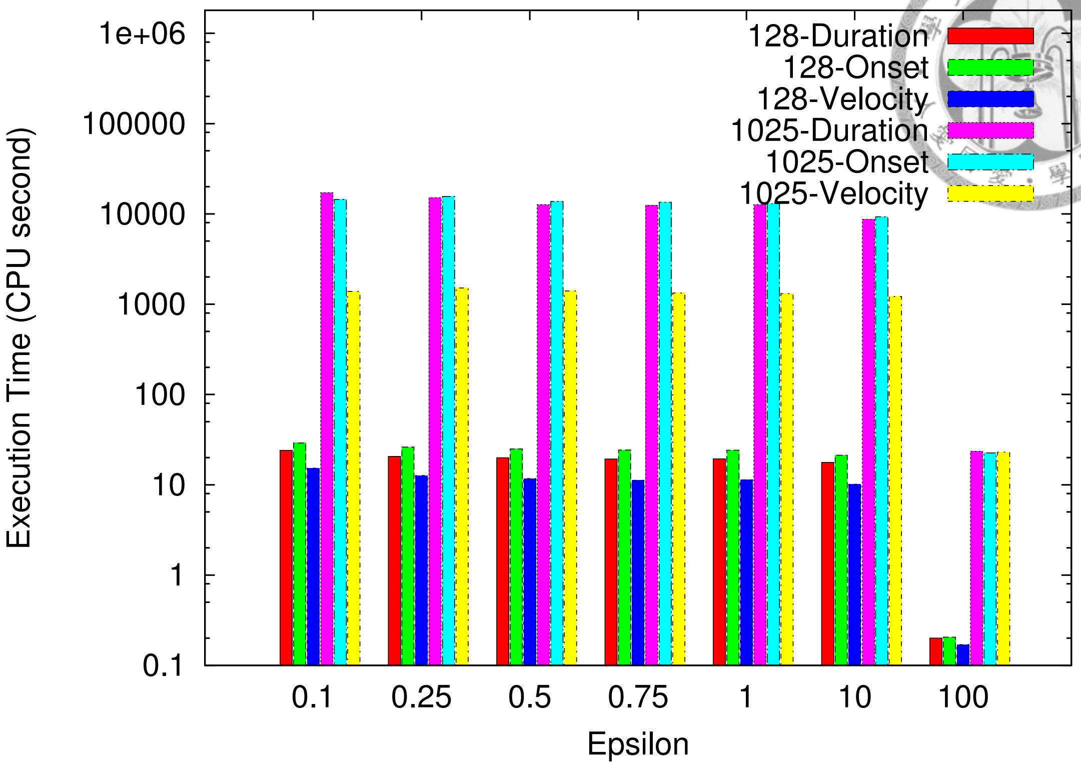

Figure 5.8: Execution time for differnt number of quantization levels

2. Humanness: if the performance sounds like a human was playing it.

3. Musicality: how musical the performance is in terms of tone and color, phrasing, flow, mood and emotions

4. Expressive variation: how much expressive variation (versus deadpan) there is in the performance.

In RenCon, each judge was asked to give separate ratings for each criteria. But we believe that this will be too demanding for less-experienced participant, so we asked each test subject to vote an single overall rating from one to five. One being very bad, five being very good. The test subjects are also asked to report their musical proficiency in a three level scale:

1. No experience in music

2. Amateur performer

3. Professional musician, musicologist or student majored in music

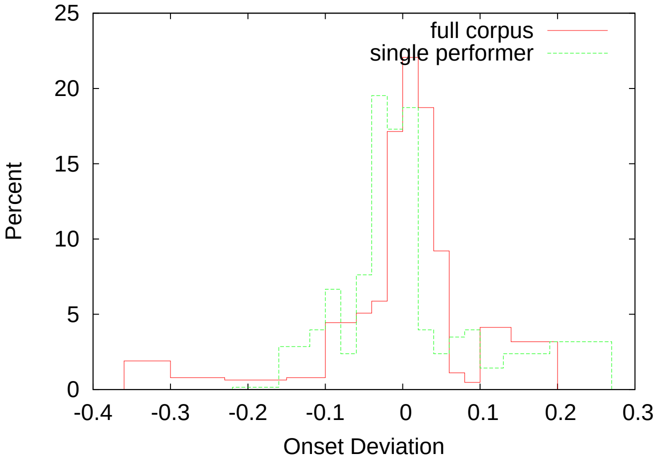

Figure 5.9: Distribution of onset deviation values from full corpus versus single performer's corpus

To generate the expressive performance phrase. We follow a six-fold cross validation pattern: for each performer in the corpus, we use all his/her recorded phrases of Clementi's Op.36 No.2 to No.6 to train a model. Then the model is used to generate all phrases from Clementi's Op.36 No.1. The generate phrases and the performer's recordings of piece No. 1 will all be included to be rated. The process is repeated, but each time the piece excluded for training will be changed to No.2, No.3 and so on. So all six pieces will have a computer generated version and recorded version trained by each player's corpus.

