Creating Expressive Piano Performance Using a Low-dimensional Performance Model

Yupeng Gu
Indiana University
yupgu@indiana.edu

Christopher Raphael
Indiana University
craphael@indiana.edu

ABSTRACT

A model is presented for representing and generating piano performance. The model has far fewer parameters than the number of notes. This model explicitly addresses one of the fundamental characteristics of music performance that different areas in a performance have very different kinds of objectives or strategies that are employed. A graphical model is introduced to represent the evolution of the discrete strategies and tempo and dynamic progression. We design interactive procedures that allow users to modify the model intuitively. An algorithm is described to estimate parameters from partial performances that represent the skeleton of the music. Experiments are presented on the two-piano version of Rhapsody in Blue by George Gershwin.

1. INTRODUCTION

Music performance is an indispensable link in the chain connecting composer and listener. Performers use their skills, passions, expressions and desires to bring the music to life. Musicians have been serving this honored role for centuries. With the rapid development of computer technology, a growing interest appears over the role of the computer in this process. We propose an attempt to structure the problem. Although the idea could be generalized to many types of music, this work concerns itself in the context of Western classical music.

Musical performance usually does not have as many parameters to it as there are notes in a piece. We believe a performance is much lower dimensional than the note-by-note detail level (e.g. the parameters used in MIDI). Most notes are not acting independently, they are guided by higher-level notions or “inner motion” [1, 2]. There are usually strong correlations within a group of notes. This higher-level notion fits how musicians think of and communicate about music.

Many works have been done for modeling piano performances. While some [3–5] focus on providing methods of performance analysis, we want to design a performance model that is aiming for reproducing, modifying and creating expressive performances. Thus, it is not necessary for our model to have an understanding of musical structures that are often described by musicians. Rather, we seek a mathematical model to represent performance with a higher-level notion that can adapt to most situations. The model will consist of discrete states that describe different performance behaviors and continuous variables that describe tempo, timing and dynamic details of the different states.

Applications of such a generative performance models are numerous. One of the motivations of this work is to provide an easier way for more people to perform music, though the model could be applied broadly.

While almost all of us enjoy listening to music, being able to play music is also a very rewarding experience. However, it is not as easy to perform as to listen. To fill the gap between musical ideas and performance, musicians usually spend decades learning, developing, practicing and refining their techniques. Take piano as an example, the technique includes how to hit the correct notes at correct times, how to balance the volume within a group of notes, how to figure out the fingering etc. To make it even harder, a pianist once exaggerated, “It is not considered ready for a pianist to be able to play something right, you need to play everything 10 out of 10 times right to be ready for a performance.” As a result, non-professionals can hardly enjoy performing music that requires certain level of technique to play. This left us singing, humming, describing and roughly playing to express and exchange our musical ideas. These methods are not ideal, but they require much less skill. iPad apps that allow one to play complicated music just by tapping the screen also gain a lot of attention recently including the million sold app “magic piano” [6]. These apps fulfilled people’s needs to play but they don’t allow much expressive control from the individual. Practicing still seems to be the only way towards good expressive performance. But it is fair to say some parts of practicing are quite “mechanical”. It would be great if we can have a performance model that will always generate correct notes and reasonable correlation among them, but still capable of being expressive. We attempt to use such model to ease the process of practicing and hope to bring the joy of performing to more people. Our goal is to create a complete performance based on music ideas in few simple and incomplete reductions played by an user.

As a sub-problem, the question of how to systematically change a digital performance meaningfully can find a possible solution using our performance model. For a very long time, the only way to create an expressive perfor-
performance digitally is having someone perform on an electronic instrument and record it directly. In this case often we may have a decent performance recorded with some parts of the performance unsatisfying. The only thing we could do to improve such performance is to “tweak” parameters at the individual note level and hope some combinations might work. This is clearly an unnatural and unmusical way to modify an expressive performance. It would be better to operate on a higher-level representation of the interpretation such as our proposed performance model that understands the notion of gestures and phrases. For instance, when we modify the timing or dynamic of a single note, some other parameters must compensate to retain a musical sense as specified in the performance model. Score-writing and MIDI-creating program are two of the obvious examples that could benefit from this method.

Such a method for creating performances could be considered as a special example of the expressive rendering problem. The rendering problem comes with growing interest in generating performances that can match the level of trained musicians along with the development of the computer technology. The existing rendering systems are mostly rule-based or case-based. Some systems include extracting and applying rules with parameters [7,8], while others take advantages of statistical model that can learn from a large dataset and generate predictions for new performances based on similar music context from the dataset [9,10].

