A Class of Physical Modeling Recurrent Networks for Analysis/Synthesis of Plucked String Instruments

Alvin W. Y. Su, *Member, IEEE,* and Sheng-Fu Liang

***Abstract*—A new approach is proposed that closely synthesizes tones of plucked string instruments by using a class of physical modeling recurrent networks. The strategies employed in this paper consist of a fast training algorithm and a multistage training procedure that are able to obtain the synthesis parameters for a specific instrument automatically. The training vector can be recorded tones of most target plucked instruments with ordinary microphones. The proposed approach delivers encouraging results when it is applied to different types of plucked string instruments such as steel-string guitar, nylon-string guitar, harp, Chin, Yueh-chin, and Pipa. The synthesized tones sound very close to the originals produced by their acoustic counterparts. In addition, this paper presents an embedded technique that can produce special effects such as vibrato and portamento that are vital to the playing of plucked-string instruments. The computation required in the resynthesis processing is also reasonable.**

***Index Terms*—Physical modeling, plucked string instruments, portamento, recurrent networks.**

I. Introduction

TRANSIENT responses of most acoustic instruments are very difficult to reproduce. This is also the main reason that synthetic sounds are not realistic enough with traditional approaches such as wavetable and FM methods. Model-based approaches claim to be able to reproduce such dynamics by modeling the sounding mechanism of a target instrument physically. There are plenty of works focusing on analysis and modeling of piano soundboards, and top plates and air cavities of guitars and violins [1]–[3]. Techniques such as finite element based methods and ray-tracing methods are useful in analyzing musical instruments but none of them are practical enough to be used to synthesize musical tones in real-time applications. The most successful applications of model-based techniques are compression, synthesis, and recognition of speech signals by simulating human vocal tracts with a class of digital lattice filters [4], [5]. Among several physical-modeling music synthesis methods, the digital waveguide filters (DWFs) [6]–[8], [11] and the wave digital filters (WDFs) [9], [10] are the most popular and practical ones. An efficient way of applying the DWF method to plucked-string instruments has been proposed in [12], [13]. There are some problems with these approaches, however. First, the synthesizer design is complicated. Second, the excitation wavetable takes lots of memory space. Finally, the solution for special effects such as portamento usually seen in plucked-string instruments is not addressed.

In [14] and [15], a recurrent network based approach called scattering recurrent network (SRN) for simulating the vibration of a plucked string succeeded in synthesizing plucked-string tones. The structure of the SRN is similar to a lattice filter because this is the basic form that simulates one-dimensional (1-D) wave propagation. One of the major contributions of this approach is that the system parameters can be determined automatically by using the backpropagation through time (BPTT) training algorithm [16]. However, there exist some difficulties when this technology is applied to practical music synthesis systems. First, structures of musical instruments are usually too complicated to be modeled by such a simple 1-D model. Even if a multidimensional architecture is used [17], the computation for the re-synthesis processing will be enormous. Second, it is usually difficult to measure the time domain responses of a played instrument at various positions so that the measurement can be used as the training vector. Third, the BPTT method takes lots of iterations to converge. Finally, the simple waveforms used by SRN as the excitation signals can no longer produce good synthesis results in any case.

What we want to achieve is to accurately synthesize the tones for any specific plucked-string instrument with reasonable cost. Furthermore, it is desired that the synthesizer design can be done automatically. Therefore, several modifications to the SRN method are proposed. First, the architectures of the networks are simplified so that the complexities required in the training stage and the synthesis stage can be reduced. Second, the training vectors can be musical tones recorded by using ordinary microphones. This allows easy measurement for users without complicate measurement devices. Third, a new training algorithm modified from simulated annealing resilient backpropagation (SARPROP) is used to speed up the training and obtain better system parameters [16], [18]. Fourth, the excitation wavetable should be kept small in its size and can be obtained in the training process. It is noted that it is very difficult for the training to converge to a good solution without this step. Finally, portamento and vibrato effects should be embedded.

In Section II, a class of physical modeling recurrent networks is proposed and the simplified version of the networks is presented. Its connection with the 1-D string model is described. In Section III, resynthesis processing with the proposed technique is presented. In Section IV, a multistage training procedure and a new training algorithm are presented. Synthesis model parameters and the excitation signal are obtained in this stage.

Manuscript received November 15, 1999; revised March 19, 2001. This work was supported in part by National Science Council, Taiwan, R.O.C., under Grant NSC 89-2218-E-006-132.

A. W. Y. Su is with the Department of CSIE, National Cheng-Kung University, Tainan, Taiwan.

S.-F. Liang is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan.

Publisher Item Identifier S 1045-9227(02)03982-6.

Techniques to produce vibrato and portamento are described in Section V. In Section VI, analysis and synthesis works are performed over several plucked-string instruments. Conclusion and suggestions of future work are given in Section VII.

II. Physical Modeling Recurrent Network for Plucked-String Instruments

The basic idea of physical modeling synthesis techniques is to simulate the dynamic behavior of a musical instrument. The major problem with this synthesis technique is the determination of the synthesis model parameters. A recurrent network synthesis model called SRN [14], [15] was proposed to solve the parameter determination problem when a simple musical string is modeled.

The structure of the SRN model is constructed based on the physical model of an acoustic string and this network is successfully used to synthesize some realistic tones for a plucked musical string that was analyzed by SRN. Morse derived the 1-D wave equation for a vibrating string [1]. The wave equation of a string with purely resistive loss is

$$K\frac{\partial^2}{\partial x^2}y(t,x) = u\frac{\partial}{\partial t}y(t,x) + \varepsilon\frac{\partial^2}{\partial t^2}y(t,x) \tag{1}$$

where K is the string tension, u is the resistive parameter and ε is the string density. Here, the force is assumed to be linearly proportional to the transverse velocity [2], [15], [19]. The general solution to (1) can be obtained as

$$y(t,x) = e^{-(u/2\varepsilon)(x/c)}y_r(t-x/c) + e^{(u/2\varepsilon)(x/c)}y_l(t+x/c) \tag{2}$$

where $c = \sqrt{K/\varepsilon}$ is the traveling wave speed. $y(t,x)$ is used to represent the displacement of a vibrating string as the function of position and time. Let the sampling period be T, the discrete-time signal representation of (2) is shown as

$$y(t_n,x_m) = e^{-(u/2\varepsilon)mT}y_r((n-m)T) + e^{(u/2\varepsilon)mT}y_l((n+m)T). \tag{3}$$

Since a real string may not be uniform in its construction, scattering junctions are applied to model a nonuniform string [2], [19]. If there is a nonuniform junction on a string, let the characteristic impedances of the two sides be Z_1 and Z_2, as shown in Fig. 1. The right-going traveling wave flowing to this junction from the left-hand side and the left-going traveling wave flowing to this junction from the right-hand side are φ_r^1 and φ_l^2, respectively. The relation among the traveling waves can be described as follows (readers can refer to [15], [19] for thorough physical explanation)

$$y^J = (1-\rho)\varphi_r^1 + (1+\rho)\varphi_l^2 \tag{4}$$

and

$$\begin{cases} \varphi_l^1 = (-\rho)\varphi_r^1 + (1+\rho)\varphi_l^2 \\ \varphi_r^2 = (1-\rho)\varphi_r^1 + \rho\varphi_l^2. \end{cases} \tag{5}$$

According to (2)–(5), the SRN model is shown in Fig. 2(a). Electromagnetic pickups are used to measure the vibration of a plucked musical string at various sampling positions. The measurement is used as the training vector to obtain the system parameters with the BPTT method [16]. In the synthesis phase, the initial excitation is the waveform obtained with an interpolation method by using the magnitudes of the measurement at $t = 0$ of the pickups. Fig. 3(a) shows such an initial excitation of SRN for the modeling of a Chin E string. In our experiments, simple triangular-like waveforms can be used as the initial excitation to simulate the "plucks."