We have also tried using all performer's recording to train a single models. However, the expressive variation from the model is much smaller than a single performer's model. This is because expression from different performer may cancel each other out. The distribution histogram for each performance features are presented in Fig. 5.9, 5.10 and 5.11. The features generated from the full corpus are slightly more concentrated, which results in less dramatic expression.

TODO:Turing test result and discussion

We received 119 valid samples for the survey. Fifty of them are from people with no music background, 59 are from amateur musicians, and the rest 10 are from professional musicians. The average rating given to computer generated performance and human recordings are listed in Table 5.1. It is clear that for professional and amateur musician, the average rating given to human performances are higher than computer performances.

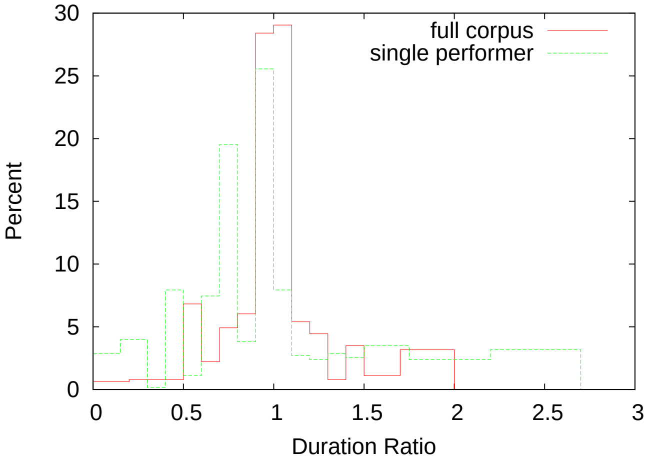

Figure 5.10: Distribution of duration ratio values from full corpus versus single performer's Corpus

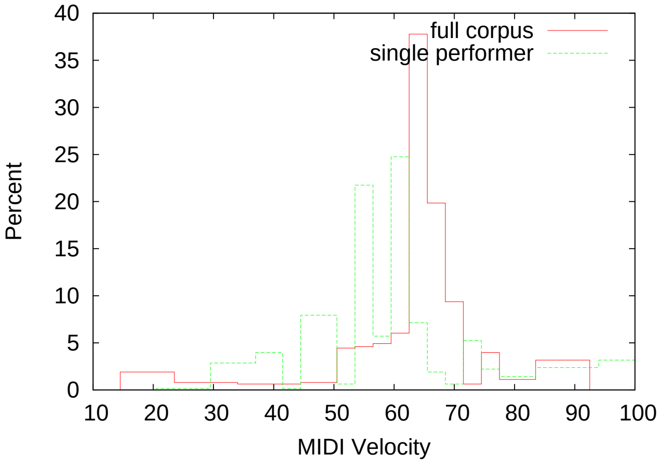

Figure 5.11: Distribution of MIDI velocity values from full corpus versus single performer's corpus

Table 5.1: Average rating for generated performance and human recording

	Computer	Human
No experience	3.243	3.391
Amateur	2.798	3.289
Professional	2.430	3.010
Total	2.952	3.306

Table 5.2: Average rating for generated performance and human recording under different part of the corpus

	A,B		C-F	
	Computer	Human	Computer	Human
No experience	3.067	3.302	3.363	3.451
Amateur	2.680	3.347	2.863	3.286
Professional	2.048	3.162	2.708	2.921
Total	2.776	3.313	3.066	3.323

However, for participants who have no experience in music, the rating is much closer. A Student T-test on the two ratings for participants with no experience yields a p-value of 0.0312, therefore we can't reject the null hypothesis that the two ratings are different under a significance level of 99%.

In order to get more insight from the ratings, we can further divide the performers in the corpus in to two categories based on their piano skill. By the number of mistakes made (Table 4.2), performer A and B are considered more skillful than performer C, D, E and F. The average rating given to the two categories are listed in Table 5.2. The distance between computer and human performances are smaller for less-skillful group (C to F) than the skillful group (A and B). This is probably because our system makes some mistakes that are similar to the mistakes made by less-skillful performers. For example, unsteady tempo, sudden change in loudness, hesitation and emphasising notes that are structurally less important are all common problems that exists in both less-skillful performance and computer generated performance. But for skillful performer, who have better technical control and sense of musical structure, the problems described above will happen less often.

