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 年〇〇月〇〇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

請打開並編輯abstractCH.tex

關鍵字：壹、貳、參、肆、伍、陸、柒

Abstract

Open and edit abstractEN.tex

Key words:A, B, C, D, E, F, G

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Computer generated music, such as synthsized MIDI, are often considered robotic and unexpressive. But we have already witnessed the fluid and lively sound generated by state-of-the-art text-to-speech systems. This inspired us to develop a system that can read a music score and play it in an expressive, humanly way. Such system can be used for audiolizing score notation editing software, creating interactive media content, and generating royality-free music.

Established pianists always has his/her own distinctive style. Such sytle distinguised himself/herself from all the other pianists. If the expressive performance system can learn the style of a performer, it might be able to provide musicological insight of performance styles. Furthermore, we can even make a mastero who is no longer with us play music he/she never played in his/her lifetime.

TODO:Discuss Mazolla's theroy of expressive performance

1.2 Goal

The ultimate goal of a expressive performance system is to be able to play any music in a expressive and human-like manner, or even have creative exprssions. But due to the technical and time constrain, we need to define a more pratical goal for our research. We

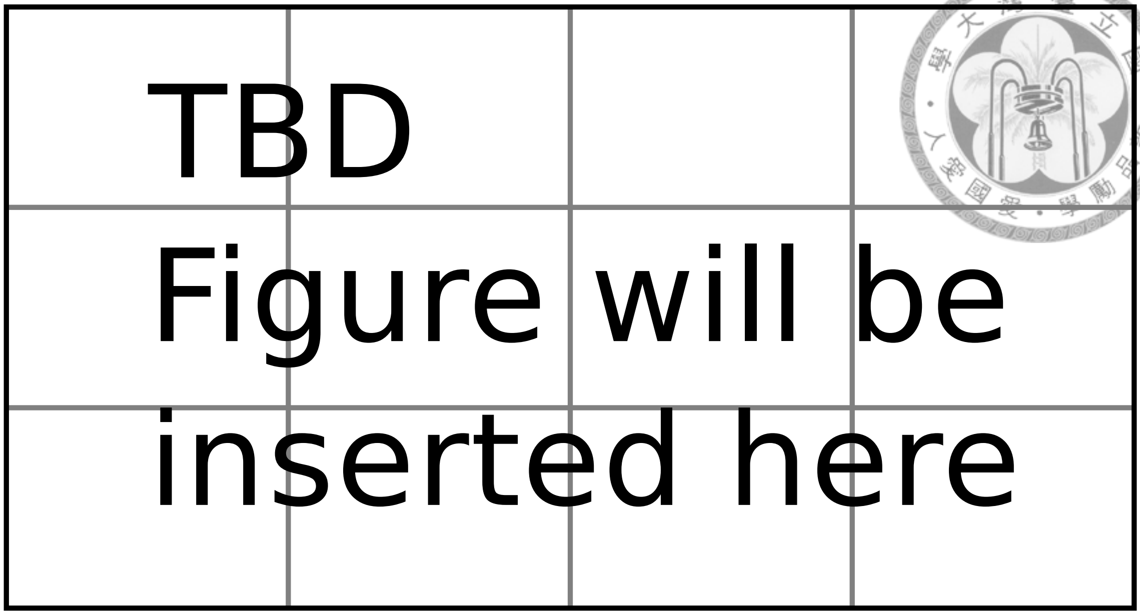

Figure 1.1: From Composer to Performance

wish to build a computer exprssive performance system based on offline supervised learning. The system will be able to learn any play monophonic melodic phrases. By learning from human recordings, the system will be able to generate expressive performance from previous unseen music notation (also in phrases). The performance style will be controlled by the learning material, if a lot of recordings from different performers are fed in, the system should be able to perform a general human-like expression. If only recordings from a single performer is given, it should learn the particular style of the performer.

1.3 Contribution

The major contribution of this thesis is appling structural support vector machine on expressive performance problem. Also we tried to provide a public corpus for expressive performance. TODO:Overview of the chapter sctructure of this thesis

Chapter 2

Previous Works

Researches on computer expressive music performance lags computer composition by almost a quarter of a century, starting around the end of the 1980s [1]. Early work includes the KTH system [1]. Reviewing computer expressive performance systems is a hard task, because there are exists large difference in the goals they wants to achieve, which makes evaluation and comparison difficult. To make things worse, although there is a contest called RenCon [2] in which researches compete their performances, there is no training and benchmarking corpus available, so if a system has never entered the contest and doesn't make its source code available, there is no way to compare them.

In this chapter, we will follow the categories summarized in [1]. Readers are suggest to refer to [1] because it gives very detailed discussion on all the systems. First we will discuss the various goals of computer expressive performance. Second, systems will be reviewed by their methods used. Finally, we will highlight some systems with special features.

2.1 Varying Goals and Evaulation

The general goal of a computer expressive performance system is to generate expressive music, as opposed to the robotic and nu-human sound of rendered MIDI or other digital score format. But the definition of “expressive” is ambiguous. The following are the most popular goals a computer expressive performance system wants to achieve:

1. Reproduce human performance.
2. Perform music notations in a non-robotic way.
3. Accompany a human performer.
4. Validate musicological models of expressive performance.
5. Directly render computer composed music works.

Systems that are designed to reproduce human performance usually has a certain performer in mind, like the Zenph re-performance CD [3] which will reproduce the performance style of Rachimaninov, it can even use Rachimaninov's style to perform pieces the musician never played in his lifetime.

Some systems try to perform music notations in a non-robotic way, without a certain style in mind. These systems has been employed in music typesetting softwares, like Sibelius [4], to play the typesetted notation.

Accompaniment systems try to render expressive music that act as an accompaniment for human performer. This kind of systems has great value in music education. The challenge is that the system must be able to track the progress of a huamn performance, which is itself another established topic called score following. And the rendering must be done in real-time, which is uncommon in the field of computer expressive music performance.

The next goal is to validate musicological models. Musicologist may develop theories on music performance expressivity, and wants to validate them by experiments. These system focus more on the specific features that the theory tries to explain, so may not cover every aspect of performance.

Finally, some systems combines computer composition with performance. The benefit of this approach is that the performance module can understand the intention of the composition without the need to interpret them from the 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.

TBD	TBD
TBD	TBD

Table 2.1: List of Reviewed Systems

The goals discussed above imply different level of musical creativity. Huamn performance reproduction requires mimicry over creativity, on the other hand, system that plays its own composition can have a large range of creativity to explore. The methods employed also limits the ability of creativity, which will be discussed in the next section.

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

2.2 Researches Classified by Methods

Dispite the difference between goals of different expressive performance systems. All of them needs some way to create and apply performance knowledge on unexpressive music. The performance can come from predefined rules or being learned.