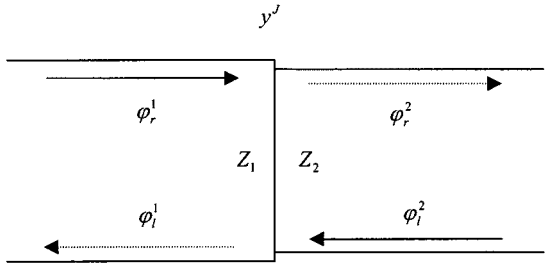
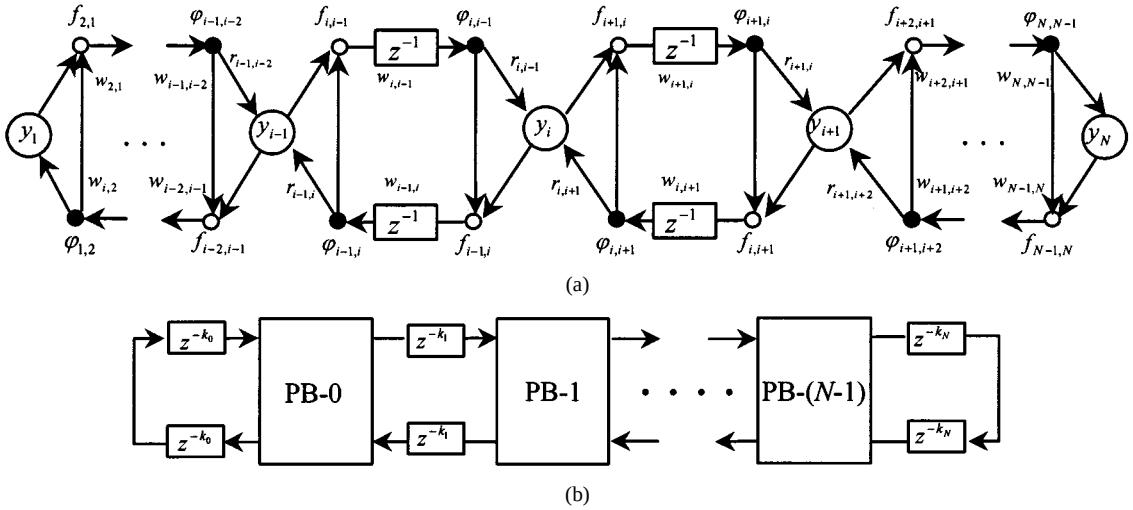


Fig. 1. A nonuniform junction where the acoustic impedances on the two sides are not identical.

Although SRN succeeds in synthesizing the tones of plucked strings, it fails to be the synthesis model of a musical instrument because a tone produced by a plucked-string instrument is the combined responses of strings, bridge, body, and air cavity with respect to a pluck. Several modifications to the SRN method are proposed for synthesizing string instrument tones. First, the computation required for the SRN is too large in practice. A simplified version of the physical modeling recurrent network is proposed so that the computation required in the training stage and the synthesis stage can be reduced. The new network structure is shown in Fig. 2(b). This model consists of three basic components: processing blocks (PBs), simple delay lines and two reflective ends. Between two adjacent PBs, there is a pair of delay lines that make the connection. The PBs simulate the energy loss as well as the scattering behavior [19]. The structure of a PB is shown in Fig. 4(a) and the computation is identical with that of SRN. There are three types of neurons in a PB, displacement neurons, arrival neurons and departure neurons. The output of a displacement neuron, denoted by $y_{i,j}$, represents the amplitude at the jth sampling position in PB-i. The outputs of arrival neurons, denoted by $\varphi_{i,j}^r$ and $\varphi_{i,j}^l$, represent the right-going traveling wave and the left-going traveling wave flowing into displacement neuron $y_{i,j}$, respectively. The outputs of departure neurons, denoted by $f_{i,j}^r$ and $f_{i,j}^l$, represent the traveling waves leaving $y_{i,j}$ and injecting into the right-hand-side delay line and the left-hand-side delay line, respectively. A pair of delay lines that connect PB-$(i-1)$ and PB-i is shown in Fig. 4(b). Within each pair of delay lines, signals pass through them directly without any modification. Thus, the computation in the synthesis processing can be reduced. In our experiments, seven PBs are used in the proposed model and each PB contains three displacement neurons. If the system requires 100 unit delays from one fixed end to another, the computation cost is about $1/5$ times of the original SRN model [15]. When a traveling wave meets a fixed end, it will completely reflect back with opposite phase. The operation of two reflective ends in the proposed model is shown in Fig. 4(c).

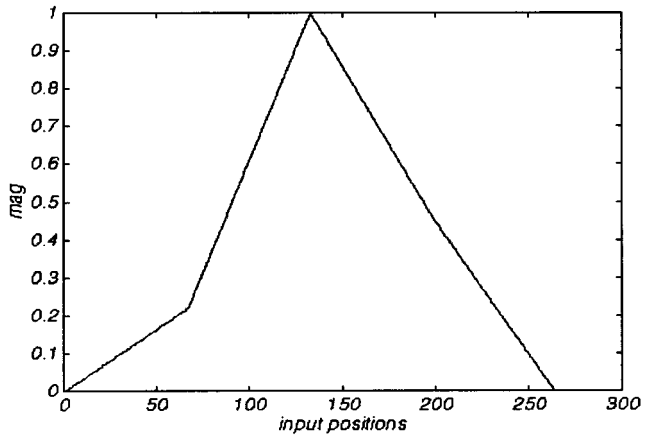
Second, electromagnetic pickups used in [15] can only measure the string vibration and the result is not what people usually



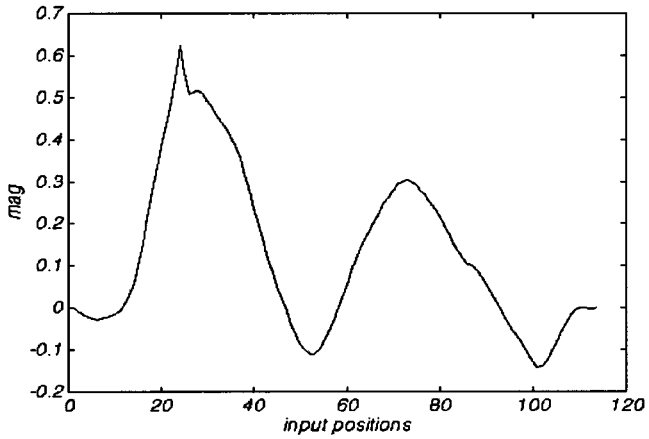
(a)

(b)

Fig. 2. The SRN model and the proposed synthesis model. (a) The SRN model for the modeling of musical strings. (b) The proposed new network structure for synthesis of plucked-string instrument tones.



(a)



(b)

Fig. 3. The excitation waveforms for string modeling and tone synthesis. (a) The excitation waveform of SRN for the modeling of a Chin E string. (b) An excitation signal for the synthesis of Chin tone.

hear. Actually, the sound picked up at some distance is closer to what we hear. Therefore, microphones instead of pickups are used to obtain string instrument tones as training vectors. Third, the tone of a string instrument is much more complex than that of a vibrating string. Simple triangular-like excitation waveforms cannot be used when such complex waveforms are analyzed. Since the wavetable size is kept to a minimum in the SRN approach, it is desired to keep this property. The excitation wavetable is also obtained in the training process and its size is equal to the length from one reflective end to the other. Fig. 3(b) shows the excitation signal for synthesizing a Chin tone. It is found that this excitation waveform is much more complicated than the one used in modeling musical strings.

III. Synthesis Processing

The synthesis processing of the proposed model contains two stages: the *initialization stage* and the *propagation stage*. In the initialization stage, the excitation waveform is loaded into the synthesis model with suitable system parameters obtained from the training stage. Then, the excitation waveform as Fig. 3(b) is distributed into the upper and lower tracks in the synthesis model shown in Fig. 2(b), respectively. After the initialization, the propagation operation starts to generate the desired synthesized data without any additional information.