If we look into each individual participant, we can check if a participant gives higher (average) rating to computer or human performances, or equal ratings for both. The number of participants who fall into each categories are shown in Table 5.3. Twenty-six of the

Table 5.3: Number of participants who gives higher rating to generated performance, human recordings or equal rating

	Computer	Equal	Human	Total
No experience	19	7	24	50
Amateur	7	3	49	59
Professional	1	1	8	10
Total	27	11	81	119

Table 5.4: Number of participants who gives higher rating to generated performance, human recordings or equal rating under different part of the corpus

	A,B			C-F			Total
	Computer	Equal	Human	Computer	Equal	Human	
No experience	5	4	6	14	3	18	50
Amateur	2	1	18	5	2	31	59
Professional	0	1	3	1	0	5	10
Total Result	7	6	27	20	5	54	119

non-experienced participants give higher or equal rating to computer than human, slightly higher than twenty-four people who gives higher rating to human. For amateur and professional musicians, the number of people who prefers human are much higher. Table 5.4 is a table similar to 5.3, but split into two categories just like Table 5.2. The results are similar to Table 5.3: the difference between computer and human is higher for skillful performers (A and B) than less-skillful performers (C to F).

TODO:discussion

Chapter 6

Conclusions

REVIEW1 TODO:summary We have created a system that can create expressive music performance from monophonic score notation (with user-annotated phrasing), based on the SVM-HMM algorithm. We have also created a corpus consisting of scores and MIDI recordings from which the system can learn performance knowledge. From our subjective test, we show that although the amateur and professional musician can still differentiate the generated performance from human recordings, subjects with no music background is already giving equal ratings to the generated performance and human recordings.

There are many room for improvement. Structural expressions such as phrasing, contract between sections, or even contrast between movements can be added. Other information like text notations, harmonic analysis and musicological analysis can be added. Supporting homophonic or polyphonic music is also important for the system to be useful. Sub-note expressions like physical model synthesizer or envelope shaping can also be applied to generate performances for specific musical instruments. It's also crucial to test the system on more samples from different genre or music style. We also believe combining rule-based model and machine learning model may be a possible direction for computer expressive music performance research, because machine learning methods can learn subconscious expressions, while rule based system can serve as a high level guideline for structural expression. User can also control the overall expression easily by tweaking the rules. TODO:error model

Bibliography

[1] R. Hiraga, R. Bresin, K. Hirata, and R. KH, ``Turing test for musical expression proceedings of international conference on new interfaces for musical expression," in *Proceedings of 2004 new interfaces for musical expression conference* (Y. Nagashima and M. Lyons, eds.), (Hamatsu, Japan), pp. 120--123, ACM Press, 2004.

[2] ``Sibelius." http://www.avid.com/us/products/sibelius/pc/Play-perform-and-share.

[3] ``Rachmianinoff - Plays Rachmaninoff." https:// www.zenph.com/ rachmaninoff-plays-rachmaninoff, 2009.

[4] A. Kirke and E. R. Miranda, ``An Overview of Computer Systems for Expressive Music Performance," in *Guide to Computing for Expressive Music Performance* (A. Kirke and E. R. Miranda, eds.), pp. 1--47, Springer, 2013.

[5] A. Friberg, R. Bresin, and J. Sundberg, ``Overview of the KTH rule system for musical performance," *Advances in Cognitive Psychology*, vol. 2, pp. 145--161, Jan. 2006.

[6] M. Hashida, N. Nagata, and H. Katayose, ``Pop-E: a performance rendering system for the ensemble music that considered group expression," in *Proceedings of 9th International Conference on Music Perception and Cognition* (M. Baroni, R. Addessi, R. Caterina, and M. Costa, eds.), (Bologna, Spain), pp. 526--534, ICMPC, 2006.

[7] S. R. Livingstone, R. Mühlberger, A. R. Brown, and A. Loch, ``Controlling musical emotionality: an affective computational architecture for influencing musical emotions," *Digital Creativity*, vol. 18, pp. 43--53, Mar. 2007.

[8] N. P. M. Todd, ``A computational model of rubato,'' *Contemporary Music Review*, vol. 3, pp. 69--88, Jan. 1989.

[9] N. P. McAngus Todd, ``The dynamics of dynamics: A model of musical expression,'' *The Journal of the Acoustical Society of America*, vol. 91, p. 3540, June 1992.

[10] N. P. M. Todd, ``The kinematics of musical expression,'' *The Journal of the Acoustical Society of America*, vol. 97, p. 1940, Mar. 1995.

