The Creation of a Music-Driven Digital Violinist

Jun Yin[+], Ankur Dhanik[#], David Hsu[+], Ye Wang[+]
National University of Singapore, Singapore 117543
+: {yinjun, dyhsu, wangye} @comp.nus.edu.sg #: g0203706@nus.edu.sg

ABSTRACT

This paper describes an initial attempt on a music-driven digital violinist (MDV) system, which automatically generates animation of a violinist based on violin music. MDV first analyzes the input audio signal and transcribes it into music notes. Next it uses the notes to synthesize the animated video of a violinist. Tests on the prototype system show that it achieves adequate visual realism and near real-time performance.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-based Systems**] Signal Processing Systems

H.5.5 [**Sound and Music Computing**] Signal Analysis, Synthesis and Processing, Systems

General Terms

Algorithms, Design, Experimentation.

Keywords

Music transcription, animation, inverse kinematics

1. INTRODUCTION

The effect of music performances depends on both the sound produced and the action of performers. Often the sound or the audio aspect of a music performance is readily available, long after the performance, as recording or broadcasting. The action of performers or the visual aspect of a performance is less so, because the video is inherently more expensive to store and transmit than the audio. As it has been widely acknowledged that computational power increases much faster than network bandwidth, we may ask: is it possible to recreate the action of performers from the audio recording of a performance?

In this work, we take the first step towards this direction. Our goal is to create a music-driven digital violinist (MDV) system, which takes as input a piece of solo violin music and outputs the animation of a violinist synchronized with the music. We have chosen violin because of the versatility of violin music and the skillful action required of violinists. The idea of combining music and animation is not new and has been applied effectively before (see, *e.g.*, [6]) in interesting, but simpler scenarios.

Systems like MDV have many potential applications, from the immediate to the more ambitious. At the minimum, it can replace the random geometric patterns for sound visualization on many media players and serve as more meaningful and potentially more interesting visualization. MDV can also be a useful tool in music education as an interactive demonstrator for beginners learning to play music instruments. Although audio recordings of violin music are widely available, learners need the visual feedback to *see* how to play the music on the recordings. With the help of MDV, learners can observe the correct playing from different viewing angles interactively, an important advantage over the conventional recorded video. They can do this in their own time, pace, and chosen location, reinforcing the knowledge from teachers' lessons. Successful MDV systems may also help create exciting new multimedia contents by adding visual feedback based on audio, *e.g.*, imagining recreating grandmasters' performances from past audio recordings.

We have implemented a prototype MDV system for proof of concept (Figure 1*b*). Although the prototype shows a violinist playing in the Indian style, the approach is general and can be applied to other major string instruments, including western violin and cello. This paper presents our preliminary results.

(*a*) (*b*)

Figure 1: A real and an animated digital violinist.

2. SYSTEM OVERVIEW

Our system takes as input the audio recording of a piece of violin music and outputs the animated video of a violinist synchronized with the audio. We propose a two-step solution. First we analyze the input audio signal and transcribe it into music notes. Next we use the notes to synthesize the 3-D animation of a violinist.

Our system has a simple architecture with two modules that perform violin music transcription and animation (Figure 2). The transcription module converts the audio signal into a note table, which is a list of note entries. Each entry contains information on the pitch, loudness, onset, and duration of a note. Optionally an entry may also contain information on playing styles, such as *vibrato* and *spiccato*. The note table can be viewed as an annotated music score. As a design goal, the information in the note table should be rich enough to describe not only the notes played but also the individual playing style of a violinist. The note table output by the transcription module is then fed into the animation module to produce the animated video. Assuming that the transcription is accurate, the audio and video can be synchronized by playing the notes in the note table successively.

MM'04, October 10–16, 2004, New York, New York, USA.

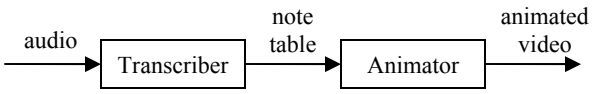


Figure 2: System diagram.