- Initialization stage

In this stage, an initial excitation waveform has to be provided. The size of the initial waveform equals to the total delay length, L, in the upper track and lower track of the synthesis model shown in Fig. 2(b). If the delay length in both tracks is L unit delays, it is computed as

$$L = \left\lceil 0.5 \cdot \frac{f_s}{f} \right\rceil \tag{6}$$

where f_s is the sampling rate of the synthesis system and f is the fundamental frequency of the desired tone. In our experiment, the size of L is only hundreds of samples. Therefore, the memory cost is much less than that of a Wavetable method as well as other traditional model-based synthesis techniques that require longer recorded tone as the excitation signal (thousands of samples with 44.1 kHz sampling rate) [12]. According to Fig. 4(a), $I^y_{i,j}$ is the initial magnitude of the displacement neuron $y_{i,j}$ in the PB-i, i.e.

$$y_{i,j}(0) = I^y_{i,j}. \tag{7}$$

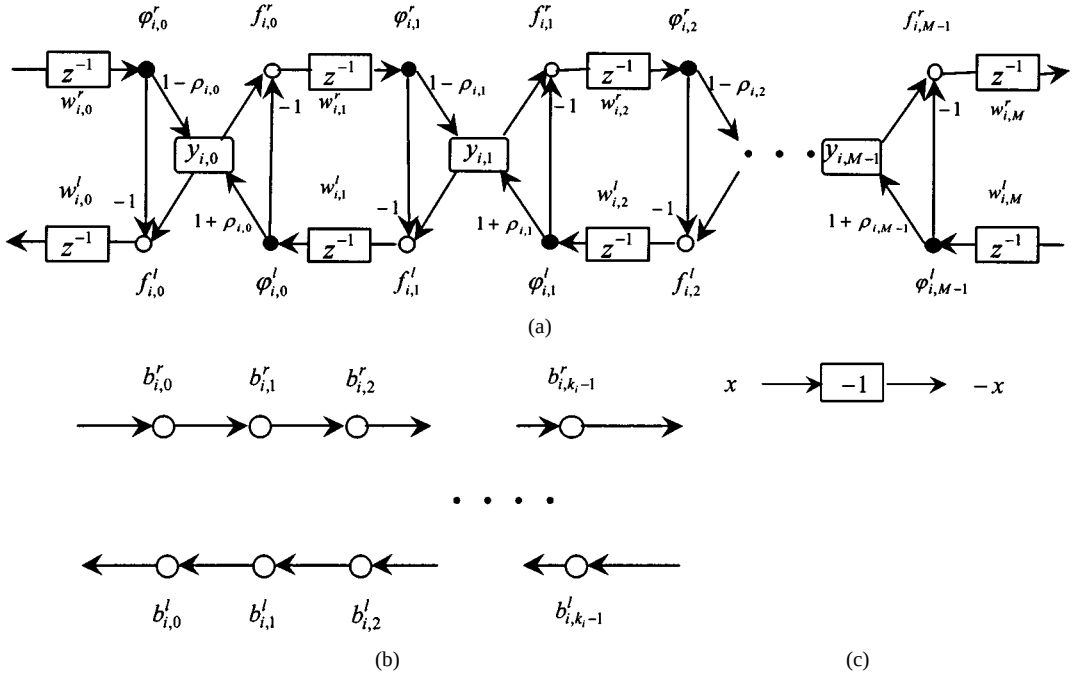


Fig. 4. The three basic components in the synthesis model shown in Fig. 2(b). (a) The structure of a PB. (b) A pair of delay lines that connect PB-$(i-1)$ and PB-i. (c) The operation of two reflective ends.

Then, it is equally distributed into the right-going departure neuron $f^r_{i,j}$ and the left-going departure neuron $f^l_{i,j}$ as follows:

$$f^r_{i,j}(0) = f^l_{i,j}(0) = 0.5 \cdot y_{i,j}(0). \tag{8}$$

According to Fig. 4(b), $I^b_{i,j}$ represents the initial magnitude of the jth delay unit of the ith pair of the delay lines. Similar to (8), the initial values of delay buffers are also one half of $I^b_{i,j}$.

$$b^r_{i,j}(0) = b^l_{i,j}(0) = 0.5 \cdot I^b_{i,j}. \tag{9}$$

After the initialization stage is finished, the propagation stage starts.

• Propagation stage

Let the number of PBs be N in the physical modeling recurrent network and the number of displacement neurons in each PB be M. According to Figs. 2(b) and 4, let $b^r_{i,j}$ and $b^l_{i,j}$ represent the jth delay buffers of the ith delay segments in the upper and lower tracks connecting PB-$(i-1)$ and PB-i, the traveling waves in the upper and lower tracks can be represented as

$$b^r_{i,j}(t+1) = \begin{cases} b^r_{i,j-1}(t), & 0 \le i \le N, \quad 1 \le j \le k_i - 1, \\ w^r_{i-1,M} \cdot f^r_{i-1,M-1}(t), & 1 \le i \le N, \quad j = 0, \\ -b^l_{0,0}(t), & i = 0, \quad j = 0 \end{cases} \tag{10}$$

and

$$b^l_{i,j}(t+1) = \begin{cases} b^l_{i,j+1}(t), & 0 \le i \le N, \quad 0 \le j \le k_i - 2 \\ w^l_{i,0} \cdot f^l_{i,0}(t), & 0 \le i \le N-1, \quad j = k_i - 1, \\ -b^r_{N,k_N-1}(t), & i = N, \quad j = k_N - 1 \end{cases} \tag{11}$$

where the upper-track boundary delay buffer $b^r_{i,0}$ receives the *right-going* output signal of PB-$(i-1)$, $w^r_{i-1,M} \cdot f^r_{i-1,M-1}$, and the lower-track boundary delay buffer b^l_{i,k_i-1} receives the *left-going* output signal of PB-i, $w^l_{i,0} \cdot f^l_{i,0}$.

There are two basic operations in a neuron. The first one sums the signals flowing into this neuron as the net-input. The second one is the so-called activation function that is a mapping between the net-input and the corresponding output. The arrival neurons receive the weighted outputs from the departure neurons or those from the delay buffers as

$$\begin{aligned} \varphi^r_{i,j}(t+1) &= a\left(\text{net\_}\varphi^r_{i,j}(t+1)\right) \\ &= \begin{cases} a\left(w^r_{i,j} \cdot f^r_{i,j-1}(t)\right), & 1 \le j \le M-1 \\ a\left(w^r_{i,j} \cdot b^r_{i,k_i-1}(t)\right), & j = 0 \end{cases} \end{aligned} \tag{12}$$

and

$$\begin{aligned} \varphi^l_{i,j}(t+1) &= a\left(\text{net\_}\varphi^l_{i,j}(t+1)\right) \\ &= \begin{cases} a\left(w^l_{i,j+1} \cdot f^l_{i,j+1}(t)\right), & 0 \le j \le M-2, \\ a\left(w^l_{i,j+1} \cdot b^l_{i+1,0}(t)\right), & j = M-1 \end{cases} \end{aligned} \tag{13}$$

where $\text{net\_}\varphi^r_{i,j}$ denotes the net-input of arrival neuron $\varphi^r_{i,j}$ and $a(\cdot)$ is the activation function. These notations are also used in the following derivations. The displacement neurons receiving the outputs from the adjacent arrival neurons is obtained by

$$\begin{aligned} y_{i,j}(t+1) &= a(\text{net\_}y_{i,j}(t+1)) \\ &= a\left((1-\rho_{i,j}) \cdot \varphi^r_{i,j}(t+1) + (1+\rho_{i,j}) \cdot \varphi^l_{i,j}(t+1)\right) \\ & \qquad 0 \le j \le M-1. \end{aligned} \tag{14}$$

Finally, the traveling waves departing from the displacement neurons to the nearby segments can be computed by

$$f^r_{i,j}(t+1) = a\left(\text{net\_}f^r_{i,j}(t+1)\right) = a\left(y_{i,j}(t+1) - \varphi^l_{i,j}(t+1)\right), \quad 0 \le j \le M-1 \quad (15)$$

and

$$f^l_{i,j}(t+1) = a\left(\text{net\_}f^l_{i,j}(t+1)\right) = a\left(y_{i,j}(t+1) - \varphi^r_{i,j}(t+1)\right) \quad 0 \le j \le M-1. \quad (16)$$

Equations (10) through (16) represent a propagation cycle to produce a synthesized sample for one time step. The synthesized signal is the output of a chosen displacement neuron. Although nonlinear activation functions with trainable parameters could possibly be used for better performance, it increases computation complexity. Therefore, the activation function for each neuron is an identity function in all of the experiments and activation functions are discarded in the later sections to simplify our notation.

