Expressive Performance of Polyphonic Piano Music using Hierarchical Structure

Bastiaan J. van der Weij
5922151

Bachelor thesis
Credits: 15 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Prof. dr. ir. R.J.H. Scha

Institute for Language and Logic
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 24th, 2010

Abstract

Both machine learning and rule based techniques have been extensively applied to performance rendering. However, relatively few systems make explicit use of musical structure, combined with machine learning. This paper introduces a performance rendering approach that tries to learn expression exclusively at a structural level. The approach attempts to to be complementary to approaches that learn expression at note level. Musical structure is extracted automatically. A dataset containing performances aligned to scores is then used to learn how structure relates to expression. Finally, the model can be applied to a a new score to produce an expressive performance of the score.

Keywords: performance rendering, musical structure, constituent structure, polyphonic piano music

1 Introduction

In Western art music tradition it is customary that composers write down their compositions in scores. The task of a performer is to some extend to accurately reproduce the score, however, a perfect reproduction of a score generally sounds robotic and unpleasant. What makes a performance appealing is that the performer deviates from the score, altering timing, articulation and loudness, creating an expressive performance.

Given a hypothetical perfect synthesizer, performing music with computers is a trivial task. However *expressively* performing music is not and much research has focussed on this issue. Computers are widely used in music production. Since support for automatic expression is mostly absent or very poor in music editing software, some genres of music have evolved to sound acceptable even without expression. If automatic expression better in this software, computer music production could greatly widen its field of application.

Some music transcription Performance rendering

YQX is state of the art

Rencon

1.1 Motivation

The YQX system, as widmer admitted, tended to sometimes produce nervous sounding changes in expression. He presents two extensions that seek to generate smoother performances. The problem with these extensions is, as widmer admits himself, that the increased smoothness comes at the expense of expressitivity. To counter this, he adds three explicit rules that he uses to postprocess the performances. We think the reason that Widmer stumbled upon this tradeoff between expressiveness and nervousness because the nervousness is inherent to the way performances are generated at note-level. The system simply misses one component of expression, namely structure level expression. (that Widmer had to artificially introduce using rules.) ?

The system presented here will use structure exclusively to generate performances and will completely ignore note level expression. Section **??** will describe the system as a whole. What we mean by structure level expression is clarified in section **??**. The approach for extracting structure is described in section **??**.

The expressiveness of the performances it generates is therefore limited. In section XX we will describe how this system could be integrated with a note level performance rendering system to hopefully produce some variant of YQX that doesn't need explicit rules to produce expressive performances.

1.2 Musical structure

When listening to music, the listener's musical intuition assigns a certain hierarchical structure to the music: Notes make up phrases, phrases make up themes and themes make up a piece. In a performance, this structure may be accentuated in different ways. Accentuation happens at different levels, at note level performers may slow down at the end a phrase or introduce small pauses in the performance at phrase transitions. At constituent level one phrase may be played very loud, fast or staccato, while the next may be played slow, soft and legato.

To formally describe musical structure, we can look at music in a way similar to the way we look at natural language processing(NLP). In this analogy we see a piece of music as a sentence, which consists of constituents that individually can be made of constituents as well. We can recursively subdivide constituents in more constituents until we reach a terminal symbol. In NLP this may be a word, in music, this may be a note. We could represent musical structure as a parse tree. This paradigm corresponds to the intuition that a melody is not simply a sequence of notes but that notes form phrases and we can look at phrases at different levels.

It must be noted that the resulting parse tree can be highly ambiguous and even experienced listeners may not agree on the correct parsetree of a piece. Quite often there may simply be more than one parse tree that makes musical sense. This should not be a problem for a performance rendering system: different expressive interpretations of one piece can be very diverse and still be accepted as sensible interpretations of the piece. As long as the parse tree does make at least some musical sense, a performance rendering system should be able to use it.

Although the YQX does have some notion of structure[1], expression is still only predicted per note. The authors admit that the first simple version of the system "tended to sometimes produce unstable, 'nervous' sounding performances". We see this as a symptom of a note level expression based system.

2 Approach

In this thesis, we propose a structure based performance rendering (SBPR) system. The system presented here ignores note level expression. Instead we will try to predict only constituent level expression. The assumption is that this kind of expression really exists in performances and that is different and independend from note level structure. We think that a constituent level system also corresponds better to how actual human performers play music. A performance redering system that only predicts note level expression would have rather meaningless fluctuations in tempo and dynamics as it does not have a notion of constituent level expression.

[1] One of the note features is distance to nearest point of closure

The system will be similar to YQX in a number of ways, but instead of predicting expression per note, it will predict expression per constituent. Every constituent will be played with consistend expression, the articulation, dynamics and tempo change only at constituent breaks.