Motivated by the intended applications, one of our design goals is to achieve real-time or near real-time performance. This leads us to choose simple and efficient methods for transcription and animation. Clearly more sophisticated alternatives exist, but they are often computational intensive, and it is more difficult to achieve real-time performance.

3. VIOLIN TRANSCRIPTION

The violin transcription module automatically generates note information from a solo violin wave file. At the basic level, note information contains the onset, duration, pitch and loudness of notes, which are used to control the animation module. Piszczalski and Galler created the first automatic single-instrument transcription system [8]. Subsequent research by many researchers has made considerable progress in this field. Our violin transcription method is similar to that in [7]. It goes through spectrogram creation, pitch and loudness estimation, onset detection, and note information generation. These four stages are described in the subsections below. Compared with the previous work, we have significantly improved the speed of transcription by exploiting the characteristics of violin music.

3.1 Spectrogram Creation

The input wave file has the conventional CD format and is re-sampled at the rate of 22 kHz for reduced computation. The analysis window length is 4096 samples (186 ms) with a shift of 512 samples (23 ms) for continuous analysis. Hanning windowed FFT is used to create the amplitude spectrum of each frame followed by sub-band processing.

We assume that G3 (~196 Hz) is the lowest note and G6 (~1568 Hz) is the highest note the violin can play. We use this knowledge to drastically reduce the analysis frequency bandwidth thus reducing computational complexity. Our sub-band structure follows the musical note structure. That is, the central frequency of each sub-band equals the musical notes (G3, G#3, A3, A#3, etc.). The bandwidth of each sub-band equals the corresponding semitone. The sub-band energy spectrum is computed from amplitude spectrum using the following formula:

$$Z[i] = \sum_{k=LB(i)}^{UB(i)} (Y[k])^2 \quad i = 1..37 \quad \text{(G3..G6)}$$

where *i* is the subband index, Z is the sub-band energy (Z[1] is the energy of G3, Z[2] is the energy of G#3, etc), Y is the amplitude spectrum, LB(k) and UB(k) are lower bound and upper bound of a sub-band.

The output of this sub-module is a sub-band energy spectrogram shown in Figure 3.

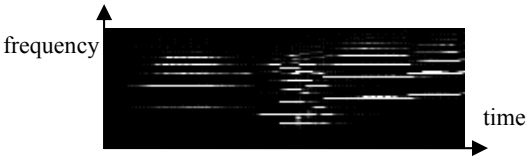


Figure 3: Sub-band energy spectrogram.

3.2 Pitch and Loudness Estimation

We have found out from experiments that the harmonic structure of a violin sound is unstable over time and most energy of a violin sound is concentrated at the fundamental and first 5 harmonics, which lie at 12, 19, 24, 28 and 31 semitones away from the fundamental, respectively. This observation enables us to design a simple yet effective pitch detector. Energy values of all possible pitches between G3 and G6 are computed using the following formula, which is simplified version of a similar on in [7]:

$$E[i] = Z[i] + Z[i+12] + Z[i+19] + Z[i+24] + Z[i+28] + Z[i+31] \quad i = 1..37$$

The pitch with the highest energy is assumed to be the dominant pitch of the frame. The logarithm of the energy is estimated as the loudness:

$$p = \arg\max E[i] \quad i = 1..37 \quad l = \log E[p]$$

Although we assume that the signal is monophonic in our current implementation, this pitch estimation algorithm can be extended to deal with polyphonic violin signal. If the pitch with the next highest energy is above a threshold, it can be considered as the secondary dominant pitch of the frame:

$$p_2 = \arg\max E[i] \quad p_2 \neq p \text{ and } E[p_2] \geq \text{threshold} \quad i = 1..37$$

The output of this sub-module is a pitch-loudness-gram, shown in Figure 4.