[11] M. Clynes, ``Generative principles of musical thought: Integration of microstructure with structure,'' *Journal For The Integrated Study Of Artificial Intelligence*, 1986.

[12] M. Clynes, ``Microstructural musical linguistics: composers' pulses are liked most by the best musicians,'' *Cognition*, 1995.

[13] M. Johnson, ``Toward an expert system for expressive musical performance,'' *Computer*, vol. 24, pp. 30--34, July 1991.

[14] R. B. Dannenberg and I. Derenyi, ``Combining instrument and performance models for high-quality music synthesis,'' *Journal of New Music Research*, vol. 27, pp. 211--238, Sept. 1998.

[15] R. B. Dannenberg, H. Pellerin, and I. Derenyi, ``A Study of Trumpet Envelopes,'' in *Proceedings of the 1998 international computer music conference* (O. 1998, ed.), (Ann Arbor, Michigan), pp. 57--61, International Computer Music Association, 1998.

[16] G. Mazzola and O. Zahorka, ``Tempo curves revisited: Hierarchies of performance fields,'' *Computer Music Journal*, vol. 18, no. 1, pp. 40--52, 1994.

[17] G. Mazzola, *The Topos of Music: Geometric Logic of Concepts, Theory, and Performance*. Basel/Boston: Birkhäuser, 2002.

[18] W. A. Sethares, *Tuning, Timbre, Spectrum, Scale*. Springer, 2005.

[19] H. Katayose, T. Fukuoka, K. Takami, and S. Inokuchi, ``Expression extraction in virtuoso music performances," in *Proceedings of 10th International Conference on Pattern Recognition*, vol. i, pp. 780--784, IEEE Comput. Soc. Press, 1990.

[20] H. Katayose, T. Fukuoka, K. Takami, and S. Inokuchi, ``Extraction of expression parameters with multiple regression analysis," *Journal of Information Processing Society of Japan*, no. 38, pp. 1473--1481, 1997.

[21] O. Ishikawa, Y. Aono, H. Katayose, and S. Inokuchi, ``Extraction of Musical Performance Rules Using a Modified Algorithm of Multiple Regression Analysis," in *International Computer Music Conference Proceedings*, (Berlin, Germany), pp. 348--351, International Computer Music Association, San Francisco, 2000.

[22] S. Canazza, G. De Poli, C. Drioli, A. Rodà, and A. Vidolin, ``Audio Morphing Different Expressive Intentions for Multimedia Systems," *IEEE MultiMedia*, vol. 7, pp. 79--83, July 2000.

[23] S. Canazza, A. Vidolin, G. De Poli, C. Drioli, and A. Rodà, ``Expressive Morphing for Interactive Performance of Musical Scores," p. 116, Nov. 2001.

[24] S. Canazza, G. De Poli, A. Rodà, and A. Vidolin, ``An Abstract Control Space for Communication of Sensory Expressive Intentions in Music Performance," *Journal of New Music Research*, vol. 32, pp. 281--294, Sept. 2003.

[25] R. Bresin, ``Artificial neural networks based models for automatic performance of musical scores," *Journal of New Music Research*, vol. 27, pp. 239--270, Sept. 1998.

[26] A. Camurri, R. Dillon, and A. Saron, ``An experiment on analysis and synthesis of musical expressivity," in *Proceedings of 13th colloquium on musical informatics*, (L'Aquila, Italy), 2000.

[27] G. Grindlay, *Modeling expressive musical performance with Hidden Markov Models*. PhD thesis, University of Santa Cruz, CA, 2005.

[28] C. Raphael, ``Can the computer learn to play music expressively?," in *Proceedings of the 8th Int. Workshop on Artificial Intelligence and Statistics* (T. Jaakkola and T. Richardson, eds.), pp. 113--120, Morgan Kaufmann, San Francisco, 2001.

[29] C. Raphael, ``A Bayesian Network for Real-Time Musical Accompaniment.," *Neural Information Processing Systems*, no. 14, pp. 1433--1440, 2001.

[30] C. Raphael, ``Orchestra in a box: A system for real-time musical accompaniment," in *Proceedings of 2003 International Joint conference on Artifical Intelligence (Working Notes of IJCAI-03 Rencon Workshop)* (G. Gottob and T. Walsh, eds.), (Acapulco, Mexico), pp. 5--10, Morgan Kaufmann, San Francisco, 2003.

[31] L. Dorard, D. Hardoon, and J. Shawe-Taylor, ``Can style be learned? A machine learning approach towards ‘performing’ as famous pianists.," in *Proceedings of the Music, Brain and Cognition Workshop -- Neural Information Processing Systems*, Whistler, Canada, 2007.