While building an artificial performer from scratch could be very difficult [7,8], creating expressive performance from a performance model can be an easier task to address. Although it is less ambitious, we think this approach has its own advantages. The first advantage of our performance model is that it is much lower dimensional than the MIDI performances. Hence it is easier to estimate our model parameters than to estimate all the details for every note. The second is that we can use such a performance model in an interactive system that can learn and improve from more specified inputs. Our model is not an answer to the original rendering problem since it may require many explicit information from human input. But it is capable of rendering expressive music without a professional performer.

A musically meaningful model of performance can also be used as a visualization tool. It is often an interesting experience for musicians to listen to a recording of themselves. As a listener, one has a different perspective and judges the performance more objectively. However, listening to a recording is time consuming, and we can only access a small amount of information at one time. Our model can be used to visualize rhythmic interpretation in a discrete way, so musicians can see and explore an entire performance at once. Furthermore, such visualization can also be used to compare different performances, so it will be easier for musicians to compare with other players.

Another possible application of such model is in creating an accompaniment system. A traditional accompaniment system seeks to create a flexible accompaniment to a live soloist that follows the player [11–13]. It could be useful in many music collaboration scenarios. Most Western classical music involves a collection of instruments. So activities such as practice, rehearsal and performance require multiple people to coordinate time and space with one another. A computer music accompanist could provide an alternative solution. A musician would practice with such a computer system when it is difficult to arrange a real rehearsal. This will be a better experience than practicing alone since a more realistic music context is provided. For amateurs and young students, such a system may enable them to play certain music in a complete form which would otherwise be impossible, making the musical experience more accessible and enjoyable. The accompaniment problem can be considered as an estimation problem for the performance model with an incomplete performance (e.g. a single instrument from an ensemble).

We present a mathematical model in section 2. There is a large literature on models that combine discrete state variables with Gaussian variables in fields such as economics, medical science and control engineering [14–17]. These models are known alternately as Markov jump process, hybrid models, state-space models with switching and switching Kalman filter. We think this type of model suits our purpose of creating a model for a piano performance. An interactive process that uses user input to complete the model is presented in section 3. Experiments are demonstrated in section 4.

2. THE MODEL

We only consider piano music in this work. Thus, a piano roll type of representation is most suitable here. A music score is represented as a series of music notes \( r = \{r_1, r_2, ..., r_N\} \). Where \( r_n = \{p_n, b_n, v_n, t_n, d_n, s_n\} \). \( p_n \) indicates the pitch of note \( r_n \), \( b_n \) indicates the music time of note \( r_n \) and is expressed in terms of measure and beat. \( v_n \) is the MIDI velocity of \( r_n \) that describes the volume. It is a integer number between 1 and 127. \( t_n \) denotes the performed onset time of note \( r_n \) and is expressed in terms of seconds. \( d_n \) describes the duration of note \( r_n \) in terms of seconds. \( s_n \) is the discrete state associated with the note. The possible discrete states are described by the set \( \Sigma = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} \) which indicate different tempo behaviors.

Although piano music is often polyphonic and has many voices, we start with a simpler case first. If a part or a voice is monophonic, we introduce a switching Kalman filter model.

One of the most important ideas of our model is the discrete states. Here is a brief explanation of the meaning of the 4 states. \( \alpha_1 \): constant speed – represents the scenario where the performer plays in a steady rhythm; \( \alpha_2 \): slowing down – represents a section of music where the performer gradually slows down; \( \alpha_3 \): speeding up – represents a section of music where the performer gradually speeds up; \( \alpha_4 \): stress – This is a common technique to make an emphasis of a certain note by taking a little extra time before playing that note. The time variables are modeled differently in each different discrete state setup.

The mathematical definition of timing and tempo behaviors in the 4 discrete states are:
\( \alpha_1 \): constant speed

If for a segment where \( s_l = s_{l+1} = \ldots = s_m = \alpha_1 \), we set the tempo \( o \) (measured in seconds per beat) to be constant

\[
t_{k+1} = t_k + (b_{k+1} - b_k) \times o
\]

for \( k = l, \ldots, m \) with an initial \( o \sim N(\mu_o, \sigma_o^2) \) with an unknown mean \( \mu_o \) that is only effective in this segment.