Using rules to generate expressive music is the earliest approach tried. Director Musices [5] being one of the early works that is still a living project now. Most of the above systems focus on expressive attributes like note onset, note duration and loudness. But Hermode Tuning System [6] focus more on intonation, thus it can generate expressions that requires string instruments techniques. Pop-E [?] is also a rule-based system which can generate polyphonic music, using its syncronization algorithm to syncronize voices. Computational Music Emotion Rule System [7] pust more emphasis on rules that express human emotions. Other systmes like Hierarchical Parabola System [5] [8] [9] [10], Composer Pulse System [11,12], Bach Fugue System [13], Trumpet Synthesis System [14,15] and Rubato [16, 17] are also some systems that use rules to generate expressive performance. Rule-based systems are effective and don't require a long training period before use. But some of the performance nuance may be hard to describe in rules, so there is a natural 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 learning. Many machine learning methods have been applied to expressive performance problem. One of the easiest form is to use linear regression, systems like Music Interpretation System [?, 18, 19] and CaRo [20--22] both uses linear regression to learn performance knowledge. But assuming the expressive performance as a linear system is clearly not true. So Music Interpretation System use try to solve it by using AND operations on linear regression results to achieve non-linearity. But still linear regression is too simple for this highly non-linear problem.

Many other learning algorithms have been tested with success: ANN Piano [?] and Emotional flute [23] uses artificial neural network. ESP Piano [24] and Music Plus One [25--27] uses Statistical Graphical Models, although the later one focus more on accompaniment task rather than rendering notation. KCCA Piano System [28] uses kernel regression. And Drumming System [29] tried different mapping models that generates drum patterns.

Evolutionary computation has also been applied, genetic programming is used in Genetic Prgramming Jazz Sax [30]. Other examples include the Sequential Covering Algorithm Genetic Algorithm [31], Generative Performance Genetic Algorithm [32] and Multi-Agent System with Imitation [?, 33]. 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 to generate performance. SaxEx [34--36] use fuzzy rules based on emotions to generate Jazz saxophone performance. Kagurame [37, 38] style (Baroque, Romantic, Classic etc.) instead of emotion. Ha-Hi-Hun [39] takes a more ambitions approach, it's goal is to generate a piece X in the style of and expressive performance example of another piece Y. Another series of researches done by Widmer at el. called PLCG [?] uses data-mining to find rules for expressive performance. It's successor Phrase-decompoisition/PLCG [?] added hierarchiacal phrase structures support to the original PLCG system. And the latest research in the series called DISTALL [?] 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 the last category of computer expressive performance we will discuss here has a great advantage over the previous ones, by combining computer composition and performance, the performance part of the system can directly understand the intention of the compoisition. Ossia [?] and pMIMACS [?] are two examples of this category. This approach provides great possibility for creativity, but they can only play their own compoisition, which is rather limited.

TODO:Figure:selected figures from previous works

2.3 Additional Specialties

Most expressive performance systems generates piano performance, because it's relativly easy to collect samples for piano. Even if no instrument is specified, the final performance will often be rendered using piano timbre. Some systmes generates music in other instruments, such as saxophone [?], trumpet [?], flute [?] and drums [?]. These systems reqires additional model for the instruments, as a result, they can produce expressions that requires instrumental skills.

The genre of music that a system plays is also a special feature one might have. For systems that doesn't specify the genre, usually western tonal music will be the genre of choice. Composers like Mozart, Chopin and so on well accepted by the public, their scores and literatures are easily accessable. However, both saxophone-based works choose Jazz music, because saxophone is an iconic instrument in Jazz performance. The Bach Fugue System [13], obviously, focus on fugue works composed by bach.

The ability to perform polyphonic music is also a rare feature of computer expressive performance systems. Performing polyphonic music requires syncronization between voices, while allowing each voice to have their own expression. Pop-E [?] use a syncronization mechniasm to acheive 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 [28]can generate homophonic music -- an upper melody with an accompnaiment -- which is common in piano music. Music Plus One [25--27] is a little bit different because it's a accompaniment system, it adapts non-expressive orchastral accompaiment track to user's performance. Other systems usually generates monophonic tracks only.

Chapter 3

Proposed Method

3.1 Overview

The high-level architecture of the purposed system is shown in figure 3.1. The system has two phases, the upper half of the figure is the learning phase, while the lower half is the performing phase. In the training phase, score and expressive performance recording pairs are feed in, they will serve as training examples for the machine learning algorithm, namely structural support vector machine with hidden markov model output (a.k.a SVM-HMM). The learning algorithm will produce a performance knowledge model, which can be used to generate expressive performance hereafter. 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 synthsizer to produce audible performance.

The system is not intend to add fixed expression to all pieces. Rather it is intended to perform music according to the style which the user wants. This kind of user interactivity can be achieved in two ways: first, the user can choose the training corpus. A user can select a subset of training sample to produce unique models. For example, if the corpus contains only one performer's recordings, the learned model will very likely have a unique style of the performer. Second, the phrasing of a song is given by the user. Since phrasing controls the overall structural interpretation of a music piece, and form a breathing feeling

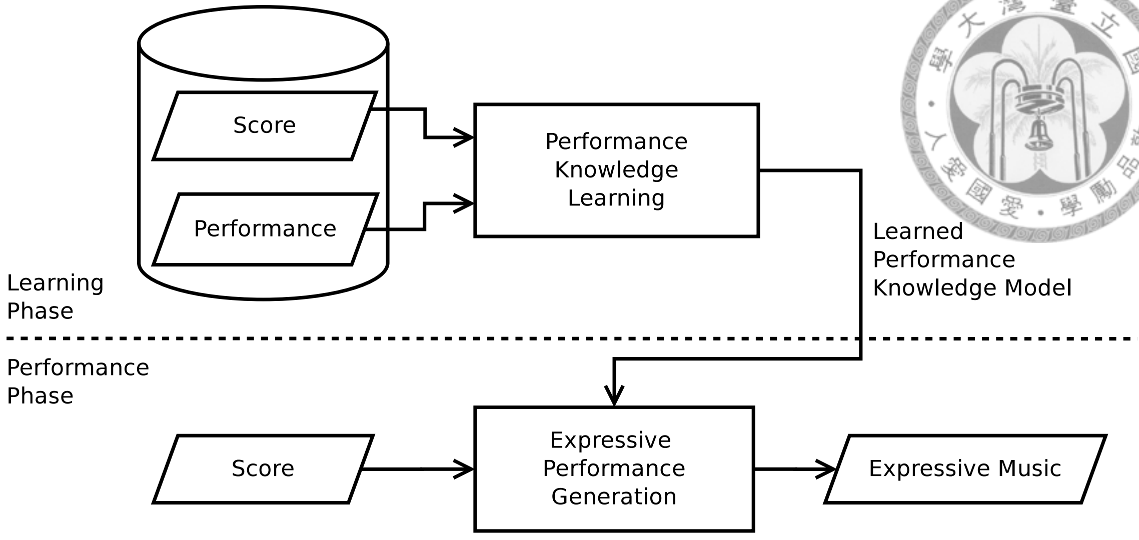
Score
Performance
Performance Knowledge Learning
Learned Performance Knowledge Model
Learning Phase
Performance Phase
Score
Expressive Performance Generation
Expressive Music

Figure 3.1: High Level System Architecture

in music, user is given direct control over the performance.

There are still some constrains for the system now. The scores must be monophonic and contains only one musical phrase. One has to manually label the phrasing for any scores used. The learning algorithm, namely SVM-HMM, can only perform offline learning, so the learning phase can only work in a non-realtime scenario. The generating phase can work much faster though, it can produce expressive music almost instantaneously after loading the score. All the scores are loaded in batch, the system currently don't accept streaming input.

In the following sections, we will discuss the detail steps in the learning and performing phases, and some implementation detail used in each step. Since SVM-HMM learning requires features to be extracted from samples, we will discuss the features used in the end of this chapter.