IV. Training

There are some improvements with the model compared to the one in [15]. First, a multistage training procedure is used to obtain multiple sets of synthesis parameters for the varying characteristics of an instrument. Second, a supervised training method is used to obtain the synthesis parameters and the initial excitation waveform automatically. Third, a hybrid-training algorithm is used to speed up the training and obtain better synthesis parameters.

A. Multistage Training Strategy

The multistage training strategy for the synthesis model is shown in Fig. 5. The mean square difference between the recorded and the synthesized tones is used to adjust the initial excitation waveform and synthesis parameters. In Stage #1, the initial excitation waveform, denoted by $I^b_{i,j}$ and $I^y_{i,j}$ in (7) and (9), as well as the first set of the synthesis parameters have to be determined. This training stage employs the recorded tone within the interval $[t_{1,0}, t_{1,1}]$ as the training vector and the resultant parameters are used for synthesizing a tone from $t_{1,0}$ to $t_{2,0}$. Because the synthesis processing no longer requires external signals after the initialization stage, it is not necessary to have the initial input waveform updated after this stage. Stage #2 begins at $t_{2,0}$ by using the recorded tone from $t_{2,0}$ to $t_{2,1}$ as the training vector and the second set of parameters is obtained when this training stage finishes. This procedure stops when all the training vectors are finished.

B. Training Algorithm

For a recurrent neural network (RNN), BPTT [16], [20] is a widely used training algorithm that is an extension of the standard backpropagation algorithm. This algorithm requires at least 10 000 epochs to converge when it is applied to the proposed synthesis model. In [18], the SARPROP method is proposed for feedforward type networks. This algorithm combines

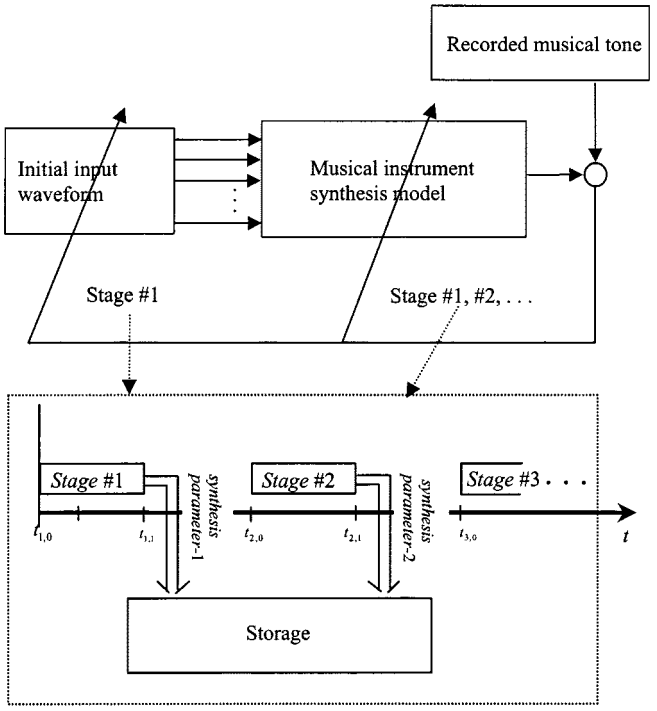


Fig. 5. The multistage training strategy for determination of model parameters.

a quick gradient descent algorithm, called resilient backpropagation (RPROP) [22], with a simulated annealing (SA)-based global searching technique [23]. The RPROP takes into account the sign of the gradient as seen by a particular parameter instead of the magnitude of the gradient. The SA involves the addition of random noise to the parameter updates as well as decreases the magnitudes of the updates in the training process gradually. The SARPROP method can indeed converge much faster compared to the BPTT method. In our experiments, it is found that the SARPROP method is very sensitive to the learning parameters and the initial condition of the system parameters. The training diverges or converges to a totally unacceptable solution for a recurrent network soemtimes.

In this paper, a hybrid-training algorithm consisting of BPTT and SARPROP as shown in Fig. 6 is used in the training procedure of the proposed physical modeling recurrent network. Since this synthesis network is a recurrent neural network, BPTT is used to calculate the magnitude of the gradient for each parameter and the corresponding parameter update value is obtained by SARPROP. In Stage #1, both synthesis parameters and initial excitation waveform must be updated. Only synthesis parameters are updated in the other stages. Particularly, the synthesis network is constructed based on a simplified physical model of a musical instrument. Therefore, each of the synthesis parameters has its physical meaning. The ρ-type parameters simulate the nonuniform characteristics at various physical positions and the w-type ones simulate the energy decay factors. The initial values of the synthesis parameters can be reasonable values derived from the physical characteristics of the target instrument instead of random values such that the training can be better. This is different

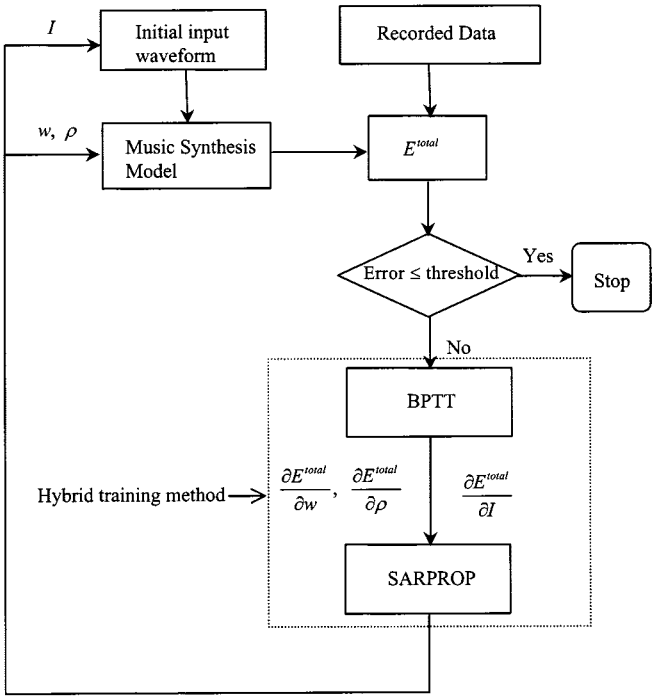


Fig. 6. The hybrid-training algorithm consisting of BPTT and SARPROP for the training procedure of the proposed model.

from most other applications using neural networks as their parameter-finding mechanisms.

The temporal operation of the proposed model can be unfolded into a multilayer feedforward architecture with synchronous update. Those who are interested in this can refer to references such as [15], [21]. A neural network layer representing one time instant is called a *time layer*. Since the synthesized signal is the output of a chosen displacement neuron, only this displacement neuron can have a teacher signal that is the recorded tone of the target instrument. The displacement neuron that generates the synthesized output to match the desired output is called the *visible neuron* and the remaining neurons are called hidden neurons. In each training stage, let $d(t)$ denote the recorded tone at time t and let the output of the corresponding displacement neuron $y_{r,s}$, which is the sth displacement neuron in PB-r, be chosen as the synthetic result of the synthesis model. The error signal at any time t is defined as

$$e(t) = d(t) - y_{r,s}(t) \tag{17}$$

and the error function is defined as

$$E(t) = \frac{1}{2} \cdot e^2(t). \tag{18}$$

The total cost function to be minimized in the interval $[t_0, t_1]$ is defined by

$$E^{\text{total}}(t_0, t_1) = \sum_{t=t_0}^{t_1} E(t) = \frac{1}{2} \cdot \sum_{t=t_0}^{t_1} e^2(t). \tag{19}$$

The gradient values for the ρ-type parameters corresponding to time layer t can be derived as

$$\begin{aligned} \delta\rho_{i.j}(t) &= \frac{\partial E^{\text{total}}(t_0, t_1)}{\partial \text{net\_}y_{i,j}(t)} \cdot \frac{\partial \text{net\_}y_{i,j}(t)}{\partial \rho_{i,j}} \\ &= \delta y_{i,j}(t) \cdot \left(\varphi^l_{i,j}(t) - \varphi^r_{i,j}(t)\right) \end{aligned} \tag{20}$$

where $\delta y_{i,j}(t)$ represents the local error of the neuron $y_{i,j}$ at time layer t. The gradient values of the w-type parameters at time layer t can also be derived as shown in (21) and (22) at the bottom of the page.