We use a corpus that contains performances and corresponding scores of Western classical piano music. Every note in every performance has been associated with the corresponding score note so we can see exactly how every note in the score was played. The performances are of high quality and played by famous pianists. See section **??** for more details on the corpus.

A structural analysis is used to derrive hierarchical structure of every score in the corpus, however, to keep the system simple we will only use this structural analysis to create a segmentation of the score into constituents. After segmentation, four score features, two of which are direct generalizations of YQX's score features, are extracted for each constituent.

So far, we have only used the score. Since we have a segmentation and every score note is associated with a performance note[2] we can also define expression per constituent. Three parameters, analogous to YQX's targets, will be used to describe expression per constituent.

The segmentation, score features and expression parameters are based only on the *melody* of the piece. In this case, melody is defined to be the highest notes in the top staff.

The resulting data is used to train a first order hidden Markov model. The system uses this model to generate performances given a score. To do this, the score is segmented into constituents, score features are extracted for each constituent. Finally viterbi decoding is used to find the sequence of expression parameters that best explains the observed score features.

The simplification of ignoring note level expression is of course unjustified and severly limits the expressiveness of the performances that can be generated. However, if succesful, the resulting performances will clearly demonstrate the phenomenon of structure level expression. A succesful performance rendering system should incorporate both structure and note level expression, section **??** will explore two possibilities for an integration of SBPR and note level performance rendering.

The succes of a SBPR system depends largely on two factors. The ability to generate musically meaningful parse trees of a piece and the ability to accurately characterize the individual constituents and their relations with other constituents in score features. The following section address these issues.

- Mention GTTM
- Give short overview of the system?

3 Method

This section will describe individual components of the system sketched in section **??** in more detail. The structural analysis of is based on the delta framework, which will be described in section **??**. Section 3.2 will discuss how the

[2]In reality, not every score note is associated with a performance note since the pianist may have missed some notes. These notes will be ignored

delta framework is applied to get a segmentation. Sections 2 and 3.4 will describe how the scorefeatures and expression parameters are calculated. Section **??** will describe how a hidden Markov model is trained on our data.

3.1 The Delta Framework

In his Phd thesis **??**, Markwin van der Berg introduces a formal way of parsing music into parsetrees: the delta framework. He relates his work to the work of Lerdahl and Jackendoff **??** but claims to have found a more general approach. Below, I will shortly describe the deltaframework as proposed by Van der Berg.

The delta framework is based on the intuition that differences between notes indicate splits between constituents. The higher the difference, the higher the level of split in the parse tree (where the root note is at the highest level). Van der Berg proposes a delta rule that converts a set of differences, or deltas, between notes into a parsetree following this intuition.

The differences between notes are defined as the difference in value of a certain note feature. More formally, we can look at a piece of music as a sequence of notes, ordered by onset time:

$$M_{ij} = [n_i, n_{i+1}, \cdots, n_j]$$

A set of basic features, Φ, is assigned to each note. These are: **onset**, **pitch**, **loudness** and **release** (called offset by Van den Berg). From these two other features can be derived: **duration** and **virtual duration**. Duration is defined as **release**(n_i) - **onset**(n_i) while virtual duration is defined as **onset**(n_{i+1}) - **onset**(n_i).

The basic note features can be used to define delta functions, for example $\mathbf{\Delta Pitch} = \mathbf{Pitch}(n_i) - \mathbf{Pitch}(n_{i-1})$. In general, a delta function $\delta(i)$ is defined as the difference of two notes in some feature ϕ: $\delta(i) = \phi(n_i) - \phi(n_{i-1})$. We can apply a delta function to every pair of succeeding notes in a sequence to get a list of deltas:

$$\Delta M_{ij} = [\delta(n_{i+1}), \delta(n_{i+2}), \cdots, \delta(n_j)]$$

A recursive rule, called the delta rule can be used to translate a list of deltas into a grouping structure. The deltarule, DeltaRule(M_{ij}), where M_{ij} is the ordered list of notes that is to be analyzed, is defined as follows[3]:

Algorithm 1 The delta rule

```
D_{i+1,j} ← ΔM_{ij}
A ← []
m ← max(D)
for δ in D do
  if δ = m then
    p = index of δ in D
    q = index of the next occurence of m in D or j
    append DeltaRule(M_{pq}) to A
  end if
end for
return A
```

[3]The version here is a slightly reformulated version of Van den Berg's deltarule

It also possible to define higher order delta function (deltas of deltas). Second order deltafunction can be used to differentiate a group of ascending notes from a group of notes that have wildly varying pitches. Van den Berg notes that this generates ambigous grouping structures as for example a second order deltarule needs at least three notes to be specified. For three notes, three different grouping structures are possible: $[[n_1, n_2], n_3]$, $[n_1, n_2, n_3]]$ and $[n_1, [n_2, n_3]]$. Because we want to use a second order deltafunction for segmentation only, we choose to only use one interpretation: $[n_1, n_2], n_3]$. EXPLAIN WHY?