\( \alpha_2 \): slowing down

If for a segment where \( s_l = s_{l+1} = \ldots = s_m = \alpha_2 \), the tempo inherited from last section \( o = (t_l - t_{l-1})/(b_l - b_{l-1}) \) is increasing with a constant unknown rate \( \alpha \sim N(\mu_o, \sigma_o^2) \) that applies only to this segment:

\[
t_{k+1} = t_k + (b_{k+1} - b_k) \times (o + (b_{k+1} - b_l) \times a)
\]

for \( k = l, \ldots, m \).

\( \alpha_3 \): speeding up

If for a segment where \( s_l = s_{l+1} = \ldots = s_m = \alpha_3 \), similar to \( \alpha_2 \), \( o = (t_l - t_{l-1})/(b_l - b_{l-1}) \) is increasing with a constant unknown rate \( \alpha \sim N(\mu_o, \sigma_o^2) \),

\[
t_{k+1} = t_k + (b_{k+1} - b_k) \times (o - (b_{k+1} - b_l) \times a)
\]

for \( k = l, \ldots, m \). \( a \) is also only relevant for this segment.

\( \alpha_4 \): stress

The stress state is modeled to only last for one note and its previous and successor states must both be state \( \alpha_1 \). So if \( s_{m-1} = \alpha_1, s_m = \alpha_4, s_{m+1} = \alpha_1, o = (t_{m-1} - t_{m-2})/(b_{m-1} - b_{m-2}) \),

\[
t_m = t_{m-1} + (b_m - b_{m+1}) \times o + c
\]

\[
t_{m+1} = t_m + (b_{m+1} - b_m) \times o
\]

\( c \sim N(\mu_c, \sigma_c^2) \) is a variable relevant only for this note.

The sequence of the discrete state is modeled as a Markov chain. Figure 1 shows the Markov model.

One of the main reasons we choose these assumptions is that the state space of switching Kalman filter grows exponentially with time [18]. Even with approximation schemes, we want the number of possible state transitions in our model to be as small as possible. We think the first three states with enough transitions that can cycle through them are capable of capturing most tempo behaviors. We add the 4th state to have the ability to “remember” an intended tempo after a single note tempo variation. We also think these assumptions are suitable for capturing local tempo behavior changes that are within few notes. For large scale tempo behaviors such as an accelerating over couple measures, our model can explain them with a combination of several state changes.

The directed acyclic graph (DAG) of the graphical model is represented in figure 2. The model has both discrete and Gaussian variables. For every fixed configuration of the discrete variables, the continuous variable have a multivariate Gaussian distribution. Thus, the \( s_1, s_2, \ldots, s_N, t_1, \ldots, t_N \) collectively have a conditional Gaussian distribution [19].

If a section of music contain polyphonic elements, we can always categorize them into one of the following two types. The first type is “chord”. A chord here means a group of notes that should be played at the same time. We model the time variables of such event in a straightforward way – the time variables in a chord are exactly the same.

The second type is “multiple voices”. There are occasionally notes that share a same music time belong to different voices (or even different instruments). As a result, we shouldn’t assume they will be played at the same time. In the case of more than one voice, we choose one of the voices as the leading voice (usually the melody), which operates exactly the same as the switching Kalman filter in the monophonic case we introduced earlier. As for the other voices, they are also similar switching Kalman filters but subjected to constraint that the timing of certain notes have to be the same with some notes in the leading voice. These notes are called “anchor points”, the parameters of the voices are independent if the timing of these anchor points is given.

Figure 3 is an example of a section of music with many chords and two voices.
Figure 3. An excerpt and its DAG describing the dependency structure of the time variables in polyphonic case. Circles represent discrete variables while squares represent continuous variables. Although the arrows from discrete variables to continuous variables are omitted in the graph for the sake of clarity, they are present in the actual model.

Figure 3 shows the dependency structure if we consider the 1st and 5th groups (the 1st and 5th chords in bass clef) of notes that share same music times to be the “anchor points” while the 2nd-4th groups of notes are considered to be in two voices.

At the first glance, specifying the “anchor points” could be a complicated problem itself and require a lot of manual labor. However, we will introduce an interactive process to choose them semi-automatically in the next section.

Now let’s introduce the model for dynamic and duration. Since we set them to be exactly the same, only dynamic model is discussed here. The modeling assumption for these two variables can be summarized as “if something similar happened before, it will most likely act the same”. Here is an example: figure 4 shows a music excerpt and one of its possible dependency structure. We will introduce how to construct the dependency structure in the next section too.