Figure 4: Pitch-loudness-gram.

3.3 Onset Detection

Onsets are detected in each sub-band (from G3 to G6) over time. When any of the two following criteria is satisfied, the location of this sub-band and this frame is considered an onset:

- Sub-band loudness is above a pre-determined threshold

  $$F[j] = \begin{cases} 1 & L[j] \geq \text{threshold1} \\ 0 & L[j] < \text{threshold1} \end{cases}$$

  if $F[j] - F[j-1] = 1$, j is the onset frame

- The derivative of the sub-band energy is over than a pre-determined threshold

  if $L[j] - L[j-1] \geq \text{threshold2}$, j is the onset frame

The output of this sub-module is the onset locations shown in Figure 5. The two thresholds are determined experimentally.

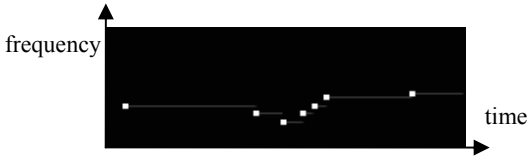


Figure 5: Onset locations.

3.4 Note Information Generation

The note information is generated based on the output of onset detector described in the previous subsection. Each onset is a

potential note. The sub-band with the highest energy is considered the note's pitch. The detected onset is considered the note's starting time. Our algorithm then tracks the subsequent frames, until the sub-band loudness decays below a predetermined threshold. This allows the estimation of the note's duration. The average loudness along a note is computed as the note's loudness.

The output of this sub-module is the note table, shown in Table 1. This table is fed to the animation module as the input.

Onset	Duration	Pitch	Loudness
42	113	24	0.79877
160	23	20	1.2373
182	22	15	2.8975
…	…	…	…

Table 1: An example note table.

3.5 Performance Evaluation

Transcription errors result mainly from pitch estimation and onset detection. We tested the accuracy of our violin transcription module with four violin music samples and compared it with that of commercial software. The accuracy is measured by the number of correctly detected notes divided by the total number of detected notes. A detected note is deemed correct if it has the same pitch and onset as the one in the music. Loudness is not considered in determining accuracy. Table 2 shows that our transcription method performs much better. Although the tested commercial software systems do not reveal what methods they use, we suspect that our method has two main advantages. First it specifically takes advantage of the fact that violin music is monophonic most of the time, while the commercial software is designed for general polyphonic music. Second it employs adaptive threshold for pitch estimation. To understand these issues better, we plan to implement and compare with other strong transcription methods in the literature.

Song	Our System	Amazing MIDI v1.70 [9]	intelliScore v5.1 [10]
Swan	89%	14%	48%
Thai	90%	23%	34%
ViolinM1	97%	11%	19%
ViolinM2	96%	41%	27%
Average	94%	23%	29%

Table 2: Transcription accuracy of our transcription module versus two commercial software systems.

4. ANIMATING A VIOLINIST

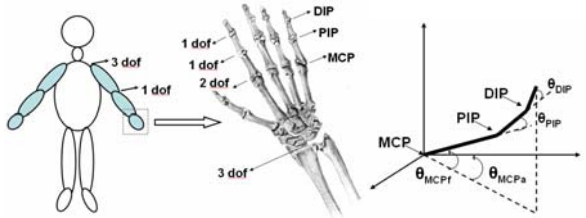
While playing, violinists execute intricate motions of fingers, hands, and arms. The left hand shifts among different positions along the neck of the violin; the fingers press the strings and control the pitch of the sound produced (Figure 1). Simultaneously the right arm moves the bow back and forth on the strings to produce the sound. Reproducing this coordinated motion realistically is a challenging task, due to the complexity of human anatomy. From the shoulder to the tips of fingers, each limb consists of many joints with nearly 30 degrees of freedom (dofs), some of which are interdependent. See Figure 6. To play the desired music notes, all the joints have to move in a coordinated fashion to place the fingertips at the intended positions on the strings.