The local error of a displacement neuron can be obtained by (23). If a displacement neuron is a hidden neuron, its gradient value can be computed based on the collection of the local errors of the departure neurons connected with it. If this displacement neuron is a visible neuron (denoted as $y_{r,s}$), it means that it will directly contribute the error signal, as shown in (17), to the total

$$\begin{aligned} \delta w^r_{i,j}(t) &= \begin{cases} \dfrac{\partial E^{\text{total}}(t_0, t_1)}{\partial \text{net\_}\varphi^r_{i,j}(t)} \cdot \dfrac{\partial \text{net\_}\varphi^r_{i,j}(t)}{\partial w^r_{i,j}}, & j = 0, 1, \ldots, M-1 \\ \dfrac{\partial E^{\text{total}}(t_0, t_1)}{\partial b^r_{i+1,0}(t)} \cdot \dfrac{\partial b^r_{i+1,0}(t)}{\partial w^r_{i,j}}, & j = M \end{cases} \\ &= \begin{cases} \delta\varphi^r_{i,j}(t) \cdot b^r_{i,k_i-1}(t-1), & j = 0 \\ \delta\varphi^r_{i,j}(t) \cdot f^r_{i,j-1}(t-1), & j = 1, 2, \ldots, M-1 \\ \delta b^r_{i+1,0}(t) \cdot f^r_{i,j-1}(t-1), & j = M \end{cases} \end{aligned} \tag{21}$$

and

$$\begin{aligned} \delta w^l_{i,j}(t) &= \begin{cases} \dfrac{\partial E^{\text{total}}(t_0, t_1)}{\partial \text{net\_}\varphi^l_{i,j-1}(t)} \cdot \dfrac{\partial \text{net\_}\varphi^l_{i,j-1}(t)}{\partial w^l_{i,j}}, & j = 1, \ldots, M \\ \dfrac{\partial E^{\text{total}}(t_0, t_1)}{\partial b^l_{i,k_i-1}(t)} \cdot \dfrac{\partial b^l_{i,k_i-1}(t)}{\partial w^r_{i,j}}, & j = 0 \end{cases} \\ &= \begin{cases} \delta\varphi^l_{i,j-1}(t) \cdot f^l_{i,j}(t-1), & j = 1, 2, \ldots, M-1 \\ \delta\varphi^l_{i,j-1}(t) \cdot b^l_{i+1,0}(t-1), & j = M \\ \delta b^l_{i,k_i-1}(t) \cdot f^l_{i,j}(t-1), & j = 0. \end{cases} \end{aligned} \tag{22}$$

cost function. Therefore, this error signal must be involved in the computation of the local error corresponding to this neuron

$$\begin{aligned}\delta y_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial \text{net\_}y_{i,j}(t)} \\ &= \begin{cases} \left(-e(t)+\delta f^r_{i,j}(t)+\delta f^l_{i,j}(t)\right), & i=r, \quad j=s \\ \left(\delta f^r_{i,j}(t)+\delta f^l_{i,j}(t)\right), & \text{otherwise.} \end{cases}\end{aligned} \tag{23}$$

The local error of a departure neuron is obtained based on the local error of the arrival neuron or the delay buffer connected with this departure neuron by

$$\begin{aligned}\delta f^r_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial \text{net\_}f^r_{i,j}(t)} \\ &= \begin{cases} \delta\varphi^r_{i,j+1}(t+1)\cdot w^r_{i,j+1}, & j=0,1,\ldots,M-2. \\ \delta b^r_{i+1,0}(t+1)\cdot w^r_{i,j+1}, & j=M-1. \end{cases}\end{aligned} \tag{24}$$

and

$$\begin{aligned}\delta f^l_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial \text{net\_}f^l_{i,j}(t)} \\ &= \begin{cases} \delta\varphi^l_{i,j-1}(t+1)\cdot w^l_{i,j}, & j=1,2,\ldots,M-1. \\ \delta b^l_{i,k_i-1}(t+1)\cdot w^l_{i,j}, & j=0. \end{cases}\end{aligned} \tag{25}$$

The local error of an arrival neuron is obtained based on the collection of the local errors of both the displacement neuron and the departure neuron connected with this arrival neuron and it can be computed by

$$\begin{aligned}\delta\varphi^r_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial \text{net\_}\varphi^r_{i,j}(t)} \\ &= \left(\delta y_{i,j}(t)\cdot(1-\rho_{i,j})-\delta f^l_{i,j}(t)\right)\end{aligned} \tag{26}$$

and

$$\begin{aligned}\delta\varphi^l_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial \text{net\_}\varphi^l_{i,j}(t)} \\ &= \left(\delta y_{i,j}(t)\cdot(1+\rho_{i.j})-\delta f^r_{i,j}(t)\right).\end{aligned} \tag{27}$$

The local errors of the delay buffers in the upper track and the lower track can be computed as shown in (28) and (29) at the bottom of the page.

Since the total gradient for each parameter is the sum of the gradient value corresponding to every time layer, the total gradient for the synthesis parameters in one epoch can be computed by

$$\frac{\partial E^{\text{total}}(t_0,t_1)}{\partial \rho_{i,j}(\text{epoch})} = \sum_{t=t_0}^{t_1}\delta\rho_{i,j}(t) \tag{30}$$

and

$$\begin{cases} \dfrac{\partial E^{\text{total}}(t_0,t_1)}{\partial w^r_{i,j}(\text{epoch})} = \displaystyle\sum_{t=t_0}^{t_1}\delta w^r_{i,j}(t). \\ \dfrac{\partial E^{\text{total}}(t_0,t_1)}{\partial w^l_{i,j}(\text{epoch})} = \displaystyle\sum_{t=t_0}^{t_1}\delta w^l_{i,j}(t). \end{cases} \tag{31}$$

In addition, if the training is in Stage #1, the excitation signal used in (7) and (9) should be obtained in a similar way. When the backpropagation computation is performed back to $t=0$, the gradient value for the excitation signal corresponding to delay buffers is computed by

$$\frac{\partial E^{\text{total}}(t_0,t_1)}{\partial I^b_{i,j}(\text{epoch})} = 0.5\cdot\left(\delta b^r_{i,j}(0)+\delta b^l_{i,j}(0)\right) \tag{32}$$

and the gradient value for initial waveform corresponding to the displacement neurons is computed by

$$\begin{aligned}&\frac{\partial E^{\text{total}}(t_0,t_1)}{\partial I^y_{i,j}(\text{epoch})} \\ &\quad= \begin{cases} 0.5\cdot\left(\delta f^r_{i,j}(0)+\delta f^l_{i,j}(0)\right)+e(0), & i=r, \quad j=s. \\ 0.5\cdot\left(\delta f^r_{i,j}(0)+\delta f^l_{i,j}(0)\right), & \text{otherwise.} \end{cases}\end{aligned} \tag{33}$$

According to Fig. 6, the amount of gradient values of the synthesis parameters or the excitation waveform is obtained by BPTT for each epoch. Then, they are transferred to the SARPROP [18] to determine the amount of adjustment. The neural network used in [18] was a multilayer perceptron (MLP) structure and the initial update parameter Δ_0 was 0.1. If the initial values of the network parameters are assigned randomly, it is found that this approach is not good for our application. Phys-

---

$$\begin{aligned}\delta b^r_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial b^r_{i,j}(t)} \\ &= \begin{cases} \delta b^r_{i,j+1}(t+1), & 0\le i\le N, \quad 0\le j\le k_i-2. \\ \delta\varphi^r_{i,0}(t+1)\cdot w^r_{i,0}, & 0\le i\le N-1, \quad j=k_i-1. \\ -\delta b^l_{N,k_N-1}(t+1), & i=N, \quad j=k_N-1 \end{cases}\end{aligned} \tag{28}$$

and

$$\begin{aligned}\delta b^l_{i,j}(t) &= \frac{\partial E^{\text{total}}(t_0,t_1)}{\partial b^l_{i,j}(t)} \\ &= \begin{cases} \delta b^l_{i,j-1}(t+1), & 0\le i\le N, \quad 1\le j\le k_i-1. \\ \delta\varphi^l_{i-1,M-1}(t+1)\cdot w^l_{i-1,M}, & 1\le i\le N-1, \quad j=0. \\ -\delta b^r_{0,0}(t+1), & i=0, \quad j=0. \end{cases}\end{aligned} \tag{29}$$