3.2 A Brief Introduction to SVM-HMM

In this thesis, we use structural support vector machine(structural SVM) as the learning algorithm. Unlike traditional SVM algorithm, which can only produce univariate prediction, structural SVM can produce strctural predictions like tree, sequence and hidden Markov model. Structural SVM with hidden Markov model output (SVM-HMM) is applied to part-of-speech problem with success. This fining lead us to the idea to use SVM-HMM for the expressive performance 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, accroding to it's role in the musical phrase. To illustrate this, consider a note which is the end of the phrase, which is noramlly forms a cadence, and a note which is only a embalishment. The first note will probably be played louder and sustain longer than the second note. With this similarity in mind, we believe SVM-HMM will be a good candidate for expressive performance.

The prediction problem in SVM can be described as finding a function

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

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 (classfication) or real values (regression). But structural SVM extends the framework to generate structural output, such as tree, sequence, or hidden markov model, in this case. To extend SVM to support structured output, the problem is modified to find a discriminant function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$, in which the input/output paris 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 accomidate various kind of structures.

In order to apply SVM, we need a way to measure the accuracy of of a prediction, in other words, we have to define a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to R$. A loss function has the following property:

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

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

Also, the loss function is assumed to be bounded. Let's assume the input-output pair (x, y) is drawn from a join distrution 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, calculted 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, 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 not only want a solution, but 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 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 [40] 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.

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

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

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

But the above quadratic programming problem has a extreme large number ($O(n|\mathcal{Y}|)$) of constrains , which will take considerable time to solve. So [40] 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 violates by more than the desired precision, the constrain is added to the working set. And 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. [41], 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 follwing formula is the 1-slack version of slack-rescaling structural SVM:

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

And margin-rescaling structural SVM:

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

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 [**?**], 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

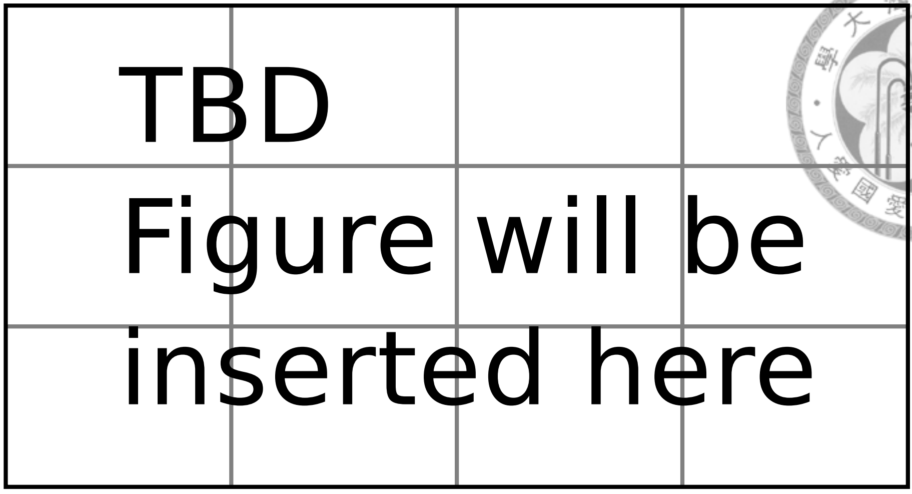

Figure 3.2: Hidden Markov Model

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.

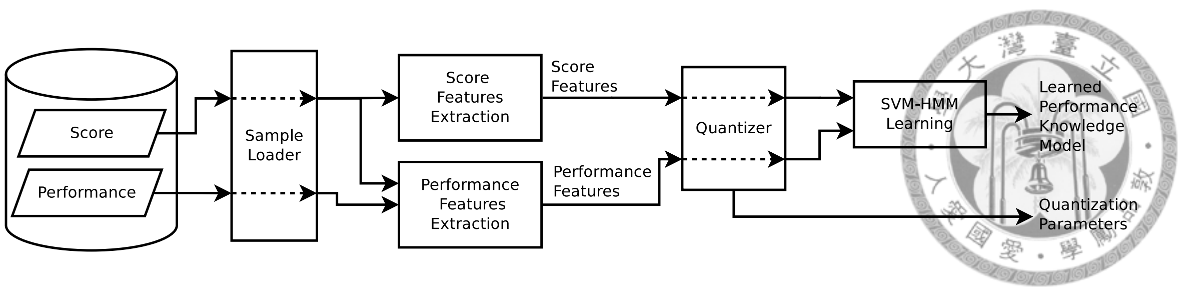

Figure 3.3: Learning Phase Flow Chart

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

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. Figure 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 training samples are loaded as machine-readable format, and have their features extracted. Two types of features will be extracted from the samples, first, the information from the score only are called score features; second, the information of the expressive performance with respect to the score is called the performance features. In order to generate new expressive performance, we want to learn a prediction model by which we can predict the performance features given the score model. 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 learning part is done by a supervised learning algorithm. In this thesis, we choose Structural Support Vector Machine with Hidden Markov Model Output (SVM-HMM) as our learning algorithm.

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` library [**?**] 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. The loaded score are then handed to the feature extractor module.

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 header [**?**], so we must explicitly specify that this value, and 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. And further more, this enables parallel processing during feature extraction. To achieve this, each feature must implement a common feature extractor interface, and feature extractors should not communicate with each other.

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. But since we want to avoid complex scheduling with dependency, the “duration” feature is extracted during the extraction of the “relative duration” feature, no matter if the “duration” feature is extracted prior of after the “relative duration”. Therefore, the “duration” feature extracted has computed twice. To avoid redundant computation of the feature extractors, we implemented a caching mechanism. Say, the “duration” feature had been computed once, it's value will be cached during this execution session; when the “relative duration” extractor calls the “duration” extractor again, the value is retrieved from cache without recalculation. This method can speed up the execution while keeping the system complexity low.

The extracted features are aggregated and stored into a JavaScript Object Notation (JSON) file. By saving the features in a human-readable intermediate file, the researcher 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 tried linear regression with limited success [42]. However, the assumption of linearity is an oversimplification. So we switch to Structural Support Vector Machine with Hidden Markov Model Output (SVM-HMM) [40, 41, 43] as our supervised learning algorithm.

The SVM-HMM learning module loads the JSON file from the previous stage, and rearrange the features to fit the required input format of the SVM-HMM learner program. However, the features from the previous stage are real numbers, but SVM-HMM only takes discrete output, so quantization is required here. For each feature, the quantizer calculates the overall mean and standard deviation from all training samples. Then the quantizer divides the range between mean minus three standard deviations to mean plus three standard deviations into 1024 intervals. Feature values below mean minus three standard deviations are quantized to the lowest bin, while values larger than mean plus three standard deviations are quantized to the highest bin. The range of three standard deviation and 1024 intervals are chosen by experience, which can be modified to fit different corpus. The mean, standard deviation and number of intervals information are stored in a file for the performing phase to dequantize the output.