The variables without a predecessor such as $d_{a1}, d_{a2}, d_{a3}, d_{b1}, d_{c1}$ are modeled as independent. In this example $d_{ak} = d_{a1} + (d_{ak} - d_{a1})$ for $k = 2, 3, 4$; $d_{ck} = d_{c1} + (d_{ck} - d_{c1})$ for $k = 2, 3, 4$ and so on. This model assumes that the balance within a chord is fixed for this excerpt.

Here is another example: figure 5 shows a music excerpt and one of its possible dependency structure. We will introduce how to construct the dependency structure in the next section too.

Again, the variables without a predecessor such as $d_{a1}, d_{a2}, d_{a3}, d_{b1}, d_{c1}$ are modeled as independent. In this example $d_{ak} = d_{a1} + (d_{ak} - d_{a1})$ for $k = 2, 3, 4$; $d_{ck} = d_{c1} + (d_{ck} - d_{c1})$ for $k = 2, 3, 4$ and so on. This model assumes that similar short sections should have same relative dynamics.

3. SYNTHESIZE PERFORMANCE

In the previous section, there is a very important missing part that we are going to introduce now – how to construct the dependency structures for timing, dynamic models. These structures are not specified by hand. Rather, besides the hidden parameters, we use an algorithm to retrieve the structures from user input of incomplete performances that represent the skeleton of the music.

Let’s look at an example of the idea first, figure 6 shows the excerpt we want to play.

Even a trained pianist needs quite some time to play this excerpt fluently. But the music ideas behind this excerpt is not so complicated. Here are two expected reductions for our model from an user: 1) a theme as shown in figure 7; 2) a rhythm voice on the left hand as shown in figure 8.

Anyone with a little piano knowledge can play the two parts shown in figure 7, 8 and play them expressively. Our goal is to complete the performance in figure 6 – specify the model and estimate the parameters – based on these
two reductions. Here are the procedures:

1) Data preparation
We use MIDI file exported from a score writing program as our starting point. Then we will have an user play the reductions that are representative in terms of musical ideas such as those shown in figure 7, 8.

2) Match the reductions to the score
This is a performance alignment problem – given a performance and a score, we want to find out what and when notes are being played. This can be achieved using Hidden Markov Model (HMM) type of approach. [20] provided an algorithm based on HMM to match a performance with a large portion of missing notes (e.g. a reduction like we have here) to the score.

3) Construct the timing model
The tasks of constructing the timing model are how to divide voices and specifying the “anchor points”. Our assumptions are: if there are multiple reductions being played for a same excerpt and they have different rhythm structures, they are considered to belong to different voices (e.g. if music shown in figure 7 and figure 8 are played separately, they are considered to be in two voices); the notes that are being played in multiple reductions are considered to be the “anchor points” where voices meet (e.g. the low notes in three chords in figure 7); the notes that are never played in any reduction belong to the voice with closest pitch. Although the last one is a naive assumption, it has a chance of work because the hand of human can only cover a small range of pitches.

4) Construct the dynamic model
The tasks of constructing the dynamic model are specifying the independent notes and creating the dependency structure. Our assumptions are: for notes that are being played in any of the reductions, their dynamics are considered to be independent; for a group of consecutive notes that are never being played in any of the reductions, they depend on the nearest note played previously in the same voice parsed in the last procedure; for this group of notes, we search for the nearest played group of notes with the exact same rhythm structure. If found, dependency relationships will be established as well. Figure 4 shows the results of applying these assumptions to the first 4 chords of reduction in figure 7.

5) Parse the discrete variables for timing model
After we have the voices divided, each voice can be considered a monophonic excerpt and will be modeled as a switching Kalman filter introduced in section 2. Let $T_1, T_2, \ldots, T_N$ be the observed timings from reductions. We define the data model for timing variables $T_n = t_n + \epsilon_n$ where $\epsilon_n \sim N(\mu_n, \sigma_n^2)$. We can address the problem as finding the most likely configuration of both discrete variables and continuous variables $s_1, s_N, t_1, \ldots, t_N$.

Then we can use the method known as the “beam search” to compute the discrete variables that guide the timing variables [21].

Note:
At this point, we finally have the complete model. For real world uses, we can manually refine the model structure and improve the discrete variables without too much effort to make the model more “realistic” since the model provides a lower dimensional structure that relates to how musicians talk about music. But for experimental purposes, we’ll proceed with algorithmically generated model.

6) Estimating & computing the parameters
With the fixed discrete variables $s_1, \ldots, s_N$ we obtained in previous procedure, estimating $t_1, \ldots, t_N$ is a standard smoothing problem for Kalman filter: $\hat{t}_1, \ldots, \hat{t}_N = \arg\max_{t_1, \ldots, t_N} P(t_1, \ldots, t_N|s_1, \ldots, s_N, T_1, \ldots, T_n)$. We can use the recursive algorithm of Kalman filter to compute the timing variables [22].