[32] M. Wright and E. Berdahl, ``Towards machine learning of expressive microtiming in Brazilian drumming," in *Proceedings of the 2006 International Computer Music Conference* (I. Zannos, ed.), (New Orleans, USA), pp. 572--575, ICMA, San Francisco, 2006.

[33] R. Ramirez and A. Hazan, ``Modeling Expressive Music Performance in Jazz.," in *Proceedings of 18th international Florida Artificial Intelligence Research Society Sonference (AI in Music and Art)*, (Clearwater Beach, FL, USA), pp. 86--91, AAAI Press, Menlo Park, 2005.

[34] R. Ramirez and A. Hazan, ``Inducing a generative expressive performance model using a sequential-covering genetic algorithm," in *Proceedings of 2007 annual conference on Genetic and evolutionary computation*, (London, UK), ACM Press, New York, 2007.

[35] Q. Zhang and E. Miranda, ``Towards an evolution model of expressive music performance," in *Proceedings of the 6th International Conference on Intelligent Systems*

Design and Applications (Y. Chen and A. Abraham, eds.), (Jinan, China), pp. 1189--1194, IEEE Computer Society, Washington, DC, 2006.

[36] E. Miranda, A. Kirke, and Q. Zhang, ``Artificial evolution of expressive performance of music: An imitative multi-agent systems approach," *Computer Music Journal*, vol. 34, no. 1, pp. 80--96, 2010.

[37] Q. Zhang and E. R. Miranda, ``Evolving Expressive Music Performance through Interaction of Artificial Agent Performers," in *Proceedings of ECAL 2007 workshop on music and artificial life (MusicAL 2007)*, (Lisbon, Portugal), 2007.

[38] J. L. Arcos, R. L. De Mántaras, and X. Serra, ``X. Serra, 1997. "SaxEx: a case-based reasoning system for generating expressive musical performances" ," in *Proceedings of 1997 International Computer Music Conference* (P. Cook, ed.), (Thessalonikia, Greece), pp. 329--336, ICMA, San Francisco, 1997.

[39] J. L. Arcos, R. L. De Mántaras, and X. Serra, ``Saxex: A case-based reasoning system for generating expressive musical performances," *Journal of New Music Research*, vol. 27, no. 3, pp. 194--210, 1998.

[40] J. L. Arcos and R. L. De Mántaras, ``An Interactive Case-Based Reasoning Approach for Generating Expressive Music," *Journal of Applied Intelligence*, vol. 14, pp. 115--129, Jan. 2001.

[41] T. Suzuki, T. Tokunaga, and H. Tanaka, ``A case based approach to the generation of musical expression," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, (Stockholm, Sweden), pp. 642--648, Morgan Kaufmann, San Francisco, 1999.

[42] T. Suzuki, ``Kagurame phase-II," in *Proceedings of 2003 International Joint Conference on Artificial Intelligence (working Notes of RenCon Workshop)* (G. Gottlob and T. Walsh, eds.), (Acapulco, Mexico), Morgan Kaufmann, Los Altos, 2003.

[43] K. Hirata and R. Hiraga, ``Ha-Hi-Hun: Performance rendering system of high controllability,'' in *Proceedings of the ICAD 2002 Rencon Workshop on performance rendering systems*, (Kyoto, Japan), pp. 40--46, 2002.

[44] G. Widmer, ``Large-scale Induction of Expressive Performance Rules: First Quantitative Results,'' in *Proceedings of the 2000 International Computer Music Conference* (I. Zannos, ed.), (Berlin, Germany), pp. 344--347, International Computer Music Association, San Francisco, 2000.

[45] G. Widmer and A. Tobudic, ``Machine discoveries: A few simple, robust local expression principles,'' *Journal of New Music Research*, vol. 32, pp. 259--268, 2002.

[46] G. Widmer, ``Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries,'' *Artificial Intelligence*, vol. 146, pp. 129--148, 2003.

[47] G. Widmer and A. Tobudic, ``Playing Mozart by Analogy: Learning Multi-level Timing and Dynamics Strategies,'' *Journal of New Music Research*, vol. 32, pp. 259--268, Sept. 2003.

[48] A. Tobudic and G. Widmer, ``Relational IBL in music with a new structural similarity measure,'' in *Proceedings of the 13th International Conference on Inductive Logic Programming* (T. Horvath and A. Yamamoto, eds.), pp. 365--382, Springer Verlag, Berlin, 2003.