The theoretical background of SVM-HMM is already described in Section 3.2, to implement the algorithm we leverage Thorsten Joachims's implementation called SVM^{hmm} [44]. SVM^{hmm} is an implementation of structural SVMs for sequence tagging [43] using the training algorithm described in [40] and [41]. The SVM^{hmm} package contains a model training program called `svm_hmm_learn` and a model prediction program called

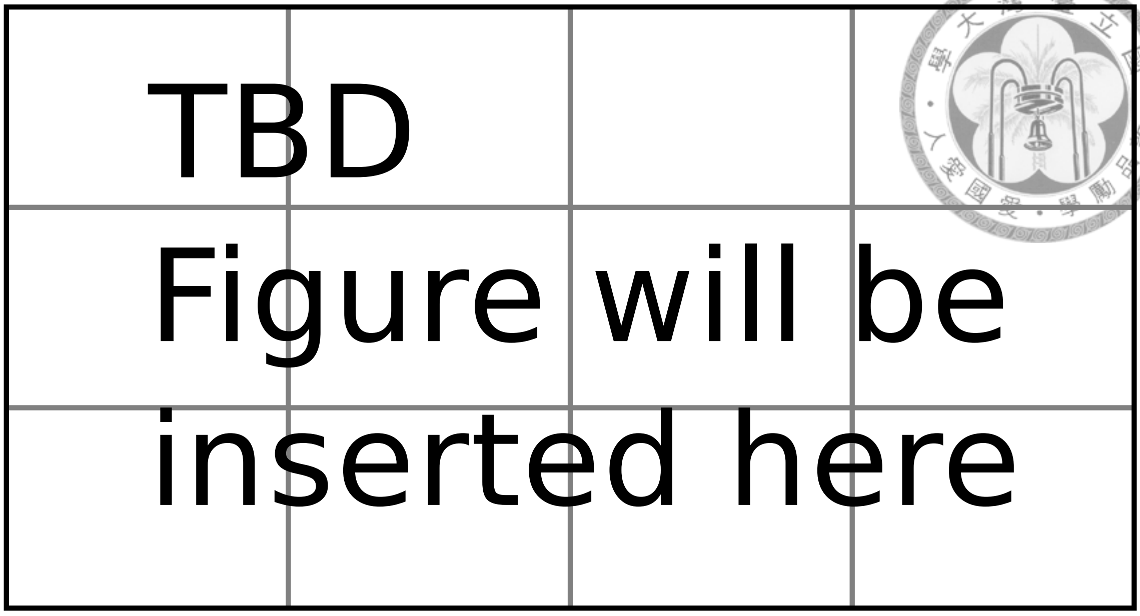

Figure 3.4: Example input file

svm_hmm_classify, which will be used in the performing phase. For structural simplicity, we train a separate model for each quantized performance feature, each model uses all the quantized score features to try to predict a single performance model. Thesvm_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
```

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 space. And anything after the # is comments. An example of the training file is shown in Figure 3.4

TODO: partial model

There are some key parameters need to be specified for the training process. First the C parameter in SVM, which controls the trade-off between low training error and large margin. Larger C will result in lower training error, but the margin may be smaller. Second, the ϵ parameter controls the required precision for the constrains. The smaller

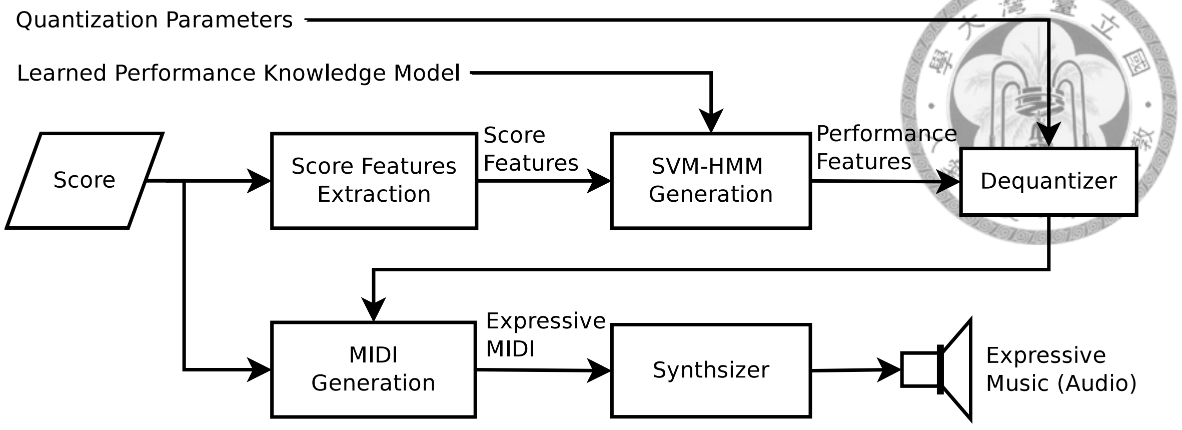

Figure 3.5: Performing Phase Flow Chart

the ϵ, the precision will be higher, but may require more computation. 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 binary representation of the SVM-HMM model parameters, such as the support vectors and other informations, which represents the performance knowledge learned. Since it takes considerable time 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 offline 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.

$\boxed{\text{BOOKMARK}}$

3.4 Expressive Performance

With the performance knowledge learned, we can start to perform music expressively. The performing pharse share the same sample loader and feature extractor module with the learning phase. But in the performing phase, the input is only a score file (musicXML containing a single phrase), which is loaded and has its score features extracted. The extracted score features are also stored into a JSON file for the SVM-HMM generation module to load. The performance knowledge model is also an input in this phase. The SVM-HMM generation module will use the learned model and the quantized score features to determine the performance features for the given score. An MIDI generation module will apply those performance features onto the musicXML 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 is used to render the MIDI file into WAV or MP3 format. BOOKMARK

3.4.1 SVM-HMM Generation

As in the learning phase, the score features are stored in a JSON file. The SVM-HMM generation module first load this JSON file, and preform the same quantization as the learning phase. The quantized score features are then transformed into the same format as the training file, but the `PERF` fields 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. If we already know the real performance feature value, say, we use part of the corpus as training set and the rest as testing set, the `PERF` can be set to the "answer" of the testing set, so the `svm_hmm_classify` program will calculate the accuracy of the prediction.

3.4.2 MIDI Generation

de-quantize Since the output of the SVM-HMM learner is the quantized label for the performance feature, dequantization is required to turn those values back to real-valued performance features. The dequantizer will load the quantization parameters from the learning stage to understand the range and intervals used in the quantizer. Each quantization label is dequantized into the mean value of the interval it belongs.

The performance features are than applied onto the input score. For example, if a performance feature represents the note's duration should last for 1.2 times of its nominal value, the duration in the score is multiplied by 1.2. After all the performance features are applied, the expressive version of the score is stored in MIDI format using the `music21` library.

TODO:Dramatization/post processing

3.4.3 Audio Synthesis

In order to actually hear the expressive performance, the MIDI file is then rendered by a MIDI synthesizer. Since the output is standard MIDI file, the user can choose any compatible software or hardware synthesizer. We choose `timidity++` software synthesizer for this job. The output is an WAV(Waveform Audio Format)file, which is then compressed into MP3 (MPEG-2 Audio Layer III) by `lame` audio encoder.