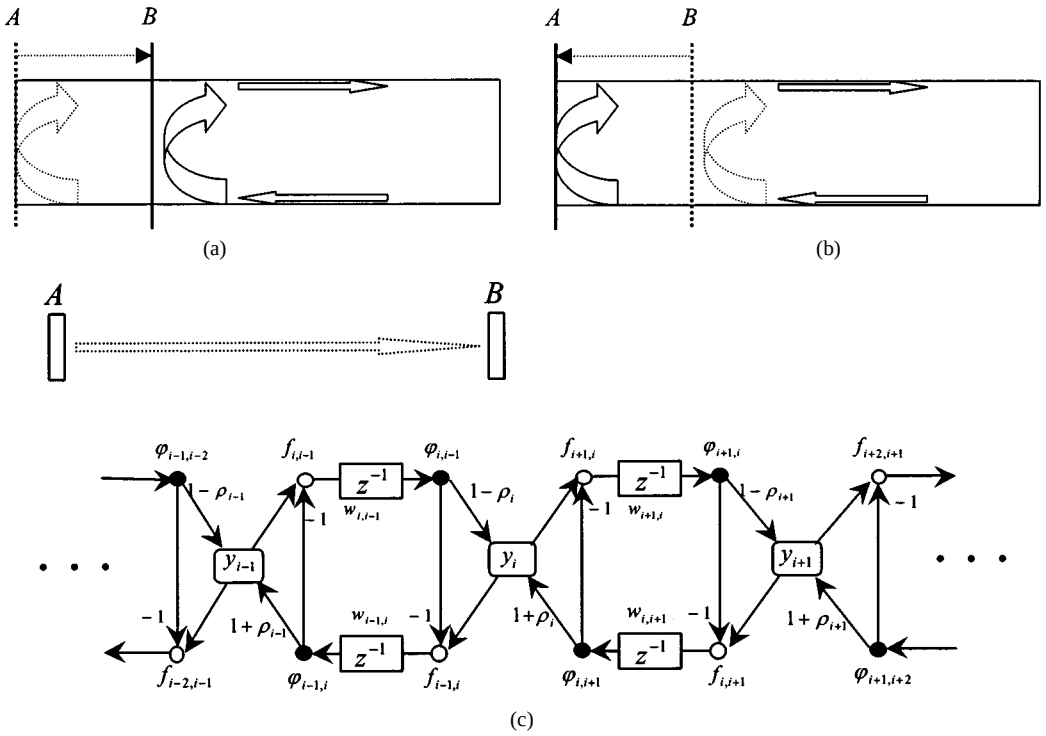


Fig. 7. Simulation of left-hand finger-gliding playing behavior. (a) The left-hand finger glides along the string from position A to position B. (b) The left-hand finger glides along the string from position B to position A. (c) Modified structure of the proposed model for simulating the finger-gliding effect.

ically, the ρ-type parameters are the reflection coefficients and their range should fall between $[-1,1]$. For the w-type parameters, they represent the energy loss factors and the range should be around unity. The following learning parameters are determined empirically and found to be useful in many experiments. The initial update parameter Δ_0 is 0.0001. The temperature parameter T is 0.01. The rest of the constants are set as follows. $\eta^+ = 1.2, \eta^- = 0.5, \Delta_{\max} = 0.2, \Delta_{\min} = 1 \times 10^{-10}$.

In [15], the training processing of SRN required at least 10 000 iterations by using BPTT. In this paper, it is reduced to less than 1000 epochs for each training stage and the results are superior when SNR tests are concerned. BPTT is used to update the synthesis parameters and the excitation waveform in Stage #1 to avoid unstable situation caused by SARPROP.

V. Embedded Vibrato and Portamento Processing

Some plucked-string instruments have no fret. A player's fingers can glide along strings to produce effects such as wide range vibrato and portamento. In this section, an embedded efficient method for such effects is introduced. An example is shown in the next section.

When a left-hand finger is gliding along a string from position A to position B, as shown in Fig. 7(a), the length of a vibrating string becomes shorter gradually and the pitch of the tone also changes from low to high. On the contrary, when the left-hand finger glides along the string from position B to position A, as shown in Fig. 7(b), the length of the vibrating string becomes longer gradually and the pitch changes from high to low.

Since the proposed model is constructed based on the physical model of vibrating strings, the length of this model must change as the gliding behavior stated above to simulate the vibrato and portamento. In order to realistically produce

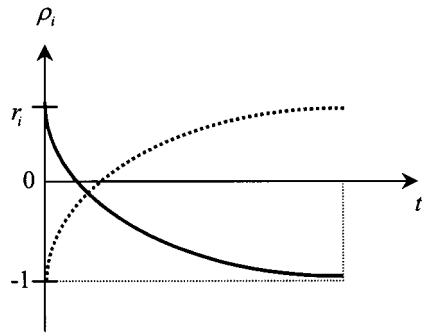


Fig. 8. Operations of ρ-type parameters used for portamento and vibrato. The solid curve represents the situation that the reflection coefficient of the position is changed to -1, which makes the junction be totally reflective. The dash curve represents the situation that the original reflection coefficient is gradually restored.

the finger-gliding effect, the structure of the delay lines shown in Fig. 4(b) within the corresponding gliding region has to be changed to the PB structure shown in Fig. 4(a). For example, if the w-type parameter is 1 and the ρ-type parameter is 0, the traveling waves in the upper track and the lower track will pass through the displacement neuron directly without any modification. Therefore, the proposed model shown in Fig. 2(b) is changed to the one shown in Fig. 7(c). If we want to simulate the behavior of shortening the string such as the situation of shortening the length of the model from position A to position B, it can be realized simply by changing the ρ-type parameters.

If the original value of ρ_i is r_i, the value of ρ_i is decreased gradually from r_i to -1 along the solid curve shown in Fig. 8. When ρ_i equals to -1, the position y_i becomes a fixed end where the left-going traveling wave in the lower track reflects back to the upper track with opposite phase. This means that the left-hand