[49] A. Tobudic and G. Widmer, ``Learning to play Mozart: Recent improvements,'' in *Proceedings of 2003 International Joint conference on Artifical Intelligence (Working Notes of IJCAI-03 Rencon Workshop)* (K. Hirata, ed.), (Acapulco, Mexico), 2003.

[50] P. Dahlstedt, ``Autonomous evolution of complete piano pieces and performances,'' in *Proceedings of ECAL 2007 workshop on music and artificial life (Music AL 2007)*, (Lisbon, Portugal), 2007.

[51] A. Kirke and E. Miranda, ``Using a biophysically-constrained multi-agent system to combine expressive performance with algorithmic composition,'' 2008.

[52] L. Carlson, A. Nordmark, and R. Wikilander, *Reason version 2.5 -- Getting Started*. Propellerhead Software, 2003.

[53] T. Joachims, T. Finley, and C.-N. J. Yu, ``Cutting-plane training of structural SVMs,'' *Machine Learning*, vol. 77, pp. 27--59, May 2009.

[54] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, ``Large Margin Methods for Structured and Interdependent Output Variables,'' *Journal of Machine Learning Research*, vol. 6, pp. 1453--1484, 2005.

[55] Y. Altun, I. Tsochantaridis, and T. Hofmann, ``Hidden Markov Support Vector Machines,'' in *Proceedings of the 20th International Conference on Machine Learning*, vol. 3, (Washington DC, USA), pp. 3--10, 2003.

[56] M. Cuthbert and C. Ariza, ``music21 [computer software],'' 2013.

[57] S. H. Lyu and S.-k. Jeng, ``COMPUTER EXPRESSIVE MUSIC PERFORMANCE BY PHRASE-WISE MODELING,'' in *workshop on Computer Music and Audio Technology*, 2012.

[58] T. Joachims, ``SVMˆhmm: Sequence Tagging with Structural Support Vector Machines,'' 2008.

[59] R. P. Brent, *Algorithms for Minimization Without Derivatives*. 2013.

[60] M. Hashida, T. Matsui, and H. Katayose, ``A New Music Database Describing Deviation Information of Performance Expressions,'' in *International Conference of Music Information Retrival (ISMIR)*, pp. 489--494, 2008.

[61] S. Flossmann, W. Goebl, M. Grachten, B. Niedermayer, and G. Widmer, ``The Magaloff project: An interim report,'' *Journal of New Music Research*, vol. 39, no. 4, pp. 363--377, 2010.

[62] W. Goebl, S. Flossmann, and G. Widmer, ``Computational investigations into between-hand synchronization in piano playing: Magaloff's complete Chopin,'' in

Proceedings of the Sixth Sound and Music Computing Conference, pp. 291----296, 2009.

[63] M. Grachten and G. Widmer, ``Explaining musical expression as a mixture of basis functions," in *Proceedings of the 8th Sound and Music Computing Conference (SMC 2011)*, 2011.

[64] S. Flossmann, W. Goebl, and G. Widmer, ``Maintaining skill across the life span: Magaloff's entire Chopin at age 77," in *Proceedings of the International Symposium on Performance Science*, 2009.

[65] M. Grachten and G. Widmer, ``Linear basis models for prediction and analysis of musical expression," *Journal of New Music Research*, 2012.

[66] S. Flossmann, M. Grachten, and G. Widmer, ``Expressive performance rendering with probabilistic models," in *Guide to Computing for Expressive Music Performance* (A. Kirke and E. R. Miranda, eds.), pp. 75--98, Springer London, 2013.

[67] S. Flossman and G. Widmer, ``Toward a model of performance errors: A qualitative review of Magaloff's Chopin'," in *International Symposium on Performance Science*, (Utrecht), AEC, 2011.

[68] S. Flossmann, W. Goebl, and G. Widmer, ``The Magaloff corpus: An empirical error study," in *Proceedings of the 11th ICMPC*, (Seattle, Washington, USA), 2010.

[69] G. Widmer, S. Flossmann, and M. Grachten, ``YQX Plays Chopin," *AI Magazine*, vol. 30, p. 35, July 2009.

[70] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music*. 1983.

[71] ``KernScores." http://kern.ccarh.org/.

[72] M. Clementi, *SONATINES pour Piano a 2 mains Op. 36 VOLUME I [Musical Score]*. Paris: Durand & Cie., plate d. & c. 9318 ed., 1915.

[73] M. Good, ``MusicXML: An Internet-Friendly Format for Sheet Music,'' in *XML Conference hosted by IDEAlliance*, 2001.

[74] T. Joachims, T. Finley, and C. Yu, ``Cutting-plane training of structural SVMs,'' *Machine Learning*, vol. 77, pp. 27--59, May 2009.