Because sub-note-level expression is not the primary goal of this research, we choose to use standard MIDI synthesizer 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

3.5 Features

The system is trying to mimic the process of human performance: the musican reads the explict and implict cues from the score and transform them into musical expressions. So the features can be categorized into two category: score features and performance features. Score features are information contained in the score. Performance features corresponds to the musical expression. The basic time unit for both features are a note.

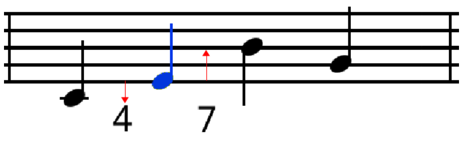

Figure 3.6: Interval from/to neighbor notes

3.5.1 Score Features

Score features includes:

Relative position in a phrase: The relative position of a note in the phrase. From 0% to 100%. This feature can catch the musical hint of the opening and closing of a phrase.

Relative pitch: The pitch (in semitone) of a note relative to the pitch range of the phrase. 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 direction of melody movement. Measured in semitone.

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

See figure 3.6 for example.

Interval to the next note: The direction of melody movement.

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

See figure 3.6 for example.

Note duration: The duration of a note in beats. TODO: discuss how grace notes are handled, currently its

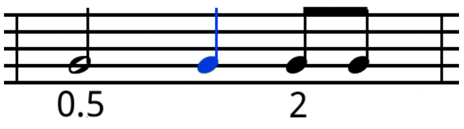

Figure 3.7: Duration from/to neighbor notes

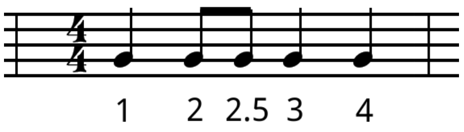

Figure 3.8: Metric position

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 figure 3.7 for example.

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 figure 3.7 for example.

Metric position: The position of a note in a measure, measured by the beat unit defined by the time signature. For example, a $\frac{4}{4}$ time signature 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. See figure 3.8.

3.5.2 Performance Features

Performance features includes:

Relative onset time bias: The onset time of a recording will not be exactly as the ones

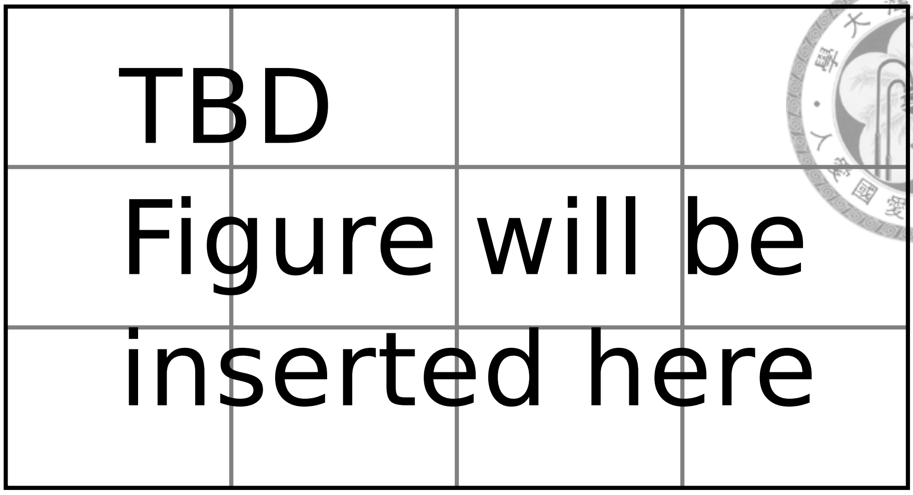

Figure 3.9: Example Score Features

indicated on the score. Given a fixed tempo (beats/second), the score timing of each note can be calculated as tempo $\times$ (beats from the start of phrase) . The relative onset time bias is the difference of onset timing between the performance and the score, divided by the total length of the phrase. Namely,

$$ROB = \frac{O_i^{perf} - O_i^{score}}{length(phrase)}$$

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.

Relative loudness: The loudness of a note divided by the maximum loudness in the phrase. Measured by MIDI velocity level.

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

Relative duration: The actual duration of note divided by the total length of the phrase.

$$RD = \frac{D_i^{perf}}{length(phrase)}$$

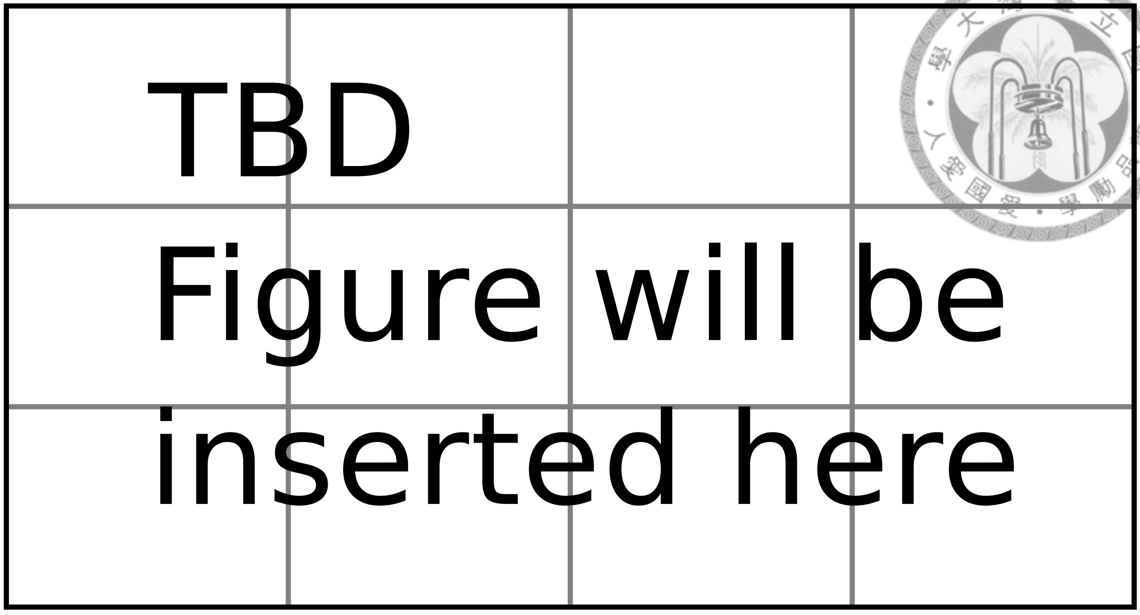

Figure 3.10: Example Performance Features

3.5.3 Normalizing Onset Timing

The relative onset timing bias feature must be normalized to get meaningful result. For example, a phrase is played very fast by the performer, say, the played length is only 70% of the notation. If the first note is aligned, the onset timing bias will grow linearly until the end of the phrase, the last note will have a onset timing bias of about 30% of the phrase. But from the definition of this feature, the timing bias should be roughly less than half of a note's length, and it's mean should be around zero. This 30% value will introduce a relative large value in the training corpus, and thus there may be very large output value from the generation phase. If the model predict that a note should be played early or later for 30% of the phrase length, the melody will be messed up.

There are four possible type of normalization: the first note can either be aligned at onset, or not aligned at all. The last note can be aligned at onset, or aligned at note off event. The incentive for not aligning the first note is that the performer may intend to use an early start or delayed start as an expression, if the first note is aligned by it's onset, the first note in every phrase will have a onset timing bias feature of value zero. In other words, the early/ delayed start expression is lost. But experiments shows that none of these methods can fit

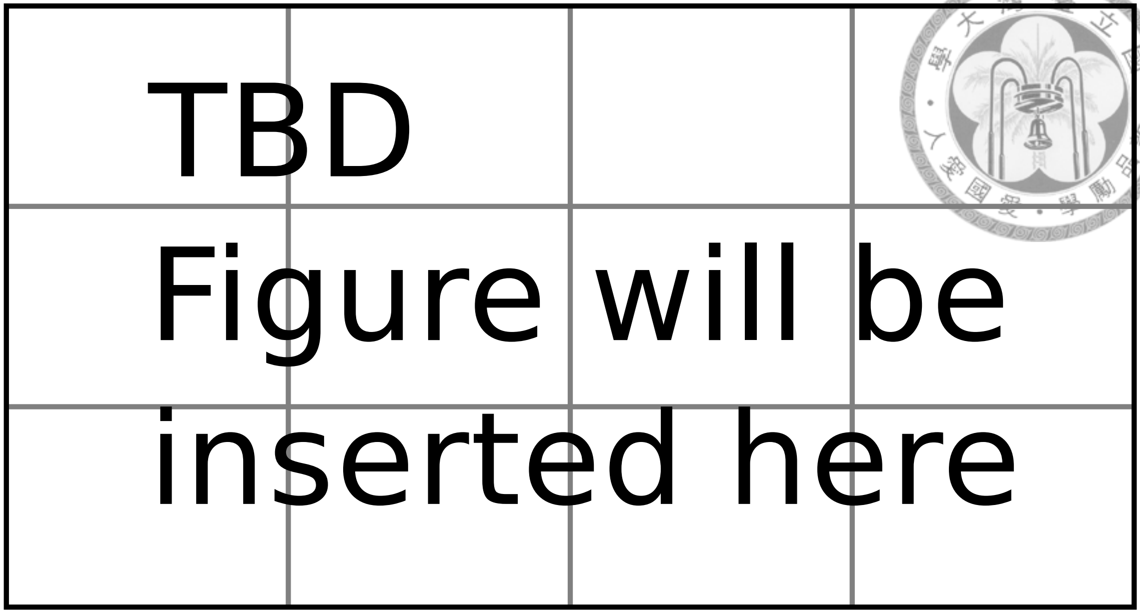

Figure 3.11: Normalization Schemes