For dynamics, we treat $v = (v_1, v_2, \ldots, v_N)$ as a random vector where we observe a subset of the variables $\{v_k_1, v_k_2, \ldots\}$. Then we can compute the rest using the dynamic model we introduced in section 2.

4. EXPERIMENT
We choose the two piano version of Rhapsody in Blue by American composer George Gershwin as our experiment material. The MIDI score is exported from a score-written program. In general, the data can be collected from any reproducing piano or digital piano. We use a high quality hybrid piano AvantGrand N2 made by YAMAHA. The reason we choose such an instrument is to ensure that the reproduction will be exactly as performed. The piano keyboard is the same as YAMAHA C3 grand piano which provides the same touch of a real grand piano.

For demonstration of how the model works, we choose 3 excerpts from the piece and have an user play some reductions of these excerpts. The reductions represent the user’s idea of the model structure which will be captured using the methods described in section 3. There could be multiple performances of a same reduction but we let the user pick the best one. These examples can be heard at https://dl.dropbox.com/u/6449856/Web/smc2013.html.
The following table shows the number of timing and dynamic parameters in MIDI file, the reductions played by an user and the result of our model for the three excerpts Ex1, Ex2 and Ex3.

<table>
<thead>
<tr>
<th># of timing Parameters</th>
<th>Ex1</th>
<th>Ex2</th>
<th>Ex3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDI</td>
<td>244</td>
<td>446</td>
<td>240</td>
</tr>
<tr>
<td>Played reductions</td>
<td>93</td>
<td>73</td>
<td>53</td>
</tr>
<tr>
<td>Our model</td>
<td>20</td>
<td>22</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 1. The comparison of number of timing parameters in MIDI file, the reductions and the result of our model

<table>
<thead>
<tr>
<th># of dynamic Parameters</th>
<th>Ex1</th>
<th>Ex2</th>
<th>Ex3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDI</td>
<td>244</td>
<td>446</td>
<td>240</td>
</tr>
<tr>
<td>Played reductions</td>
<td>99</td>
<td>97</td>
<td>60</td>
</tr>
<tr>
<td>Our model</td>
<td>99</td>
<td>97</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 2. The comparison of number of dynamic parameters in MIDI file, the reductions and the result of our model

The examples show that with far fewer parameters than the MIDI file as shown in the table, we still capture much expressiveness from human input and use them to render a complete expressive performance accordingly. Our timing model is capable of reducing the # of parameters in a performance to 10% - 20% of those in MIDI files. With further development, we are expecting a more advanced dynamic model that can achieve a similar percentage.

5. DISCUSSION

This performance model definitely needs more development. There are many assumptions made because of their simplicity. It is also not the most intelligent model either since it requires a lot of human-computer interaction. However, this model makes an attempt to capture the low-dimensional nature of music performances and creates a framework for reproducing and synthesizing expressive performance. This model explicitly addresses the way that different areas in music performance have very different kinds of objectives or strategies that are employed. This is a fundamental characteristic of music performance that has not been developed much. We try to make mathematical scientific sense out of this important aspect of performance. The model along with procedures introduced in section 3 provide a computer system that allows anyone with some basic piano skills to play very technical pieces such as the Rhapsody in Blue with their own music ideas. The model also offers a platform for systematically changing a performance meaningfully and intuitively.

This is our first step towards a good performance model. We believe there are many aspects that can be researched and improved. The discrete states for timing model are clearly something we can work on to make it better. Tempo can progress quadratically instead of linearly. We are also developing more sophisticated model for dynamics which now is almost solely rely on human input. There are much more dynamic relations among notes that wait for us to explore.

Another possible follow-up for this model is accompaniment system. Our proposed model can be considered as an offline version of accompaniment system. Since the goal of an accompaniment system is essentially generating a complete musical performance with partial performance data that is played by one soloist. We hope with proper modifications, an online version of our model can be used as an accompaniment system and play concerto type of music in real time.

This model also opens a new way of approaching expressive rendering problem. With this model, what we need for constructing an expressive performance is the different areas and few key numbers that represent the performing strategy in those areas. Hence we have far fewer parameters to estimate. But of course for fine detail of performance, the model may need to be more sophisticated than simple linear ones.

We look forward to presenting more generally useful applications of the performance model framework as it develops.

6. REFERENCES