[75] E. Selfridge-Field, *Beyond MIDI: The Handbook of Musical Codes*. MIT Press, 1997.

[76] ``LilyPond.'' http://www.lilypond.org.

Appendix A

Software Tools Used in This Research

REVIEW1 This research won't come into reality without many open-source software tools and free resources, we will walk you through a brief introduction to the softwares we used in this research. Please note that Internet resources come and go very quickly, if the links listed below are no longer valid, you can try to search it using search engines.

Linux Operating System

Most of the tools introduced below runs on modern Linux distributions. The distribution we are using is **Linux Mint Debian Edition (LMDE)**[1] (Linux kernel 3.10), which is a user-friendly Linux distribution based on Debian Testing. User who want to try music-related softwares without installing Linux on their harddrive can try **64 Studio**[2] Linux, which is a live CD distribution with many of the following softwares pre-installed. It also has many kernel optimization for real-time music manipulation. **Ubuntu Studio**[3] is also an option, which has many pre-installed music softwares and is based on the popular Ubuntu Linux.

However, many Linux distribution uses PulseAudio audio server to manage audio device. But a badly configured PulseAudio server will introduce severe latency, which is not acceptable while doing MIDI recording. One workaround is to remove PulseAudio

[1] http://www.linuxmint.com/download_lmde.php

[2] http://www.64studio.com/

[3] http://ubuntustudio.org/

and use raw ALSA (Advanced Linux Sound Architecture) driver instead. But be careful, hardware volume key may fail without PulseAudio.

Programming Languages

Python

Many researcher will choose Matlab or Octave for research projects because they have many useful toolboxes included. However, we believe that research project doesn't exist in vacuum. Drawing insight from the famous 80-20 rule, only 20% of the code are actually doing the core algorithm, the rest 80% are doing file manipulation, configuration, user interaction, and result display and plotting. Therefore, choosing a powerful and easy to write general-purpose programming language is extreme crucial. **Python**[4] construct most of the infrastructure code for this project. Python is super easy to code, and has almost every tool you need to construct a fully functional experiment environment. We will highlight some useful module:

`Music21`[5]

We would like to give special thanks to the `music21` developemnt team. `Music21` is a python toolbox for music notation manipulation and analysis, developed by MIT. `Music21` can parse many score notations like MusicXML, MIDI[6] and more into a very convenient `music21` object data-structure. Researcher can easily filter, split, search, and transform music notations. There are also many music analysis methods and feature extractors included. If you want to do music research, `music21` is a god-sent resource.

[4] `https://www.python.org/`

[5] `http://web.mit.edu/music21/`

[6] By default, `music21` will quantize MIDI input, so if you want to import MIDI recorded from human performance, you need to bypass the default parser and manually disable the quantizer

SciPy[7]

SciPy is a project that contains many useful toolboxes for scientific computation in Python. The SciPy core library and NumPy provides numerical and vector calculation for Python, with similar capability to Matlab. Matlibplot provides plotting tools also similar to Matlab. It's useful for small scale calculation. For heavy duty mathematical calculation, we suggest `R` programming language, which will be discussed in later section.

Simplejson

JSON (JavaScript Object Notation) is a plaintext data-interchange format, similar to XML but much light-weight. JSON is useful in experiment code for two purpose: first, JSON can serve as configuration file, it easy to parse and easy to edit. Second, JSON can serve as intermediate data file between each experiment module. For example, we use JSON to send extracted features from feature extractors to the machine learning module. Although plaintext takes more storage than binary file, but it's much easier for debugging because it's human readable. And you can simply parse the intermediate values and plot it using other plotting program. Python provides build-in support for JSON format via `json` and `simplejson` packages.

Argparse

`Argparse` provides command line argument parser for python scripts, using commandline arguments with configuration file, you can create very flexible, extendible and easy to automate scripts.

Logging

The built in `logging` module can print logging information with predefined format, and supports log level. By using log level, you can print debug information during development, and shutdown all debug message during production simply by changing the log

[7] http://www.scipy.org/

level flag.

R[8]

R is a programming language for statistical calculation, but it can also do general purpose math and plotting very well. R follows a functional programming design, so it may take some time to learn for people who only have experience in C/C++, Java or other imperative/Object-oriented programming language. But it is definitely a great tool for mathematical operations and plotting. We use R for experiment data analysis and for linear regression in early version of this research. R and Python can work seamlessly through the rpy package.