all training samples. One may get good result by using one method on part of the corpus, but may result in large bias when applying it to the rest of the corpus. Since none of the methods can fit all, we take a different approach: let the program find the best fitting ratio for normalization. To achieve this goal, we first have to define how "fit" two phrases are, or how to define the distance between phrase. If the onset time of all the notes in a phrase are concatenated into a vector, 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. So the problem becomes how to find a optimal scaling ratio such that the scaled recording has the minimum distance from the score. The Brent's method [**?**] is used to find the optimal ratio. To speed up the optimization and prevent extreme values, we imposed a range of $[initial_guess \times 0.5, initial_guess \times 2]$ to 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 arbitrary chosen, but most of the empirical data suggest is valid most of the time. $\boxed{\text{TODO: conclusion}}$

Chapter 4

Corpus Preparation

Since this research is based on a supervised learning algorithm, a high-quality corpus is essential to our success. In this chapter, we will review the existing corpora, various formats and specifications of a corpus, and how we construct the corpus used in this reasarch.

A expressive performance corpus is a set of performance samples. Each sample consists of a score and a recording. The score is simply the music score being played, which contains notational information; and the recording is a digital or audio recording of a human musician playing the said score. With the two elements, a learning algorithm can learn how the music notation is translated into real performance. These two elements can come in many format, in the next sections we will discuss the pros and cons of some possible candidate.

4.1 Existing Corpora

There exist very few public accessable corpus on the Internet. CrestMusePEDB [?] (PEDB stands for "(Music) Performance Expression Database", created by Japan Science and Technology Agency's CREST program, is one of the example. They claimed to have the following data in their database: PEDB-SCR - score text information, PEDB-DEV - performance deviation data and PEDB-IDX - audio performance credit. The database is free to use via email request. But until the time this thesis is written, the author can not get any response from the database andministrators, so the quality and format of the data is unknown.

Another example is the Magaloff Project [?] conducted by Flossmann et al., a joint effort from a few universities in Austria. Russian pianist Nikita Magaloff recorded all works for solo piano by Frederic Chopin on a Bösendorfer SE, a computer-controlled grand piano. This corpus became the material for many subsequent researches [?]. Flossmann et al. also trained a expressive performance system call YQX [?], which won the 2008 RenCon contest, on this corpus. However, the corpus is not placed in the public domain.

Without any public corpus available, we must record our own corpus, which will be discussed in detail in the next section.

4.2 Score Representation

There are many way to represent a music score in machine-readable format, such as MusicXML [45], LilyPond [46], Finale, Sibelius, ABC, MuseData, and Humdrum. For more information, please check out [47]. For research purpose, proprietary format like Finale and Sibelius is abandoned because of their limited support from open source tools. MusicXML is based on XML (eXtensible Markup Language), it can express most music notations and metadata. LilyPond is a LaTeX-like language for music typesetting. ABC, MuseData and Humdrum are based on ASCII codes and each defines their unique representation for music score. Other formats such as image files (scanned or typesetted by computer) or PDF files are an alternative, but they are not an option for direct computer analysis. MIDI can also be shown as music score in some music notation software, but MIDI is design as control signals for digital music equipments, so it lacks some music notations. Furthermore, the model of low-level music instrument control signals doesn't fit well with music notation, so it is not considered a good way to representation for music score.

In this research, we use MusicXML as the main vehicle for music score, because of the following reasons: first, it covers most music notations and metadata need for this research. Second, it is supported in most music notation software, including the one used in this research -- MuseScore. Finally, the music21 toolbox can convert many other formats