3.2 Segmentation

- We only need a segmentation for now
- Using the deltaframework to define segmentation

3.3 Constituent Features

- We are talking about SCOREFEATURES here, mention this

We can now convert a piece of music into a serie of constituents. These constituents will be used to predict expression so we must be able to charaterize them in a way correlates with the way they are performed. Analogous to YQX we are looking for the *context* of the constituent as well as some description of the constituent itself.

YQX uses a set of three score features: pitch interval, duration ratio and I-R arch. The pitch interval is simply the difference in pitch between the current note and the next note. The duration ratio is the logarithmic ratio of the duration of the current note and the duration of the next note. The I-R arch is is the distance to the nearest point of closure, where closure is calculated from the Implication-Realization analysis. (Too literally?)

We can generalize pitch interval and duration ratio per note to constituent features: *average pitch interval* and *average duration ratio*. Definitions can be found in table 2. Since I-R arch is related to note-level expression it does not generalize well to a constituent level feature.

The two features above provide information about the constituent context: if they are both zero the constituent is apparently similar in average pitch and average duration. At note level there is not much to say about the current note besides the pitch and duration. However at constituent we would also like to say something about the constituent itself. For this purpose *average delta pitch* and *average delta duration* are used. These features say something about the amount change in pitch and the amount of change in note duration.

The complete set of score features consists of two *context features* and two *constituent features*.

(Clearly, expressive markings in the score play a role in how the constituent should be played. If the segmentation is specific enough there will hopefully be at most expressive marking within each constituent.)?

3.4 Expression Parameters

- Use average expression parameters per constituent

Average pitch interval	The difference between the average pitch of the current constituent and the average pitch of the next constituent, zero if the current is the last constituent
Average duration ratio	The logarithmic ratio between the average note duration of the current constituent and the average note duration of the next constituent

Table 1: Context features

Average delta pitch	The average of all absolute pitch intervals within one constituent.
Average delta duration	The average of all absolute differences in duration of succeeding notes within one constituent

Table 2: Constituent Features

3.5 Model

4 Implementation

5 Performance Rendering

- Extract melody
- Extract scorefeatures from melody
- lookup notes in deviation, ignore missing notes
- Extract expressive and non-expressive melody and extract expression features
- Perform notes other than melody notes the same as the nearest melody note

5.1 Dataset and representation

- attack release
- tempo curves

5.2 Dealing with polyphony

5.3 Discretization

Distinction of expressive tempo and attack and release.

6 Evaluation and Results

The proposed system actually requires more training data to do meaningful statistics. However, since expressive interpretation can be

6.1 Subjective Listening

6.2 Correlation

7 Integration with Note Level Performance Rendering

The YQX system defines expressive tempo implicitly by predicting the logarithmic ratio of the IOI in a performance and the IOI in a score. Timing alterations of notes are always defined relative to the base tempo. This does not correspond to the intuition that the *the tempo itself* is altered during the performance and that rhythmic changes should be seen relative to the local tempo. The same applies to the way YQX looks at loudness. This is specified as the logarithmic ratio between the notes loudness and the average loudness of the performance.

An intergration of constituent level expression and note level expression can provide a solution to this problem. We can define expressive tempo relative and dynamics relative to the expression parameters of the constituent

8 Conclusion

[1]

9 Discussion

Unfortunately we do not have acces to the large dataset that YQX uses. The dataset we use is smaller. Since we do not learn per note but per group of notes, the impact of a smaller dataset is even larger. We have discretize into rather large bins for this reason, resulting in cartoonistic performances.

We think we can afford to do this because:...

- Top notes is not always the melody, musical attention
- Bass and harmony shouldn't be played with the same expression as melodynotes
- Dataset size
- Targets for expression level expression

10 Future Work

- Combine deltatrees
- Use loudness and tempo direction instead of averages (requires more data)
- Generalize approach to incorporate hierarchical structure, let structure extend to note level, so note level expression and structurelevel expression become integrated

10.1 Repetition

A notion of repetition and similarity would certainly improve the system. Repetition is a very good indicator of constituent breaks. Repetition and similarity could also be used to improve expressiveness of performances. It is probably telling when a phrase is repeated three times and then slightly altered the fourth time. Although finding similarity and musically significant repetition is a subject of its own the delta framework could help to define repetition arbitrarily of transposition or rhythm. A list of pitch deltas can for example be used to detect repetition independend of transposition and a list of duration deltas can be used to detect repetition of rhythm independend of the notes used.

References

[1] M.J. van den Berg. Aspects of a formal theory of music cognition. 1996.