Score Manipulation and Corpora

MusicXML and MuseScore

MusicXML[9] is a digital score notation format based on XML. It is well supported in most commercial music typesetting software. For other music typesetting format you may want to look at LilyPound. To view and edit musicXML score, we use the open-source software **MuseScore**[10], it provides basic editing capability, and can export score as PDF. However, MuseScore often crash while loading bad-formatted musicXML file, so sometimes you need to look into it log file and fix the ill-formated XML via a text editor.

Corpora

Music21 contains a corpus[11], which will be automatically installed if you accept the licence term during music21 installation. It covers a wide range of composers from early music, classical music to folk songs, with various genre and musical style. Another public

[8] http://www.r-project.org/

[9] http://www.musicxml.com/

[10] http://musescore.org/

[11] http://web.mit.edu/music21/doc/systemReference/referenceCorpus.html

available corpus is called **KernScore**[12], which provides a better search engine. You can find works by composer, genre, form or other criteria. There are even a special section containing monophonic works. Scores from both corpus can be loaded and transformed in to desired format via `music21`.

MIDI Recording

Rosegarden[13] is a digital audio workstation (DAW) software designed for MIDI. It can record, edit, mix and export MIDI tracks. To actually hear the music, you need a MIDI synthesizer to work with Rosegarden. **Timidity++**[14] is built-in in many Linux distribution, and it provides a commandline interface to synthesize MIDI directly into a WAV file. However, the default sound quality from Timidity++ is not very satisfying, so we suggest using the qSynth, which is a QT front end for **FluidSynth**[15]. The default soundfont that comes with FludiSynth has very good sound quality.

With all these music softwares, it will soon be very hard to control the interconnection between software modules. This is when **JACK**[16] comes to help. JACK is like a virtual "plug-board" for software that implements the JACK interface. It provides a central place in which you can control how the music data flow interconnects between software and hardware.

Audio Manipulation

When MIDI files are synthesized into WAV format, there are many tools that can help editing them. The most easy to use software with GUI is **Audacity**[17], it can edit and mix audio tracks.For commandline tools (in case you need scripting), **lame**[18](MP3 encoder),

[12] `http://kern.ccarh.org/`
[13] `http://www.rosegardenmusic.com/`
[14] `http://timidity.sourceforge.net/`
[15] `http://sourceforge.net/projects/fluidsynth/`
[16] `http://jackaudio.org/`
[17] `http://audacity.sourceforge.net/`
[18] `http://lame.sourceforge.net/`

oggenc[19](ogg vorbis encoder) and **FFmpeg**[20] are very helpful for file format transformation. To cut and combine audio tracks from commandline, use **SoX**[21].

Data Visualization

As mentioned before, `R` and `Matlibplot` are good candidate for visualizing experiment data. But if you don't want to learn the syntax of R or Python, you can try **gnuplot**[22]. Gnuplot is a interactive (and scriptting) environment for generating various types of plot like line plots or bar charts. It works particularly well if you use `grep` to extract datas for many files, say, extracting execution time information from logs.

SVM$^{\text{hmm}}$

SVM$^{\text{hmm}}$[23] is an implementation for Structural Support Vector Machine with Hidden Markov Model output. It's developed by Thorsten Joachims from Cornell University. It is based on SVM$^{\text{struct}}$, a more general framework for Structural Support Vector Machine. There are many other SVM$^{\text{struct}}$ extensions such as Python or Matlab API.

Other

Sometimes the machine learning algorithm will run for a very long time. Then it's better if you can find a server that runs 24-7 in your home or laboratory. You can install a **ssh** server on that machine, and controls the experiment execution remotely. However, the experiment program will be terminated once you log out the `ssh` session. You can run your experiment program in **tmux**[24], a terminal multiplexer, instead. It will keep your program running even if you log out.

[19] `http://www.vorbis.com/`
[20] `http://www.ffmpeg.org/`
[21] `http://sox.sourceforge.net/`
[22] `http://www.gnuplot.info/`
[23] `http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html`
[24] `http://tmux.sourceforge.net/`

Modern machines often have multi-core CPUs. But if your program only runs in one core, you waste the CPU resources and also your time. **Gnu-parallel**[25] can dispatch multiple instances of your script or program to each core. It will automatically find new job to run when the previous one is finished, so the CPU will always run on its full capacity.

We use **git**[26] for source control. LaTeX[27] is used to typeset this document.

Summary

We have reviewed many software tools used to construct this research. We want to emphasize that it is totally possible to use *only* free and open-source software to do all these heavy lifiting. We encourge the reader to try these tools out, spread the words and even contribute to these projects. By doing so we can create a more friendly academia ecosystem and make the world a better place.

[25] `http://www.gnu.org/software/parallel/`

[26] `http://git-scm.com/`

[27] `http://latex-project.org/`