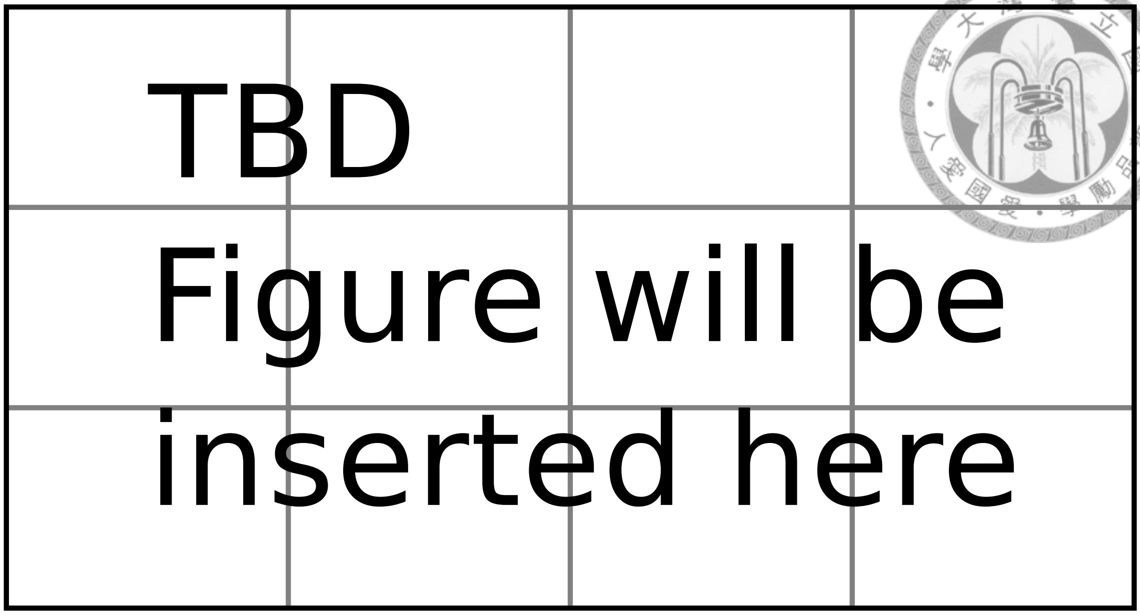

Figure 4.1: Example MusicXML score

into MusicXML without problem.

4.3 Recording Format

A recording of expressive performance can be in MIDI or audio such as PCM WAV. Although audio recording has higher fidelity than MIDI, it takes extra effort to be analyzed by computer. Since onset detection and pitch detection algorithms still can't achieve perfect accuracy, manually labeling each notes required, which will soon become impossible to do when the texture of music become polyphonic. On the other hand, a multichannel MIDI recording can provide exact timing, key pressure, and pitch for each note, but it lacks timber and envelope information, which makes it hard to analyze how the musician use instrument-specific skills. Considering the above arguments, MIDI is used in this research.

A human musician can't play every note exactly on the beat, even if playing along with a metronome. There are two ways to record this behavior: first, record the exact note-on and note-off time, while keeping the tempo fixed; second, keep the notes on the beat, so the

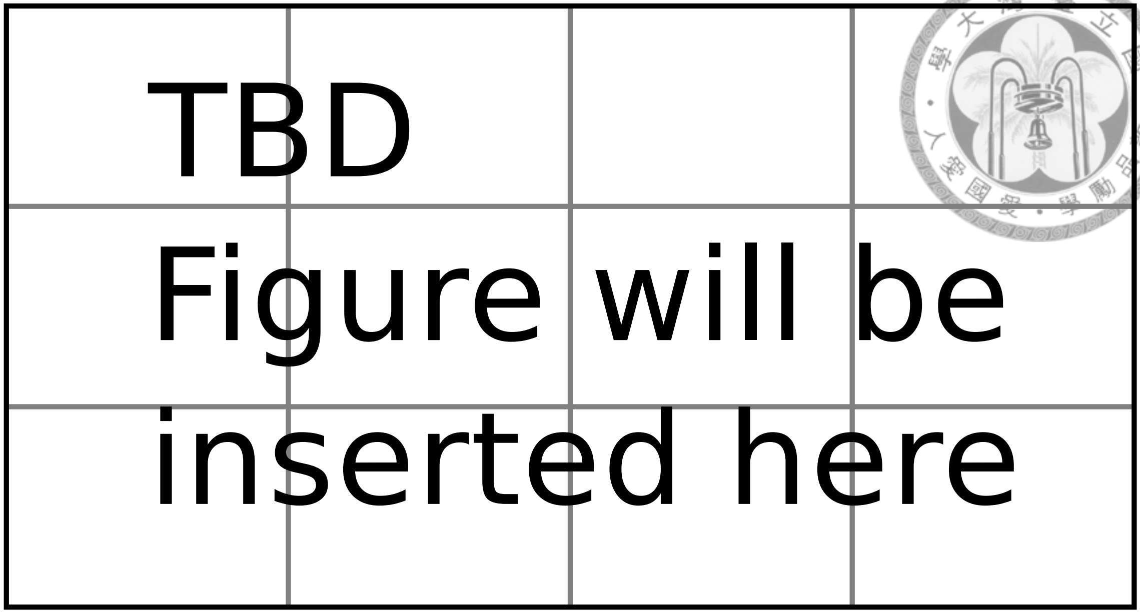

Figure 4.2: Example MusicXML Score (Rendered)

MIDI looks the same as the score. Then insert tempo-change event between each notes, so notes of the same length can be rendered differently because they have different tempo marker. The second method may look smart, because the score and performance can be stored in one MIDI file instead of two, but it would involve complex calculation when linearly scaling the tempo. Since tempos from different samples need to be normalized during feature extraction, the first method is superior than the second.

TODO: discuss how I do the phraseing

4.4 Specfication/Implementation

Because of the limited ability of current implementation of our system, and the limited time and resource we have, we have to impose some constrain on the recordings:

1. All the samples are monophonic. They only contain single line of melody, without chords.
2. A song is divided into phrases by manually labeling.

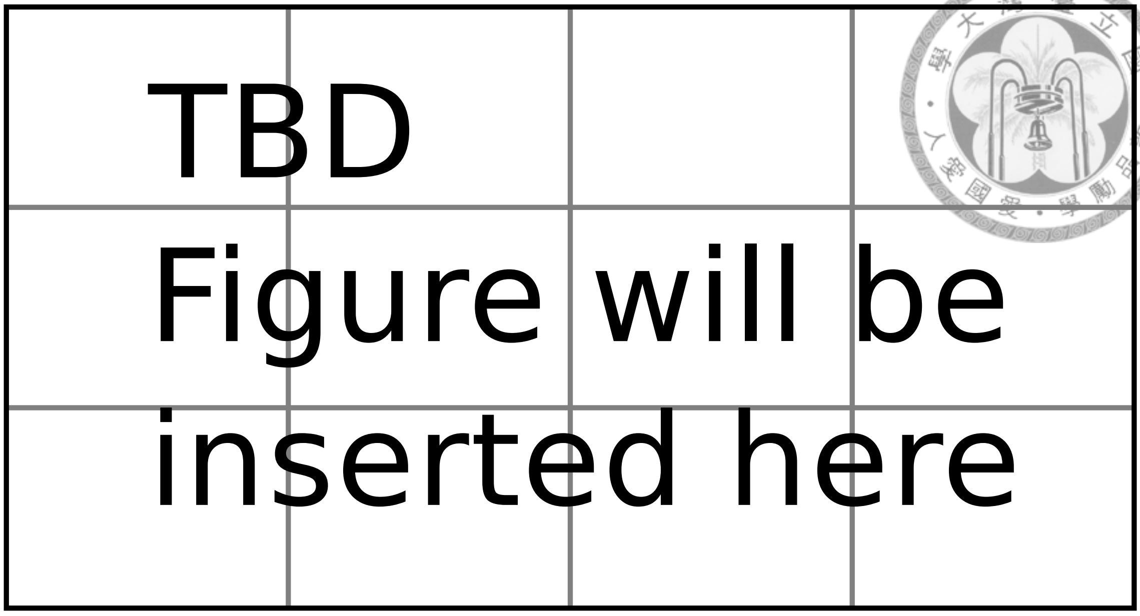

Figure 4.3: Example Recording Compared to Score (Pianoroll)

3. No human error exist in the recording; the score and recording are matched note-by-note.

4. The tempo label in MIDI recordings are the tempo by which the musician played.

For our corpus, we choose Clementi's Sonatina Op. 36. This is because Clementi's Sonatina is a basic repertoire almost every piano student in Asia will learn, so it' s easy to find performers with different skill level to record the corpus. Clementi wrote these sonatinas in classical style, so the skill required to play them can be easily extended to other classical era works like Mozart or Haydn. This fact makes the learned model a very general one, which is good for evaluation.

The digital score used is downloaded from KernScore website [48]. The original format is in Hundrum file format (.krn), then transformed into MusicXML by music21 toolkit. Because this research focus on monophonic melody only, so the chords in the scores are manually reduced to the highest note in the chord, which is usually the most salient melody line. All the other possible errors in the downloaded score is manually corrected to make them ready for printing and computer analysis. We use MuseScore notation editor to view and edit MusicXML; some metadata errors are corrected by editing the MusicXML with text editors .

TODO: early version: touch pad recordings To record the MIDI performances, we used a Yamaha MIDI keyboard with pressure sensitive keys. The Yamaha keyboard was connected to a MIDI-to-USB converter so it acted as a USB MIDI device on our Linux computer. On the Linux computer the Rosegarden Digital Audio Workstation (DAW) is used to record the MIDI. The Rosegarden DAW also generated the metronome sound to help the performer maintain a steady speed. One may argue that the tempo variation is also a part of the expression, but if the performer plays freely, the tempo is hard to determine afterwards, which will make learning scaling the performance in time domain very hard. So the performers are asked to follow the speed of the metronome, but they can apply any level of rubato as they like. The performers were allowed to record multiple times and edit the recording until there were no mistakes. The human error model is not part of this research now, so no mistakes are allowed in the performances to keep the problem simple.