The difficulty of synthesizing human motion has attracted strong interest (*e.g.*, the references in [3]). There is also recent work on animating guitar and violin playing. Observing the complexity of motion in playing music instruments, previous work attempts to capture this complexity using sophisticated models such as neural networks [4] and minimization of a global cost function [2].

In contrast, we believe that despite the intricacy required of violin playing, the range of motion needed is fairly restricted. A simple procedural model can capture the motion effectively without much sacrifice in visual realism. The key observation is that to a reasonable approximation, every music note has a unique posture of the left hand and arm for producing the note. So we can pre-compute a database of hand postures for all the notes and interpolate between the pre-computed postures to synthesize the continuous motion. The bowing motion of the right arm is synthesized in a similar way, but the arm postures are computed on the fly rather than pre-computed and stored in a database. We now give details on how to compute natural-looking hand-arm postures via inverse kinematics (IK).

4.1 Fingering

In violin playing, fingering means pressing the strings with left fingers to control the music note produced. To do this, a violinist first moves the left arm to position the wrist at one of the seven positions along the neck of the violin and then extends a finger, called the active finger, to place its tip at a chosen position on one of the strings. Following standard practice, we use a skeletal model of the limb, consisting of the arm, the hand, and the fingers. The limb is represented as rigid links connected with flexible joints (Figure 6). By changing the joint angles, we control the postures of the limb.



(*a*) (*b*) (*c*)

Figure 6: Modeling the dofs in the arm and the hand.

Given a music note to be played, we can determine the string for playing the note, the active finger, and the position of the active fingertip on the string. The problem is then to compute the joint angles to achieve the posture that places the fingertip at the required position to play the given note. Our solution decouples the limb into two parts, the arm and the hand, and computes the joint angles for the two parts separately. First we compute the joint angles for the arm (Figure 6*a*) to place the wrist at the desired position p_w, assuming that the shoulder is fixed. Next, given p_w, we compute the joint angles for the wrist and the active finger (Figure 6*b*-6*c*) to place the fingertip at the desired position p_f. Both steps can be accomplished by solving the IK. Although solving IK efficiently with many dofs is challenging, it has been studied extensively in robotics [1]. Due to the limited space, we omit the details of our IK methods, but highlight three issues that differentiate our problem from that of similar problems in robotics:

Natural postures Since the limb contains a large number of dofs, there are multiple postures to achieve a given wrist or fingertip position. In other words, there are multiple IK solutions. However, most of these postures do not appear natural for violin playing. To address this issue, we performed empirical measurements to determine a reference posture θ_r and choose an IK solution θ as close to the reference posture as possible. More precisely, we solve the following minimization problem:

$$\min_{\theta} \|\theta - \theta_r\|^2 \quad \text{subject to} \quad f(\theta) = x$$

where x is the desired write or fingertip position and f represents the mapping determined by the structure of the arm or finger. The minimization ensures that IK solutions lead to postures natural for violin playing.

Sympathetic motion Some finger joints are interdependent, meaning that there is a relationship between different joint angles. This phenomenon is called sympathetic motion and is particularly noticeable during violin playing: the passive fingers, the fingers not controlling the notes being played, usually assume postures similar to that of the active finger. This helps us to determine the joint angles for the passive fingers. Once we obtain the joint angles for the active finger from the IK procedure, we set the joint angles of the passive fingers to similar values after a scaling. The joint interdependency relationship used in our implementation is shown in Figure 7. It is based on results from physiological studies in the literature [5].

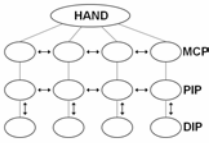


Figure 7: Joint interdependency of four fingers, excluding the thumb. Each oval indicates a joint. A double arrow between ovals indicates joint interdependency.