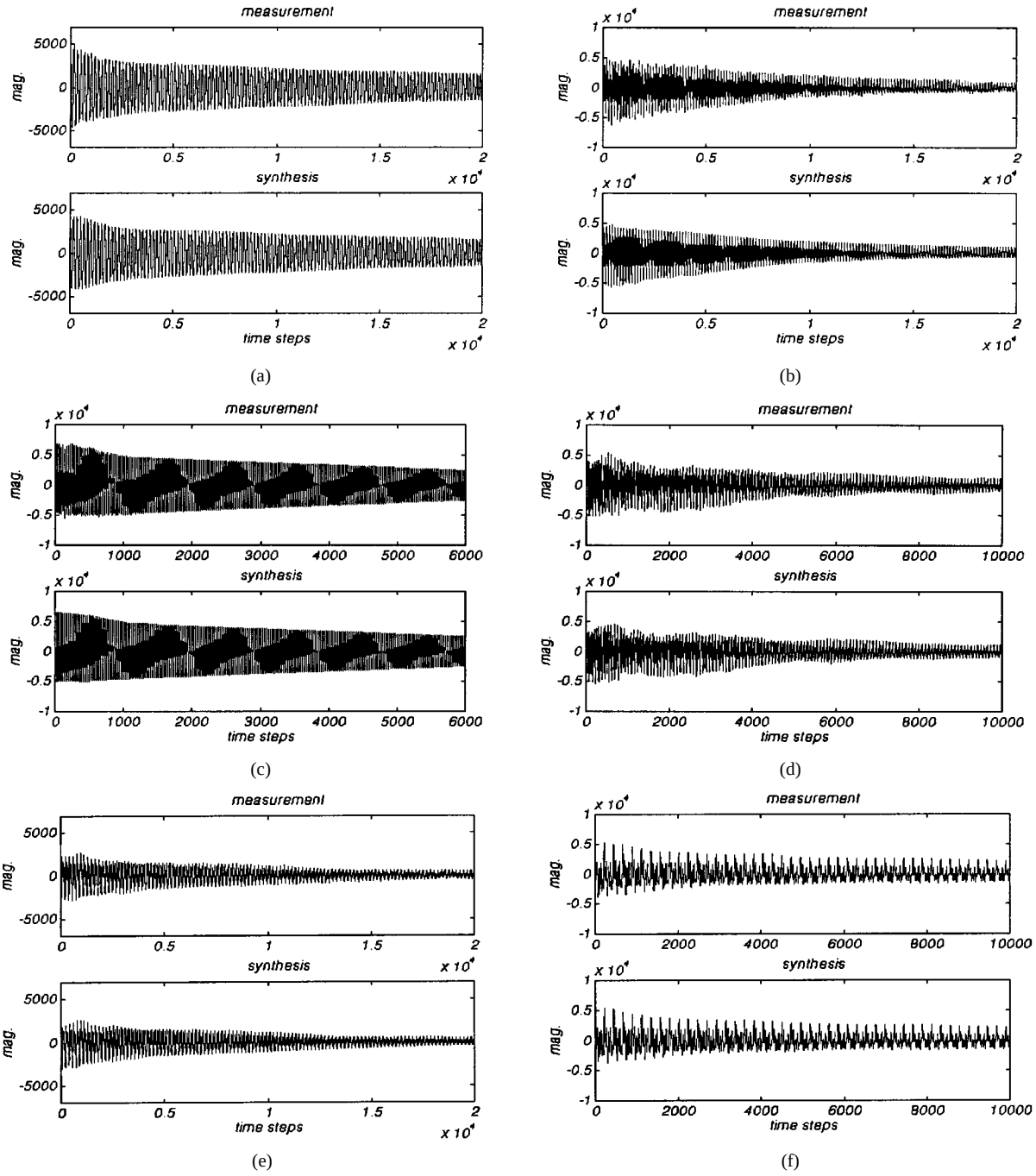


Fig. 9. Original and synthetic tones of various plucked-string instruments. (a) Steel-string guitar. (b) Nylon-string guitar. (c) Harp. (d) Pipa. (e) Yueh-chin. (f) Chin.

finger pressed firmly on the physical position corresponding to position B. In this case, outputs of all the displacement neurons in the region to the left-hand side of position B are forced to zero. On the contrary, if the value of ρ_i is restored gradually from -1 to its original value r_i along the dash curve shown in Fig. 8, the model is gradually restored to the original situation. The pitch of synthetic tone will change from high to low.

Vibrato and Portamento effects are actually produced by combining such shortening and lengthening operations. If the gliding on the string can be described by a function of time, the above effects can be easily achieved by changing the ρ-type parameters of the proposed model according to this function. The synthesized tones produced with these operations can no longer sound so similar to the tones produced by the target acoustic instruments. However, if the initial part of the synthetic tone is similar to the original, subjects tend to consider that these two are produced from the same instrument and the special effects are simply ornaments.

VI. Analysis/Synthesis of Plucked-String Instruments

The followings are the analysis/synthesis experiments with respect to various types of plucked-string instruments, steel-string guitar, nylon-string guitar, harp, Pipa, Yueh-chin,

TABLE I
THE SIGNAL-TO-NOISE RATIOS OF SYNTHETIC RESULTS CORRESPONDING TO VARIOUS MUSICAL INSTRUMENTS

Musical Instruments	steel-string guitar	nylon-string guitar	Harp	Pipa	Yueh-chin	Chin
SNRs	30.26	21.29	31.16	20.24	21.97	26.49

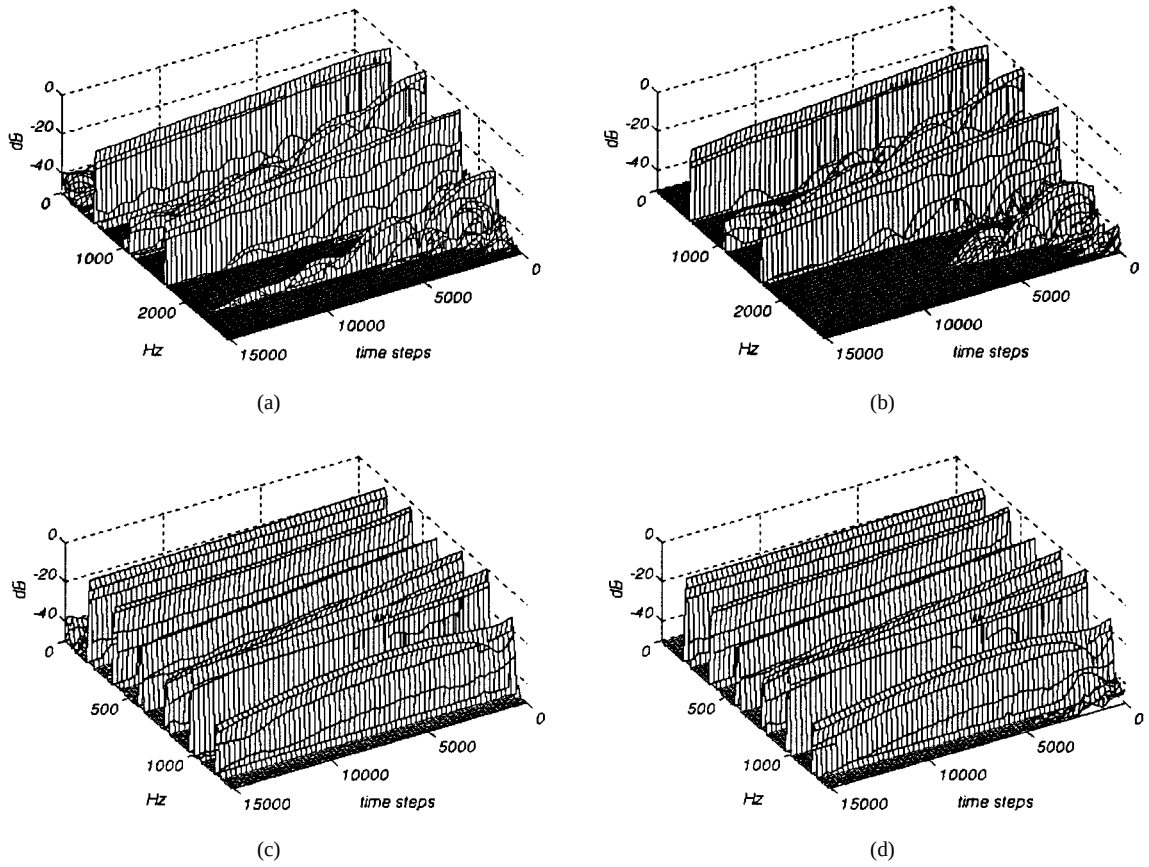


Fig. 10. Short-time-Fourier analysis of the signals shown in Fig. 9(d) and (f). (a) STFA of original Pipa tone. (b) STFA of synthesized Pipa tone. (c) STFA of original Chin tone. (d) STFA of synthesized Chin tone.

and Chin to demonstrate the performance of the proposed method. Pipa, Yueh-Chin, and Chin are three Chinese traditional plucked-string instruments [24]. In each case, there are 7 PBs in the proposed model and each PB contains three displacement neurons.

1) Synthesis Results: The analysis/synthesis results are shown in Fig. 9. Upper part of each subfigure shows the original tone and lower part shows the corresponding synthesized tone. The waveforms of the original tone and the synthesized tone are very close to each other. The SNR for each of the pairs is shown in Table I. By examining Fig. 10(a) and (b), there are still small differences coming from the high-frequency components. In general, if the sounding mechanism of an instrument is less perfect, the synthesis is more difficult. For example, Pipa, a lute-like instrument, has a very thin top plate, a nonrigid bridge and less well-constructed strings [25]. Therefore, its response is less smooth compared to instruments such as harp and Chin. This can also be seen on the STFT plots shown in Fig. 10(c) and (d) that the Chin tone has a smoother decay pattern compared to the Pipa tone. Although the SNR results do not look impressed, such performance is not possible in the past. In fact, most physical modeling synthesis methods can only reproduce the magnitude part of the frequency response. In general, the first few fractions of a second of a tone are how people judge the instrument. Listening tests show that subjects can find differences between the original and the synthesized tones but consider that they do sound very similar and regard that the tones are generated from the same instruments.