After the MIDIs are recorded, some utility scripts we wrote are employed to make sure each recording is matched note-to-note with the corresponding score; if not, we will manually correct those mistakes. For example, if the pitch was played wrongly, we will correct the pitch but keep the onset, duration and intensity as is. The matched score and MIDI pair are then splat into phrases according to a phrasing 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) of the first note in a phrase in the score. The phrasing is assigned by the researcher now for accuracy, but any phrasing algorithm can be applied.

Discuss the pre-split check, split script, post-split check, corpus preparation scirpt, etc.

TODO: discuss how to specify the phrase if the phrase start at 1/3 position

The coupus used in this research is recorded by five graduate students, with a wide range of piano skill. A total of (TBD) movements of Clementi's Sonatina is recorded, which contains (TBD) phrases, equals to (TBD) notes.

Because of the scope of the research, we didn't use all the information available, here are a few pieces of potentially useful information:

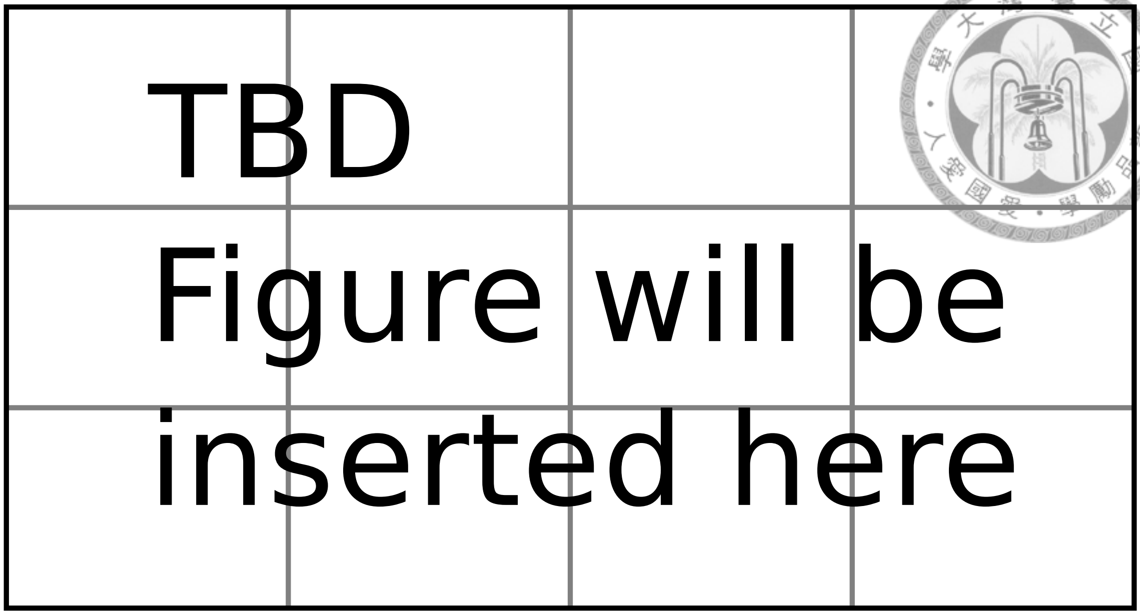

Figure 4.4: Example Phrase File

1. Polyphonic recording can provide information for polyphonic performance
2. The gaps between phrases can be learned to create models for inter-phrase delay.
3. Recordings with human error can make the system play more like human.

TODO: distribution of corpus

Chapter 5

Experiments, Results and Discussion

TODO:general introduction to the experiments

5.1 Parameter Selection

5.1.1 Experiment Design

Structural Support Vector Machine has some parameters that needed to be adjusted. We will leave the others to the defaults and change the SVM C trad-off parameter in this experiment. Since three models are learned for three performance features, we have three parameters to adjust.

[TODO: phrases count] phrases from [TODO:song counts] songs are used for training. Every first, fifth, and tenth phrases from each song is not included in the training sample, but used as testing samples. A three-by-three grid is layed out for three C parameters, each C takes the value of the powers of tenfrom [TODO: Cs] 10^{-5} to 10^{4}, so [TODO: num of experiment] paramenters are tested. Then the result is validated

1. Are all the output samples successfuly generated? (Generation may fail if the performance features are unreasonable, for example, negative onset timeing.)

2. Is the order of the notes preserved? Sometimes the first note is delayed too long and the second note is played too early, so the order is swaped.

3. Are there any extreme parameters that makes the expressive performance unnatural?

The first two criterias are checked by python scripts, the last one is done by manual inspection.

5.1.2 Results and Discussion

TODO:experiment result

5.2 Human-like Performance

TODO:experiment result

5.2.1 Experiment Design

TODO:experiment result

5.2.2 Results and Discussion

TODO:experiment result

5.3 Performer Style Reproduction

TODO:experiment result

5.3.1 Experiment Design

TODO:experiment result

5.3.2 Results and Discussion

TODO:experiment result

5.4 Comparing with State-of-the-Art Works

TODO:experiment result

5.4.1 Experiment Design

TODO:experiment result

5.4.2 Results and Discussion

TODO:experiment result

Chapter 6

Conclusions

TODO:summary TODO:futurework

Bibliography

[1] 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.

[2] 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.

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

[4] “Sibelius.” http://www.avid.com/us/products/sibelius/pc/Play-perform-and-share.

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

[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] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural SVMs," *Machine Learning*, vol. 77, pp. 27--59, May 2009.

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

[43] 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.

[44] T. Joachims, “SVMˆhmm: Sequence Tagging with Structural Support Vector Machines,” 2008.

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

[46] “LilyPond.” http://www.lilypond.org.

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

[48] “KernScores.” http://kern.ccarh.org/.

[49] T. Joachims, T. Finley, and C. Yu, “Cutting-plane training of structural SVMs,” *Machine Learning*, vol. 77, pp. 27--59, May 2009.

[50] 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*, vol. 2000, (Berlin), pp. 348--351, International Computer Music Association, 2000.

Appendix A

Software Tools Used in This Research

Appendix B

Using the Expressive Performance Corpus