Reaching multiple concurrent fingertip positions Sometimes a violinist does not release the active finger for the previous note before pressing down the active finger for the current note. In this case, we must solve the IK to place both fingertips at the required positions. This makes the problem more difficult because of the increased number of dofs. Our current solution is to prioritize the two fingers. We first solve for the joint angles of the current active finger to place its tip as close to the required position as possible and then compute the joints angles for the other finger. This is natural, because only the current active finger controls the pitch of the note played and is more important.

We apply the above IK procedure and compute a database of postures, one for every possible note on the violin. Although the database creation may be time consuming, it is done in a pre-computation stage, when time is not critical. Using the posture database, the animation module reads the notes from the note table one by one, looks up the posture from the database, and interpolates the joint angles of postures corresponding to successive notes to generate the continuous motion.

4.2 Bowing

The bowing motion of the right arm is created similarly using IK and interpolation. For simplicity, we attach the right hand rigidly to the bow and use the joints of the right arm to move the wrist along one of four possible lines, each corresponding to the transverse movement of the bow along one of the four strings. Here the IK solution involves only 4 dofs of the right arm and is computed on the fly.

5. SYSTEM IMPLEMENTATION

Our prototype system implements the violin transcription module in Matlab and the animation module in C++. OpenGL is used for 3-D graphic rendering. The transcription module takes a single wave file as input and feeds the computed note table into the animation module. The animation module generates the animation of the violinist and, at the same time, plays the wave file, in a synchronized manner.

In the current implementation, the transcription module takes roughly 20 seconds to transcribe one minute of violin music on a Pentium IV PC. The animation module can generate animation at the rate of 25 frames/second. These test results indicate that our approach is suitable for real-time applications. We plan to convert the Matlab code for transcription into C++ and integrate with the animation module to further improve performance.

6. DISCUSSION

We have presented preliminary results on the MDV system. Using a two-step approach consisting of music transcription and animation, the system generates realistic animation synchronized to the audio with good performance.

Our work is just beginning. We are currently exploring several directions to improve the system. An immediate step is to test our transcription method on polyphonic violin music. Playing polyphonic notes also requires us to expand the posture database to include postures with two active fingers. We would also like to develop specialized methods for detecting playing styles such as *vibrato*. In addition, our assumption that every note corresponds to a unique hand-arm posture is not exactly true for advanced violin playing. Every note can be played with a small finite number of postures. The choice of postures depends on the nearby notes. This is an interesting problem that has been studied in earlier work [4]. Finally on the implementation side, we plan to integrate the transcription and the animation modules to build a real-time application.

7. REFERENCES

[1] J. Craig. Introduction to Robotics: Mechanics and Control. Addison-Wesley, 1989.

[2] G. ElKoura, K. Singh. Handrix: Animating the Human Hand, *ACM Symp. Computer Animation,* pp. 110-119, 2003.

[3] P. Faloutsos, M. van de Panne, D. Terzopoulos. Composable Controllers for Physics-based Character Animation. *SIGGRAPH Conference Proceedings*, pp. 251-260, 2001.

[4] J. Kim, F. Cordier, N.M. Thalmann. Neural Networks Based Violinist's Hand Animation. *Computer Graphics International*, pp. 37-41, 2000.

[5] J. Lee and T. Kunii. Model-based Analysis of Hand posture. *IEEE Computer Graphics and Applications*, 15(5), 77-86, 1995

[6] W. Lytle. More Bells and Whistles. *ACM SIGGRAPH 2001 Electronic Theatre*, ACM SIGGRAPH Computer Animation Festival, 2001.

[7] P. M. Martins, J. S. Ferreira. PCM to MIDI Transposition. *Audio Engineering Society 112th Convention Paper*, May 2002.

[8] M. Piszczalski, B.A. Galler. Automatic Music Transcription. *Computer Music Journal*, 1(4):24-31, 1977.

[9] Amazing MIDI, http://www.pluto.dti.ne.jp/~araki/amazingmidi/index.html

[10] intelliScore, http://www.intelliscore.net/