2) Portamento Effects: Portamemto and vibrato are frequently used in the playing of many stringed instruments. Chin is an ancient Chinese plucked-string instrument that consists of a shallow rectangular-like wooden chamber and seven strings and is the known instrument that uses portamento and vibrato most. Since there is no fret on the top plate, the player's left-hand fingers can glide along strings to produce vibrato and portamento effects.

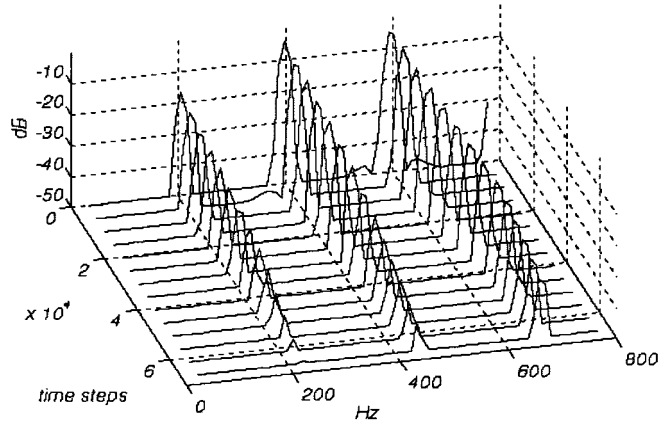


Fig. 11. The STFT plot of the synthetic tone with portamento effect.

The technique described in the previous section is used to simulate portamento. Fig. 11 shows the STFT plot of the simulation. The fundamental frequency is shifted from 190 Hz to 215 Hz. Though the fundamental frequency is shifted to the desired pitch, the timbre has changed and is different from the Chin used in this experiment. Because the beginning transient sounds similar enough to the original, this timbre difference is usually ignored. Nevertheless, to simulate these effects without changing the timbre is still an interesting and challenging topic.

VII. Conclusion and Future Work

A class of physical modeling recurrent networks is proposed to synthesize musical tones of plucked-string instruments. All the required parameters of the synthesis model can be efficiently and automatically obtained by a hybrid BPTT/SARPROP learning algorithm. It is possible to closely synthesize for a specific instrument if electronic musicians consider the sound of this particular instrument is indispensable. The approach is also tested over a wide range of plucked-string instruments and proven to be a very general method. Based on this synthesis model, portamento effect can be easily synthesized. Because the training vector is easy to obtain, it is possible for users to design their own synthesizers. Although the computation complexity in the resynthesis processing is still large, it is close to the computation complexity of speech synthesis. Based on the rapid progress of current DSP processor design, computation cost in this range should not cause much trouble.

Our future works are stated as follows. First, the SNR of the synthetic tone to the original tone is still not good enough. Actually, the high-frequency part contributes most of the error. This will be our major focus. Second, playing techniques play very important roles in how an instrument sounds. For example, Chin has thousands of techniques and each technique produces a different timbre. How to handle this problem is a challenging issue. Finally, it is desired to extend the proposed methodology to other types of instruments such as wind or struck-string instruments. Because different types of instruments have different structures, it is necessary to design suitable physical models for them.

Acknowledgment

The authors would like to thank the editor and the reviewers for their valuable comments.

References

[1] P. M. Morse, *Vibration and Sound*. Woodbury, NY: Amer. Inst. Phys./Acoust. Soc. Amer., 1936.
[2] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. New York: Springer-Verlag, 1991.
[3] L. Cremer, *The Physics of the Violin*. Cambridge, MA: MIT Press, 1984.
[4] A. V. Oppenhim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: PrenticeHall, 1989.
[5] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. New York: Macmillan, 1993.
[6] J. O. Smith, "Physical modeling using digital waveguides," *Comput. Music J.*, vol. 16, no. 4, pp. 74–87, 1992.
[7] ——, "Music Application of Digital Waveguide," Stanford Univ., Stanford, CA, CCRMA Tech. Rep. STAN-M-67.
[8] ——, "Efficient synthesis of stringed musical instruments," in *Proc. 1993 Int. Comput. Music Conf.*, 1993, pp. 64–71.
[9] A. Fettweis, "Wave digital filters: Theory and practice," *Proc. IEEE*, vol. 74, pp. 227–327, Feb. 1986.
[10] A. Sart and G. De Poli, "Toward nonlinear wave digital filters," *IEEE. Trans. Signal Processing*, vol. 47, pp. 597–, Sept. 1999.
[11] V. Duyne *et al.*, "The 3-D tetrahedral digital waveguide with musical applications," in *Proc. 1996 ICMC*, Hong Kong, Aug. 1996, pp. 9–16.
[12] V. Välimäki *et al.*, "Physical modeling of plucked string instruments with application to real-time sound synthesis," *J. Audio Eng. Soc.*, vol. 44, no. 5, pp. 331–353, 1996.
[13] M. Karjalainen *et al.*, "Plucked-string models: From the Karplus-Strong algorithm to digital waveguides and beyond," *Comput. Music J.*, vol. 22, no. 3, pp. 17–32, 1998.
[14] A. W. Su and S. F. Liang, "Synthesis of plucked-string tones by physical modeling with recurrent neural networks," *Proc. IEEE 1997 Workshop Multimedia Signal Processing*, pp. 71–76, June 1997.
[15] S. F. Liang, A. W. Su, and C. T. Lin, "Model-based synthesis of plucked string instruments by using a class of scattering recurrent networks," *IEEE Trans. Neural Networks*, vol. 11, pp. 171–185, Jan. 2000.
[16] F. J. Pineda, "Recurrent backpropagation and the dynamical approach to adaptive neural computation," *Neural Comput.*, vol. 1, pp. 161–172, 1989.
[17] H. Krauß and R. Rabenstein, "Application of multidimensional wave digital filters to boundary value problems," *IEEE Signal Processing Lett.*, vol. 2, pp. 183–187, July 1995.
[18] N. K. Treadgold and T. D. Gedeon, "Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm," *IEEE Trans. on Neural Networks*, vol. 9, no. 4, pp. 662–668, 1998.
[19] L. E. Kinsler *et al.*, *Fundamentals of Acoustics*, 3rd ed. New York: Wiley, 1982.
[20] R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Comput.*, vol. 2, pp. 490–501, 1990.
[21] S. Haykin, *Neural Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[22] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. ICNN93*, San Francisco, CA, 1993, pp. 586–591.
[23] H. Szu, "Fast simulated annealing," in *Neural Networks for Computing*, J. S. Denker, Ed. New York: Amer. Inst. Phys., 1986, pp. 420–425.
[24] H. D. Bodman, *Chinese Musical Iconography: A History of Musical Instrument Depicted in Chinese Art*. Taipei, Taiwan, R.O.C.: Asian-Pacific Cultural Center, 1987.
[25] S. Feng, "Some acoustical measurements on the chinese musical instrument p'i-p'a," *J. Acoust. Soc. Amer.*, vol. 75, no. 2, pp. 599–602, 1984.

Alvin W. Y. Su (M'97) was born in Taiwan in 1964. He received the B.S. degrees in control engineering from National Chiao-Tung University (NCTU), Taiwan, in 1986. He received the M.S. and Ph.D. degrees in electrical engineering from Polytechnic University, Brooklyn, NY, in 1990 and 1993, respectively.

From 1993 to 1994, he was with Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, Stanford, CA. From 1994 to 1995, he was with Computer Communication Lab of the Industrial Technology Research Institute (CCL. ITRI.), Taiwan. In 1995, he joined the Department of Information Engineering and Computer Engineering at Chung-Hwa University, where he serves as an Associate Professor. In 2001, he joined the Department of Computer Science and Information Engineering, National Cheng-Kung University. His research interests include digital audio signal processing, physical modeling of acoustic musical instruments, human computer interface design, video and color image signal processing, and VLSI signal processing.

Dr. Su is a Member of IEEE Computer Society and Signal Processing Society. He is also a Member of Acoustical Society of America and Audio Engineering Society.

Sheng-Fu Liang was born in Tainan, Taiwan, in 1971. He received the B.S. and M.S. degrees in control engineering from the National Chiao-Tung University (NCTU), Taiwan, in 1994 and 1996, respectively. He received the Ph.D. degree in electrical and control engineering from NCTU in 2000.

Currently, he is a Research Assistant Professor in electrical and control engineering at NCTU. His research activities include model-based music synthesis, neural networks, and image processing. His current projects include audio processing and video signal processing.